Технология комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения

Под общей редакцией
доктора технических наук
профессора В.В. Голенкова

Минск
«Бестпринт»
2023

УДК 004.8

Технология комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения / под. ред. В. В. Голенкова – Минск : Бестпринт, 2023. – 1064 с. – ISBN 978-985-7267-25-5.

В издании представлено описание текущей версии открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем (Технологии OSTIS). Предложена стандартизация интеллектуальных компьютерных систем, а также стандартизация методов и средств их проектирования, что является важнейшим фактором, обеспечивающим семантическую совместимость интеллектуальных компьютерных систем и их компонентов, что существенное снижение трудоемкости разработки таких систем.

Книга предназначена всем, кто интересуется проблемами искусственного интеллекта, а также специалистам в области интеллектуальных компьютерных систем и инженерии знаний. Может быть использована студентами, магистрантами и аспирантами специальности «Искусственный интеллект».

Табл. 8. Ил. 223. Библиогр.: 665 назв.

Рекомендовано Советом БГУИР, протокол № 2 от 23.09.2022 г.

Рецензенты:
профессор кафедры цифровой экономики экономического факультета Белорусского государственного университета, доктор техн. наук, проф. *Б.Н. Паньшин;*

главный научный сотрудник отдела совместных программ космических и информационных технологий государственного научного учреждения «Объединенный институт проблем информатики Национальной академии наук Беларуси»
доктор техн. наук, проф. *С.Ф. Липницкий*

ISBN 978-985-7267-25-5

Оглавление

Предисловие

В настоящее время практически для любых государств является актуальным рассмотрение проблем обеспечения технологического суверенитета.

Всеобщая цифровизация потребовала массовой разработки софта и в программисты массово хлынула молодежь, начиная уже со школьной скамьи.

Действительно, зачем толковым школьникам идти на инженерные, естественнонаучные, конструкторские, сложносистемные специальности, если можно быстро вписаться в "технологический конвейер" разработки софта и гарантированно получать зарплату, которая намного выше зарплаты инженеров, конструкторов, проектировщиков сложных систем? И этот технологический конвейер быстро и массово превращает, среднестатистического программиста в ремесленника, даже, наверное, более точно в IT-пролетариат. Причем это не локальный IT-пролетариат, а глобальный IT-пролетариат (можно сказать даже глобально управляемый), формируемый по клише "граждан мира". Хотя, с резким торможением глобализации перспективы гражданства мира стали не такими уж и очевидными и ясными.

Известный английский историк А.Дж.Тойнби в свое время ввел понятие "опасные классы". Он использовал его при описании процессов конца 18-начала 19 веков, когда вырванное из сельской местности население превратилось в "опасные классы". За десятилетия удалось их превратить в более-менее устойчивый пролетариат. Возможно сейчас процессы с опасными классами повторяются, в частности, с айтишниками, которые оказываются мало приспособленными к реалиям современной жизни. Но глобализация с цифровизацией превратила их в "граждан мира" (как им очень хотелось), но в виде недоформированного IT-пролетариата (что их жестко и неприятно приземлило).

Сейчас затруднительно адекватно использовать термины айтишник, программист. Всеобщая цифровизация практически быстро размывает эти понятия. На постсоветском пространстве понятие айтишник, а иногда даже и программист часто рассматривается как нечто единое, объединяющее и тестировщиков, и кодировщиков, и специалистов по искусственному интеллекиу, анализу больших данных, и системных архитекторов и так далее. В современной мировой практике этой цельности нет – все это совершенно разные категории специалистов. Поэтому, когда иногда говорят о новом классе – айтишников, то такого класса строго говоря и нет. Есть, наверное, достаточно массовый айтишный пролетариат (IT-пролетариат), которому очень хотелось бы идентифицировать себя с хайтеком. Но, если специалисты в области искусственного интеллекта, анализа больших данных, системные архитекторы, бизнес-аналитики, получившие серьезное образование, будут востребованы, то профессии тестировщика и кодировщика и близкие к ним, но с модными названиями могут быстро исчезнуть. В частности, искусственный интеллект способен этому сильно помочь. А своего рода миф о том, что нужно так много программистов отчасти поддерживается тем, что просто создается очень много некачественного кода (софта).

Мы всегда должны помнить, что в эпоху всеобщей цифровизации роль технологического суверенитета резко возрастает. Но без молодежи невозможно развивать технологический суверенитет. А молодежь мы упускаем. То есть молодежь вроде бы и стремится к новым технологиям, мы вроде бы и поощряем такое стремление, но этот процесс какой-то однобокий. Возьмем опять IT-сферу на постсоветском пространстве.

В университетах, где готовят IT-специалистов, много учебных центров и классов по продуктам Microsoft, Oracle, Cisco, SAP и тому подобные. Но это не научно-исследовательские центры, а банальные центры навыков, компетенций по простому кодированию, тестированию, внедрению и эксплуатации. Конечно, это неплохо, если бы в итоге не подменялись основные университетские курсы. Студенты видели, что в таких центрах разработки не нужны серьезная математика, физика. Терялась мотивация. Наиболее способные молодые специалисты плавно перетекали из центров разработки в центры за пределами страны, где этапы жизненных циклов разработки более интересны. Как правило, они не возвращались. Например, для нашей небольшой страны это существенные потери. Как выясняется, нередко в подобной эмиграции главное даже не заработки, а возможность творчества вместо ремесла и промышленного программирования. По сути, такая модель ускорила разрушение инженерного, конструкторского, общесистемного образования. Утрачивалась мотивация к изучению математических, физических, сложносистемных дисциплин. Действительно, студенту незачем тратить время на их глубокое освоение, если достаточно получить навыки по кодированию и тестированию софта на основе несложных технологических платформ и легко найти работу в одной из конвейерных компаний по разработке софта. По сути образование подстраивалось под такую модель и планомерно разрушалось при активном участии тех же западных конвейеров.

Вспомним, что традиционные западные обыватели мало интересуются глобальными вещами. Их интерес в основном локален. Например, решения муниципальных властей для них интереснее решений властей штата или земли, не говоря уже о решениях федеральных властей. Конечно могут быть исключения, но они лишь, как обычно,

подтверждают общность. Точно таким же формируется у нас и молодое поколение, особенно айтишное. Действительно система образования и воспитания не воспроизводит сейчас системность во взглядах и оценках. Хорошая системность предполагает умение анализировать и синтезировать информацию, в том числе и по-глобальному. Советская система образования и воспитания, пусть и несколько односторонне, все-таки этому способствовала. Теперь же, в связи со старением и уходом поколений-носителей такой концепции, такая система образования и воспитания полностью исчезает. А новая не создана, вернее лихорадочно, бессистемно создается как ухудшенная калька с быстроразрушающейся западной системы, причем, зачастую, даже не с исконно западной, а как калька с восточно-европейской кальки. И у молодежи формируется основная парадигма, зачем думать о глобальном, системно, главное, чтобы в локальном мирке было все хорошо и как следствие нынешние молодежные установки: "жизнь в моменте", "хорошо в моменте", "хорошо здесь и сейчас" и тому подобное.

И именно айтишный пролетариат являются основным носителем таких установок. И понятно почему. Они работают практически полностью в западном (американском) конвейере, причем выполняя в основном самые примитивные операции, то есть работают как современный IT-пролетариат, даже не как инженера, технологи, не говоря уже о конструкторах или архитекторах. То есть совершенно не элита.

По такой модели мы получили лишь ускорение потери суверенитета технологического, и как следствие разрушение странового суверенитета.

Хотя последние несколько лет наконец-то многие в мире спохватились, что нужен суверенитет для стран. Глобалистский мир оказался не столь уж безопасен и привлекателен, как был распиарен за последние лет тридцать, граждане мира, как мировые кочевники, не столь уж и конструктивны и креативны – собственно, все как всегда: хороша только виртуальная картинка, в существенно иных условиях идеи глобализма могли бы быть и продуктивны, но народ не тот, элиты не те и тому подобное. Модно стало говорить о цифровом суверенитете, суверенитете в области высоких технологий, суверенитете в области инфокоммуникационных технологий и так далее. Соответственно, откликаясь на моду, хайп (а сейчас многое пытаются делать на волнах хайпа) стали писать концепции и стратегии для разного вида суверенитета. Хотя после стольких лет построения глобалисткого мира (как видим теперь не очень-то удачного), строить суверенитеты многим странам крайне затруднительно.

Для небольших стран хотелось бы отметить следующее.

В условиях достаточно жесткого противостояния и противоборства нет смысла стремится к топовым с точки зрения глобальных экспертов, существующих, как обычно, на средства транснациональных компаний, решениям в области инфокоммуникационных технологий. Те же топовые чипы 2-3 нано зачем нужны на данном этапе для оборонки, государства, индустрии, социальной сферы и тому подобное. Можно и 90 нано обойтись. Не нужно путать требования потребительского рынка и требования для решения государственных, страновых задач. Тот же Apple, или Microsoft, или Samsung, создавая гаджеты и сервисы для конечного потребителя решают совершенно иные задачи. Им нужно удержать лидерство на рынке, чтобы сохранить миллиардные прибыли, Отсюда и 2-3 нано в чипах, и все новые и новые операционные системы, и все новые и новые приложения. Бег по кругу, новый системный и прикладной софт требует нового железа, новое железо требует нового софта – и за все расплачивается конечный потребитель. А решение задачи сохранения суверенитета страны в текущей ситуации такой безудержной гонки совершенно и не требует. И чипы нужны попроще, но понадежнее, и операционные системы нужны не навороченные, и приложения нужны не такие уж многофункциональные. Классический пример с MS Word. Обычный пользователь использует не более 5% возможностей редактора. А, например, госслужащим и им подобным и 5% много. Точно также и с большинством софта. Нужны гораздо более простые, но стабильные и надежные системы. И тогда, чтобы создать такие системы нужны не сотни тысяч IT-пролетариата, а сотни, ну может быть тысячи высококвалифицированных разработчиков и относительно небольшое количество IT-пролетариев. Еще раз подчеркну, страна для своего суверенитета и транснациональный IT-бизнес решают совершенно разные задачи и преследуют совершенно разные цели. Можно еще напомнить, что массы IT-пролетариата бизнесу нужны и для того, чтобы раздувать значимость фирмы, создавать иллюзию гигантизма, особенно, если твоя фирма на IPO в Нью-Йорке или Лондоне, или хотя бы в Гонконге. Массовый инвестор больше поверит в фирму с десятками тысяч работников, чем в небольшую команду.

Но коль уж мы от цифровизации никуда не денемся, то нужна цифровизация хорошо продуманная, просчитанная, с глубоким прогнозированием последствий, а не безудержная, с агрессивной рекламой, что создает у людей совершенно ложные ориентиры. В подавляющем случае это делается опять же непрофессионально айтишным пролетариатом, втянутым в разработку на волне хайпа о неограниченной цифровизации всего и вся. А для того чтобы это понимать, молодежи нужно соответствующее образование. И выбора нет – либо мы такое образование очень оперативно сформируем, либо окончательно потеряем думающую молодежь, и, соответственно, окончательно потеряем и суверенитет в условиях глобального перехода информационных технологий на принципиально новый уровень, требующий создания комплексов эффективно взаимодействующих компьютерных систем.

А. Н. Курбацкий
доктор технических наук, профессор,
заведующий кафедрой технологий программирования
факультета прикладной математики и информатики
Белорусского государственного университета

Авторское предисловие

Ключевой особенностью ***компьютерных систем нового поколения*** является их ***интероперабельность***, то есть способность к эффективному осознанному взаимодействию, необходимым условием которой является способность к взаимопониманию, то есть ***семантическая совместимость***. Таким образом, каждая *компьютерная система нового поколения* должна:

- знать свои обязанности и возможности;
- уметь координировать свои действия с другими компьютерными системами нового поколения в ходе коллективного решения сложных задач в непредусмотренных (нештатных) обстоятельствах.

Переход от современных компьютерных систем к компьютерным системам нового поколения, которые, очевидно, должны обладать достаточно высоким уровнем ***интеллекта***, означает переход к принципиально новому технологическому укладу в области автоматизации различных видов человеческой деятельности и предполагает переосмысление и использование всего опыта, накопленного при разработке и эксплуатации различных *компьютерных систем*. Участие в проекте создания комплексной ***Технологии поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*** не требует от специалистов радикального изменения области их научных интересов. Требуется просто учет дополнительных требований к формализации своих результатов. Основным из этих требований является ***семантическая совместимость*** с другими (смежными) результатами.

К настоящему моменту

- Завершен первый этап разработки *Технологии поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*, которая названа нами ***Технологией OSTIS*** (Open Semantic Technology for Intelligent Systems). Началом указанного первого этапа является Первая конференция OSTIS (Минск, 10 – 12 февраля 2011 года).
- Сформировался работоспособный стартовый авторский коллектив в рамках созданного Учебно-научного объединения по Искусственному интеллекту, в состав которого входят ведущие университеты Республики Беларусь (в четырех из которых открыта специальность “Искусственный интеллект”), и ряд других организаций.

Результаты первого этапа разработки *Технологии OSTIS* отражены в материалах проведенных ***конференций OSTIS*** и в первой версии ***Метасистемы OSTIS***, которая непосредственно и осуществляет автоматизацию проектирования, реализации, реинжиниринга и эксплуатации ***интеллектуальных компьютерных систем нового поколения*** и гарантирует поддержку их ***семантической совместимости*** и интероперабельности не только на этапе проектирования, но и в ходе их эксплуатации.

На следующем этапе разработки *Технологии поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* требуется существенное расширение фронта работ и соответствующего авторского коллектива. Этому, в частности, была посвящена *Конференция OSTIS-2022* (24-26 ноября 2022 года), которая внесла важный вклад в развитие открытого проекта развития технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (*Проекта OSTIS*). Предлагаемая вашему вниманию монография является результатом работы указанного авторского коллектива.

Главным адресатом монографии является широкий круг лиц (включая студентов), которые хотят за короткое время приобрести знания и навыки, для разработки *прикладных интеллектуальных компьютерных систем нового поколения*. Для такого контингента читателей необходима четко структурированная документация, отражающая текущее состояние соответствующей технологии.

Вторым контингентом читателей являются специалисты в области современных информационных технологий (и, в том числе, технологий Искусственного интеллекта), желающие стать участниками дальнейшей эволюции комплексной технологии создания и сопровождения интеллектуальных компьютерных систем нового поколения — соавторами последующих версий указанной технологии, которая должна носить открытый характер. Для такого контингента читателей требуется:

- достаточно подробное обоснование принципов, лежащих в основе текущей версии указанной технологии;
- сравнительный анализ указанных принципов с известными альтернативными подходами;
- четкая формулировка проблем текущего состояния рассматриваемой технологии и направлений ее дальнейшего развития.

Написание данной монографии с ориентацией сразу на две указанные выше категории читателей создает благоприятные условия для последующего перехода инженеров прикладных интеллектуальных компьютерных систем нового поколения в категорию соавторов последующих версий соответствующей технологии, что является очень важным фактором ускорения темпов эволюции этой технологии благодаря глубокой ***конвергенции*** инженерной деятельности, научно-исследовательской деятельности и деятельности по развитию технологии.

Часть 1.

Введение в интеллектуальные компьютерные системы нового поколения и технологию комплексной поддержки их жизненного цикла

⇒ *аннотация**:
[Уточнение понятия интеллектуальной компьютерной системы. Требования, предъявляемые к интеллектуальным компьютерным системам следующего (нового) поколения и принципы, лежащие в их основе. Структура и особенности жизненного цикла интеллектуальных компьютерных систем нового поколения. Технология комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (*Технология OSTIS* — Open Semantic Technology for Intelligent Systems).**]**

⇒ *подраздел**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

Глава 1.1.

Факторы, определяющие уровень интеллекта кибернетических систем

⇒ *автор**:
- *Загорский А. Г.*
- *Голенков В. В.*
- *Шункевич Д. В.*

⇒ *аннотация**:
[Рассмотрена иерархическая система свойств (в том числе способностей) *кибернетических систем*, определяющих их качество и позволяющих сформулировать требования, которым должна удовлетворять *высокоинтеллектуальная система* (кибернетическая система с сильным интеллектом).]

⇒ *подраздел**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*
- *§ 1.1.3. Направления эволюции компьютерных систем*

§ 1.1.1. Понятие интеллектуальной кибернетической системы

⇒ *подраздел**:
- *Пункт 1.1.1.1. Типология кибернетических систем*
- *Пункт 1.1.1.2. Структура кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*
- *Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы*
- *Пункт 1.1.1.5. Комплекс свойств, определяющих качество информации, хранимой в памяти кибернетической системы*
- *Пункт 1.1.1.6. Комплекс свойств, определяющих качество решателя задач кибернетической системы*
- *Пункт 1.1.1.7. Комплекс свойств, определяющих уровень обучаемости кибернетической системы*
- *Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы*

⇒ *ключевое понятие**:
- *кибернетическая система*
- *качество кибернетических систем^*
- *уровень интеллекта кибернетических систем^*
- *интеллектуальная кибернетическая система*

⇒ *ключевое знание**:
- *Классификация кибернетических систем*
- *Обобщенная архитектура кибернетических систем*
- *Факторы, определяющие качество кибернетических систем*
- *Факторы, определяющие уровень интеллекта кибернетических систем*

⇒ *библиографическая ссылка**:
- *Глушков В.М.Кибер-1979ст*
- *Варшавский В.А..ОркесИБДР-1984кн*
- *Sherif Y.S..CompuSDQAMaM-1988art*
- *Ouksel A.M..SemanIiGIS-1999art*
- *Fry R.L.tEngin oCS-2002art*
- *Gao R..PerfoMfISaE-2002art*
- *Стефанюк.В.Л.ЛокалОИСМ-2004кн*
- *Nilsson N.J.HumanLAIBS-2005art*
- *Kerr C..aConceMfTI-2006art*
- *Hamilton Inter..Intero aKEftG-2006art*
- *Laird J.E..Claim aCiEHLIS-2009art*
- *Анцыферов С.С.ОценкКИС-2013art*

- *Neiva F.W..TowarPItSC-2016art*
- *Melekhova O..aDecenSfCD-2018art*
- *Cho J..StramMtToCBS-2019art*
- *Finn V.K.IntelIOGZiO-2021bk*
- *Lopes L.S..SemanIitIoT-2022art*
- *Zagorskiy A..Facto tDtLoIoCS-2022art*

Введение в § 1.1.1.

Уровень качества *кибернетических систем* определяется достаточно большим набором свойств (параметров, характеристик) кибернетических систем, каждое из которых определяет уровень качества *кибернетической системы* в соответствующем аспекте (ракурсе), указывая (задавая) уровень развития конкретных способностей и возможностей *кибернетической системы*. При этом важно подчеркнуть следующее:

- существенное значение имеет не столько сам набор свойств, а иерархия этих свойств, позволяющая уточнять направления проявления каждого свойства;
- существенное значение также имеет баланс уровней развития различных свойств — вклад разных свойств, обеспечивающих значение одного и того же свойства более высокого уровня иерархии, а значение этого свойства более высокого уровня может быть разным. Из этого следует, что не всегда следует акцентировать внимание на развитие некоторых свойств (характеристик). Нужен целостный, коллективный подход;
- рассмотренная иерархия свойств *кибернетических систем* является общей как для естественных, так и для *искусственных кибернетических систем*;
- приведенная иерархическая детализация свойств кибернетических систем (с помощью отношения *частное свойство** и отношения *свойство-предпосылка**), определяющих качество таких систем, (1) дает возможность четко определить направления совершенствования (развития) *кибернетических систем* и (2) дает ориентир (систему критериев) для обоснования конкретных предложений по совершенствованию компьютерных систем, а также для сравнения различных альтернативных предположений;
- особое значение для развития кибернетических систем имеют такие их свойства, как стратифицированность, рефлексивность и социализация;
- важное значение имеет не только совершенствование *кибернетических систем* в соответствии с иерархической системой их свойств, но и совершенствование (в том числе, детализация) самой этой иерархической системы свойств.

Свойства (способности), которым должны удовлетворять *интеллектуальные системы*, рассматриваются в целом ряде публикаций. Тем не менее, для практической реализации *компьютерных систем*, обладающих указанными свойствами (способностями), то есть *интеллектуальных компьютерных систем*, необходимо детализировать (уточнить) эти *свойства*, пытаясь свести их к более конструктивным, прозрачным и понятным.

кибернетическая система (см. *Глушков В.М.Кибер-1979ст*) — это система, которая способна управлять своими действиями, адаптируясь к изменениям состояния внешней среды (среды своего "обитания") в целях самосохранения (сохранения своей целостности и "комфортности" существования путем удержания своих "жизненно" важных параметров в определенных рамках "комфортности") и в целях формирования определенных реакций (воздействий на внешнюю среду) в ответ на определенные стимулы (на определенные ситуации или события во внешней среде), а также которая способна (при соответствующем уровне развития) эволюционировать в направлении:

- изучения своей внешней среды как минимум для предсказания последствий своих воздействий на внешнюю среду, а также для предсказания изменений внешней среды, которые не зависят от собственных воздействий;
- изучения самой себя и, в частности, своего взаимодействия с внешней средой;
- создания технологий (методов и средств), обеспечивающих изменение своей внешней среды (условий своего существования) в собственных интересах.

кибернетическая система
:= [адаптивная система]
:= [целенаправленная система]
:= [активный субъект самостоятельной деятельности]
:= [материальная сущность, способная целенаправленно (в своих интересах) воздействовать на среду своего обитания как минимум для сохранения своей целостности, жизнеспособности, безопасности]
⇒ *примечание**:
[Уровень (степень) адаптивности, целенаправленности, активности у систем, основанных на обработке информации может быть самым различным.]
:= [система, организация функционирования которой основано на обработке информации о той среде, в которой существует эта система]

:= [материальная сущность, способная к активной целенаправленной деятельности, которая на определенном уровне развития указанной сущности становится "осмысленной", планируемой, преднамеренной деятельностью]
:= [субъект, способный на самостоятельное выполнение некоторых "внутренних" и "внешних" действий либо порученных извне, либо инициированных самим субъектом]
:= [сущность, способная выполнять роль субъекта деятельности]
:= [естественная или искусственно созданная система, способная мониторить и анализировать свое состояние и состояние окружающей среды, а также способная достаточно активно воздействовать на собственное состояние и на состояние окружающей среды]
:= [система, способная в достаточной степени самостоятельно взаимодействовать со своей средой, решая различные задачи]
:= [система, основанная на обработке информации]

кибернетическая система динамически сопоставляет полученную информацию с выбранными действиями, относящимися к задаче, которая определяет основную цель системы (см. *Fry R.L.tEngin oCS-2002art*).

Пункт 1.1.1.1. Типология кибернетических систем

кибернетическая система
⇒ *разбиение**:
Признак естественности или искусственности кибернетических систем
= {• *естественная кибернетическая система*
:= [кибернетическая система естественного происхождения]
⊃ *человек*
• *компьютерная система*
:= [искусственная кибернетическая система]
:= [кибернетическая система искусственного происхождения]
:= [технически реализованная кибернетическая система]
• *симбиоз естественных и искусственных кибернетических систем*
:= [кибернетическая система, в состав которой входят компоненты как естественного, так и искусственного происхождения]
⊃ *сообщество компьютерных систем и людей*
}

Особенностью *компьютерных систем* является то, что они могут выполнять "роль" не только продуктов соответствующих действий по реализации этих систем, но и сами являются *субъектами**, способными выполнять (автоматизировать) широкий спектр действий. При этом интеллектуализация этих систем существенно расширяет этот спектр.

кибернетическая система
⇒ *разбиение**:
Структурная классификация кибернетических систем
= {• *простая кибернетическая система*
• *индивидуальная кибернетическая система*
• *многоагентая система*
}
⇒ *разбиение**:
Классификация кибернетических систем по признаку наличия надсистемы и роли в рамках этой надсистемы
= {• *кибернетическая система, не являющаяся частью никакой другой кибернетической системы*
:= [кибернетическая система, не имеющая надсистем]
• *кибернетическая система, встроенная в индивидуальную кибернетическую систему*
• *агент многоагентной системы*
:= [кибернетическая система, являющаяся агентом одной или нескольких многоагентных систем]
}

простая кибернетическая система — это *кибернетическая система*, уровень развития которой находится ниже уровня *индивидуальных кибернетических систем* и которая является *специализированным решателем задач*, реализующим (интерпретирующим) чаще всего один *метод решения задач* и, соответственно, решающим только

задачи заданного *класса задач*. *простая кибернетическая система* может быть *компонентом**, встроенным в *индивидуальную кибернетическую систему*, а также может быть *агентом* многоагентной системы*, являющейся коллективом из *простых кибернетических систем*.

индивидуальная кибернетическая система — условно выделенный уровень развития *кибернетических систем*, в основе которого лежит переход от *специализированного решателя задач* к *индивидуальному решателю задач*, обеспечивающему интерпретацию произвольного (нефиксированного) набора *методов* (программ) решения задач при условии, если эти *методы* введены (загружены, записаны) в *память кибернетической системы*. Признаками *индивидуальных кибернетических систем* являются:

- наличие *памяти*, предназначенной для хранения как минимум интерпретируемых *методов* (программ) и обеспечивающей корректировку (редактирование) хранимых *методов*, а также их удаление из *памяти* и ввод (запись) в *память* новых *методов*;
- легкая возможность "перепрограммировать" *кибернетическую систему* на решение других задач, что обеспечивается наличием *универсальной модели решения задач* и, соответственно, *универсальным интерпретатором любых моделей*, представленных (записанных) на соответствующем *языке*;
- наличие пусть даже простых средств коммуникации (обмена информацией) с другими *кибернетическими системами* (например, с людьми);
- способность входить в различные *коллективы кибернетических систем*.

Класс *индивидуальных кибернетических систем* — это определенный этап эволюции кибернетических систем, означающий переход к кибернетическим системам, которые способны самостоятельно "выживать". *индивидуальная кибернетическая система* может быть агентом (членом) *многоагентной системы* (членом коллектива индивидуальных кибернетических систем), но некоторые многоагентные системы могут состоять из агентов, не являющихся *индивидуальными кибернетическими системами*, представляющих собой простые специализированные кибернетические системы, выполняющие достаточно простые действия (см. *Стефанюк.В.Л.ЛокалОИСМ-2004кн*).

Примерами *кибернетической системы*, встроенной в *индивидуальную кибернетическую систему*, являются *sc-агент ostis-системы* и *решатель задач ostis-системы*.

многоагентная система представляет собой коллектив взаимодействующих автономных *кибернетических систем*, имеющих общую среду обитания (жизнедеятельности).

многоагентная система
⇒ *разбиение**:
{• *коллектив из простых кибернетических систем*
• *коллектив индивидуальных кибернетических систем*
• *коллектив индивидуальных и простых кибернетических систем*
}
⇒ *разбиение**:
{• *одноуровневый коллектив кибернетических систем*
:= [многоагентная система, агентами которой не могут быть многоагентные системы]
• *иерархический коллектив кибернетических систем*
:= [многоагентная система, по крайней мере одним агентом которой является многоагентная система]
}

коллектив индивидуальных кибернетических систем — многоагентная система, агентами (членами) которой являются *индивидуальные кибернетические системы*. Примерами *коллектива индивидуальных кибернетических систем* могут быть как коллективы людей, так и коллективы компьютерных систем и людей.

одноуровневый коллектив кибернетических систем определен как специализированное средство решения задач, реализующее либо одну модель параллельного (распределенного) решения задач соответствующего класса, либо комбинацию фиксированного числа разных и параллельно реализованных *моделей решения задач*.

Агентами ***иерархического коллектива кибернетических систем*** могут быть:

- *индивидуальные кибернетические системы*;
- *коллективы индивидуальных кибернетических систем*;
- *коллективы, состоящие из индивидуальных кибернетических систем и коллективов индивидуальных кибернетических систем* и так далее.

Пункт 1.1.1.2. Структура кибернетической системы

Рассмотрим структуру *кибернетической системы*.

кибернетическая система
⇒ *обобщенная декомпозиция**:
{• *информация, хранимая в памяти кибернетической системы*
• *абстрактная память кибернетической системы*
• *решатель задач кибернетической системы*
• *физическая оболочка кибернетической системы*
}

информация, хранимая в памяти кибернетической системы, представляет собой информационную модель среды, в которой действует (существует, функционирует) эта *кибернетическая система*, текущее состояние памяти *кибернетической системы*.

абстрактная память кибернетической системы — это внутренняя абстрактная информационная среда *кибернетической системы*, представляющая собой динамическую *информационную конструкцию*, каждое состояние которой есть не что иное, как *информация, хранимая в памяти кибернетической системы* в соответствующий момент времени. *абстрактная память кибернетической системы* является подмножеством динамической информационной конструкции.

решателем задач кибернетической системы называют совокупность всех навыков (умений), приобретенных *кибернетической системо*й к рассматриваемому моменту. Встроенный в *кибернетическую систему* субъект, способный выполнять целенаправленные ("осознанные") действия во внешней среде этой *кибернетической системы*, а также в ее внутренней среде (в абстрактной памяти).

Рассмотрим подробнее понятие ***действия кибернетической системы***.

действие кибернетической системы
⊂ *действие*
:= [целенаправленное действие, выполняемое кибернетической системой, а точнее, ее решателем задач]
⇒ *разбиение**:
{• *внешнее действие кибернетической системы*
:= [действие, выполняемое кибернетической системой в ее внешней среде]
:= [поведенческое действие]
• *действие кибернетической системы, выполняемое в собственной физической оболочке*
• *действие кибернетической системы, выполняемое в собственной абстрактной памяти*
}

Говоря о ***действиях кибернетической системы, выполняемых в собственной абстрактной памяти***, подразумеваются действия, направленные на преобразование информации, хранимой в памяти, но никак не на преобразование физической памяти (физической оболочки абстрактной памяти).

Каждое ***сложное действие, выполняемое кибернетической системой*** вне собственный абстрактной памяти, включает в себя поддействия, выполняемые в указанной *абстрактной памяти кибернетической системы*. Это означает, что все *внешние действия кибернетической системы* управляются внутренними ее действиями (действиями в абстрактной памяти).

интерфейс кибернетической системы — условно выделяемый компонент *решателя задач кибернетической системы*, обеспечивающий решение *интерфейсных задач*, направленных на непосредственную реализацию взаимодействия *кибернетической системы* с ее *внешней средой*. При этом следует отличать *интерфейс кибернетической системы* и *физическое обеспечение интерфейса кибернетической системы*.

физическая оболочка кибернетической системы — часть *кибернетической системы*, являющаяся "посредником" между ее внутренней средой (памятью, в которой хранится и обрабатывается информация кибернетической системы) и ее внешней средой.

кибернетическая система
⇒ *обобщенная декомпозиция**:
{• *память кибернетической системы*
• *процессор кибернетической системы*
• *физическое обеспечение интерфейса кибернетической системы*
• *корпус кибернетической системы*
}

физическое обеспечение интерфейса кибернетической системы
⇒ *обобщенная декомпозиция**:
{• *сенсорная подсистема физической оболочки кибернетической системы*

- *эффекторная подсистема физической оболочки кибернетической системы*

}

память кибернетической системы — физическая оболочка (реализация) абстрактной *памяти кибернетической системы*, внутренней среды *кибернетической системы*, в рамках которой *кибернетическая система* формирует и использует (обрабатывает) информационную модель своей внешней среды.

Не каждая *кибернетическая система* имеет *память*. В *кибернетических системах*, которые не имеют *памяти*, обработка информации сводится к обмену сигналами между компонентами этих систем. Появление в *кибернетических системах* памяти как среды для "централизованного" хранения и обработки *информации* является важнейшим этапом их эволюции. Дальнейшая эволюция *кибернетических систем* во многом определяется:

- *качеством памяти* как среды для хранения и обработки информации;
- качеством информации (информационной модели), хранимой в *памяти кибернетической системы*;

Компонент *кибернетической системы*, в рамках которого *кибернетическая система* осуществляет отображение (формирование информационной модели) среды своего существования, а также использование этой информационной модели для управления собственным поведением в указанной среде.

Сам факт появления в *кибернетической системе* памяти, которая (1) обеспечивает представление различного вида информации о среде, в рамках которой *кибернетическая система* решает различные задачи (выполняет различные действия), (2) обеспечивает хранение достаточно полной информационной модели указанной среды (достаточно полной для реализации своей деятельности), (3) обеспечивает высокую степень гибкости указанной хранимой в памяти информационной модели среды жизнедеятельности (то есть легкость внесения изменений в эту информационную модель), существенно повышает уровень адаптивности *кибернетической системы* к различным изменениям своей среды. Появление *памяти* в кибернетических системах является основным признаком перехода от "простых" автоматов к компьютерным системам, от роботов 1-го поколения к роботам следующих поколений.

Принципы организации *памяти кибернетической системы* могут быть разными (ассоциативная, адресная, структурно фиксированная/структурно перестраиваемая, нелинейная/линейная). От организации *памяти кибернетической системы* во многом зависит ее качество.

процессором кибернетической системы называют физически реализованный интерпретатор хранимых в *памяти кибернетической системы программ*, соответствующих базовой *модели решения задач* для данной *кибернетической системы*. Такая *модель решения задач* для данной *кибернетической системы* является *моделью решения задач* самого нижнего уровня и не может быть интерпретирована с помощью другой *модели решения задач*, используемой этой же *кибернетической системой*. Она может быть проинтерпретирована либо путем аппаратной реализации такого интерпретатора, либо путем его программной реализации, например, на современных компьютерах. В последнем случае, кроме собственного интерпретатора, необходимо также построить модель *памяти* реализуемой кибернетической системы.

Развитие рынка *интеллектуальных компьютерных систем* существенно сдерживается неприспособленностью современного поколения компьютеров к реализации на их основе *интеллектуальных компьютерных систем*. Попытки создания компьютеров, приспособленных к реализации *интеллектуальных компьютерных систем*, не привели к успеху, так как эти проекты были направлены на выполнение частных требований, предъявляемых к аппаратному уровню *интеллектуальных компьютерных систем*, что неминуемо приводило к приспособленности создаваемых компьютеров к реализации не всего многообразия *интеллектуальных компьютерных систем*, а только некоторых подмножеств таких систем. Указанные подмножества *интеллектуальных компьютерных систем* в основном определялись ориентацией на конкретные используемые *модели решения задач*, тогда как важнейшим фактором, определяющим уровень *интеллекта кибернетических систем*, является их универсальность в плане многообразия используемых *моделей решения задач*. Следовательно, компьютер для *интеллектуальных компьютерных систем* должен быть эффективным аппаратным интерпретатором любых *моделей решения задач*, как интеллектуальных, так и достаточно простых.

Таким образом, выделено следующее семейство отношений, заданных на множестве *кибернетических систем*.

отношение, заданное на множестве кибернетических систем

- ∋ *память кибернетической системы**
- ∋ *процессор кибернетической системы**
- ∋ *член коллектива**
- ∋ *внешняя среда кибернетической системы**
- ∋ *сенсор кибернетической системы**
- ∋ *эффектор кибернетической системы**
- ∋ *физическая оболочка кибернетической системы**
- ∋ *информация, хранимая в памяти кибернетической системы**
- ∋ *абстрактная память кибернетической системы**
- ∋ *часть**

⊃ *встроенная кибернетическая система**

Понятие *внешней среды кибернетической системы** является понятием относительным, так как (1) разные *кибернетические системы* могут иметь разную внешнюю среду и (2) одна *кибернетическая система* может входить в состав внешней среды другой кибернетической системы. В общем случае среда жизнедеятельности *кибернетической системы* включает в себя (1) *внешнюю среду** этой системы, (2) *физическую оболочку** этой системы и (3) ее *абстрактную память*, то есть *внутреннюю среду**, которая является хранилищем информационной модели всей среды.

Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы

Интегральная, комплексная оценка уровня развития *кибернетической системы* определяется как ***качество кибернетической системы***. Это свойство, характеристика *кибернетических систем*, признак их классификации, который позволяет разместить эти системы по "ступенькам" некоторой условной "эволюционной лестницы". На каждую такую "ступеньку" попадают *кибернетические системы*, имеющие одинаковый уровень развития, каждому из которых соответствует свой набор значений дополнительных свойств *кибернетических систем*, которые уточняют соответствующий уровень развития *кибернетических систем*. Такой эволюционный подход к рассмотрению *кибернетических систем* дает возможность, во-первых, детализировать направления эволюции *кибернетических систем* и, во-вторых, уточнить то место этой эволюции, где и благодаря чему осуществляется переход от неинтеллектуальных *кибернетических систем* к интеллектуальным.

В основе эволюционного подхода к рассмотрению многообразия *кибернетических систем* лежит положение о том, что идеальных *кибернетических систем* не существует, но существует постоянное стремление к идеалу, к большему совершенству. Важно уточнить, что конкретно в каждой *кибернетической системе* следует изменить, чтобы привести эту систему к более совершенному виду. Так, например, развитие технологий разработки компьютерных систем должно быть направлено на переход к таким новым архитектурным и функциональным принципам, лежащим в основе *компьютерных систем*, которые обеспечивают существенное снижение трудоемкости их разработки и сокращение сроков разработки, а также обеспечивают существенное повышение уровня интеллекта и, в частности, уровня обучаемости разрабатываемых *компьютерных систем*, например, путем перехода от поддержки обучения с учителем к реализации эффективного самообучения (к автоматизации организации самостоятельного обучения).

Проблема выделения критериев интеллектуальности компьютерных систем рассмотрена в ряде работ: *Finn V.K.IntelIOGZiO-2021bk,Nilsson N.J.HumanLAIBS-2005art,Kerr C..aConceMfTI-2006art,Анцыферов С.С.ОценкКИС-2013art*. Были предложены различные системные показатели для измерения *качества компьютерных систем*. Поскольку системы становятся все более сложными и включают множество подсистем или компонентов, измерение их качества в нескольких измерениях становится сложной задачей (см. *Cho J..StramMtToCBS-2019art*). Метрики качества включают в себя набор мер, которые могут описывать атрибуты системы в терминах, не зависящих от структуры, которая приводит к этим атрибутам. Эти меры должны быть выражены количественно и должны иметь значительный уровень точности и надежности (см. *Sherif Y.S..CompuSDQAMaM-1988art*).

Тем не менее, для практической реализации *интеллектуальных компьютерных систем* необходимо детализировать и уточнить эти свойства, пытаясь свести их к более конструктивным, прозрачным и понятным для реализации свойствам. Для детализации понятия качества кибернетической системы необходимо задать метрику *качества кибернетических систем* и построить иерархическую систему свойств, параметров, признаков, определяющих качество кибернетических систем.

качество кибернетической системы
⇒ *свойство-предпосылка**:
- *качество физической оболочки кибернетической системы*
- *качество информации, хранимой в памяти кибернетической системы*
- *качество решателя задач кибернетической системы*
- *гибридность кибернетической системы*
  ⇒ *частное свойство**:
  {• *многообразие видов знаний, хранимых в памяти кибернетической системы*
  • *многообразие моделей решения задач*
  • *многообразие видов сенсоров и эффекторов*
  }
- *приспособленность кибернетической системы к ее совершенствованию*
- *производительность кибернетической системы*

- *надежность кибернетической системы*
- *интероперабельность кибернетической системы*

Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы

качество физической оболочки кибернетической системы определяется как интегральное качество физической, аппаратной основы *кибернетической системы*. Выделенное множество свойств, определяющих *качество физической оболочки кибернетической системы*, приведена ниже:

качество физической оболочки кибернетической системы
$\Rightarrow$ *свойство-предпосылка**:
- *качество памяти кибернетической системы*
- *качество процессора кибернетической системы*
- *качество сенсоров кибернетической системы*
- *качество эффекторов кибернетической системы*
- *приспособленность физической оболочки кибернетической системы к ее совершенствованию*
- *удобство транспортировки кибернетической системы*
- *надежность физической оболочки кибернетической системы*

Важно, чтобы память обеспечивала высокий уровень гибкости указанной информационной модели. Важно также, чтобы эта информационная модель была моделью не только *внешней среды кибернетической системы*, но также и моделью самой этой информационной модели — описанием ее текущей ситуации, предыстории, закономерностей.

качество памяти кибернетической системы
$\Rightarrow$ *свойство-предпосылка**:
- *способность памяти кибернетической системы обеспечить хранение высококачественной информации*
- *способность памяти кибернетической системы обеспечить функционирование высококачественного решателя задач*
- *объем памяти*

Факт возникновения *памяти в кибернетической системе* является важнейшим этапом ее эволюции. Дальнейшее развитие памяти кибернетической системы, обеспечивающее хранение все более качественной информации, хранимой в памяти и все более качественную организацию обработки этой информации, то есть переход на поддержку все более качественных моделей обработки информации, является важнейшим фактором эволюции *кибернетических систем*.

способность памяти кибернетической системы обеспечить функционирование высококачественного решателя задач основывается на качестве доступа к информации, хранимой в *памяти кибернетической системы*, логико-семантической гибкости *памяти кибернетической системы*, способности *памяти кибернетической системы* обеспечить интерпретацию широкого многообразия *моделей решения задач*.

качество процессора кибернетической системы определяется его способностью обеспечить функционирования высококачественного *решателя задач*.

качество процессора кибернетической системы
$\Rightarrow$ *свойство-предпосылка**:
- *многообразие моделей решения задач, интерпретируемых процессором кибернетической системы*
- *простота и качество интерпретации процессором системы широкого многообразия моделей решения задач*
- *обеспечение процессором кибернетической системы качественного управления информационными процессами в памяти*
- *быстродействие процессора кибернетической системы*

Максимальным уровнем *качества процессора кибернетической системы* по параметру ***многообразия моделей решения задач, интерпретируемых процессором кибернетической системы***, является его универсальность, то есть его принципиальная возможность интерпретировать любую модель решения как интеллектуальных, так и неинтеллектуальных задач. Простота определяется степенью близости интерпретируемых моделей решения задач к "физическому" уровню организации процессора кибернетической системы. Качественное управление инфор-

мационными процессами в памяти подразумевает грамотное сочетание таких аспектов управление процессами, как централизация и децентрализация (см. *Melekhova O..aDecenSfCD-2018art*), синхронность и асинхронность, последовательность и параллельность.

качество сенсоров и ***эффекторов кибернетической системы*** сводится к многообразию видов сенсоров и эффекторов кибернетической системы, то есть к многообразию средств восприятия и воздействия на информацию о текущем состоянии внешней среды и собственной физической оболочки. ***приспособленность физической оболочки кибернетической системы к ее совершенствованию*** определяется гибкостью и стратифицированностью физической оболочки кибернетической системы.

Пункт 1.1.1.5. Комплекс свойств, определяющих качество информации, хранимой в памяти кибернетической системы

Качество информационной модели среды "обитания" *кибернетической системы*, в частности, определяется:
- корректностью этой модели, отсутствием в ней ошибок;
- адекватностью этой модели;
- полнотой, достаточностью находящейся в ней информации для эффективного функционирования кибернетической системы;
- структурированностью, систематизированностью.

Важнейшим этапом эволюции информационной модели среды *кибернетической системы* является переход от недостаточно полной и несистематизированной информационные модели среды к *базе знаний*.

качество информации, хранимой в памяти кибернетической системы
⇒ *свойство-предпосылка**:
- *простота и локальность выполнения семантически целостных операций над информацией, хранимой в памяти кибернетической системы*
- *корректность/некорректность информации, хранимой в памяти кибернетической системы*
- *однозначность/неоднозначность информации, хранимой в памяти кибернетической системы*
- *целостность/нецелостность информации, хранимой в памяти кибернетической системы*
- *чистота/загрязненность информации, хранимой в памяти кибернетической системы*
- *достоверность/недостоверность информации, хранимой в памяти кибернетической системы*
- *точность/неточность информации, хранимой в памяти кибернетической системы*
- *четкость/нечеткость информации, хранимой в памяти кибернетической системы*
- *определенность/недоопределенность информации, хранимой в памяти кибернетической системы*
- *семантическая мощность языка представления информации в памяти кибернетической системы*
- *объем информации, загруженной в память кибернетической системы*
- *гибридность информации, хранимой в памяти кибернетической системы*
- *стратифицированность информации, хранимой в памяти кибернетической системы*

корректность/некорректность информации, хранимой в памяти кибернетической системы определяется ее уровнем адекватности в той среде, в которой существует *кибернетическая система* и информационной моделью, которой эта хранимая информация является. ***непротиворечивость/противоречивость информации, хранимой в памяти кибернетической системы*** означает уровень присутствия в хранимой информации различного вида противоречий и, в частности, ошибок. Ошибки в хранимой информации могут быть синтаксическими и семантическими, противоречащими некоторым правилам, которые явно в памяти могут быть не представлены и считаются априори истинными.

полнота/неполнота информации, хранимой в памяти кибернетической системы — уровень того, насколько *информация, хранимая в памяти кибернетической системы*, описывает среду существования этой системы и используемые ею *методы решения задач* достаточно полно для того, чтобы кибернетическая система могла действительно решать все множество соответствующих ей задач. Чем полнее *информация, хранимая в памяти кибернетической системы*, и чем полнее информационное обеспечение деятельности этой системы, тем эффективнее (качественнее) сама эта деятельность. Полнота определяется структурированностью информации и многообразием видов знаний, хранимых в *памяти кибернетической системы*.

однозначность/неоднозначность информации, хранимой в памяти кибернетической системы определяется многообразием форм дублирования информации и частотой дублирования информации.

целостность/нецелостность информации, хранимой в памяти кибернетической системы — уровень содержательной информативности информации, то есть уровень того, насколько семантически связной является информация, насколько полно специфицированы все описываемые в памяти сущности (путем описания необходимого набора связей этих сущностей с другими описываемыми сущностями), насколько редко или часто в

рамках хранимой информации встречаются ***информационные дыры***, соответствующие явной недостаточности некоторых спецификаций. Примерами *информационных дыр* являются:
- отсутствующий *метод решения* часто встречающихся *задач*;
- отсутствующее определение используемого определяемого *понятия*;
- недостаточно подробная спецификация часто рассматриваемых сущностей.

чистота/загрязненность информации, хранимой в памяти кибернетической системы означает многообразие форм и общее количество информационного мусора, входящего в состав *информации, хранимой в памяти кибернетической системы*. Под ***информационным мусором*** понимается информационный фрагмент, входящий в состав информации, удаление которого существенно не усложнит деятельность *кибернетической системы*. Примерами *информационного мусора* являются:
- информация, которая нечасто востребована, но при необходимости может быть легко логически выведена;
- информация, актуальность которой истекла.

семантическая мощность языка представления информации в памяти кибернетической системы определяется гибридностью информации, хранимой в памяти кибернетической системы. Язык, *информационные конструкции* которого могут представить любую конфигурацию любых связей между любыми сущностями, является *универсальным языком*. Универсальность *внутреннего языка кибернетической системы* является важнейшим фактором ее интеллектуальности.

гибридность информации, хранимой в памяти кибернетической системы определяется многообразием видов знаний и степенью конвергенции и интеграции различного вида знаний.

стратифицированность информации, хранимой в памяти кибернетической системы есть способность *кибернетической системы* выделять такие разделы *информации, хранимой в памяти этой системы*, которые бы ограничивали области действия *агентов решателя задач кибернетической системы*, являющиеся достаточными для решения заданных задач. Стратифицированность определяется структурированностью и рефлексивностью *информации, хранимой в памяти кибернетической системы*. Рефлексивность *информации, хранимой в памяти кибернетической системы*, то есть наличие метаязыковых средств, является фактором, обеспечивающим не только структуризацию хранимой информации, но и возможность описания *синтаксиса* и *семантики* самых различных *языков*, используемых кибернетической системой.

база знаний является примером *информации, хранимой в памяти кибернетической системы* и имеющей высокий уровень качества по всем показателям и, в частности, высокий уровень:
- *семантической мощности языка представления информации хранимой в памяти кибернетической системы*;
- *гибридности информации, хранимой в памяти кибернетической системы*;
- *многообразия видов знаний, хранимых в памяти кибернетической системы*;
- *формализованности информации, хранимой в памяти кибернетической системы*;
- *структурированности информации, хранимой в памяти кибернетической системы*;

Переход *информации, хранимой в памяти кибернетической системы на уровень качества*, соответствующий *базам знаний*, является важнейшим этапом эволюции *кибернетических систем* (см. *Главу 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*).

Пункт 1.1.1.6. Комплекс свойств, определяющих качество решателя задач кибернетической системы

качество решателя задач кибернетической системы — интегральная качественная оценка множества задач, которые *кибернетическая система* способна выполнять в заданный момент. Основным свойством и назначением *решателя задач кибернетической системы* является способность решать задачи на основе накапливаемых, приобретаемых *кибернетической системой* различного вида навыков с использованием *процессора кибернетической системы*, являющегося универсальным интерпретатором всевозможных накопленных навыков. При этом качество указанной способности определяется целым рядом дополнительных факторов.

общая характеристика решателя задач кибернетической системы
⇒ *свойство-предпосылка**:
{• *общий объем задач, решаемых кибернетической системой*
• *многообразие видов задач, решаемых кибернетической системой*
• *способность кибернетической системы к анализу решаемых задач*
• *способность кибернетической системы к решению задач, методы решения которых в текущий момент известны*

- *способность кибернетической системы к решению задач, методы решения которых ей в текущий момент не известны*
- *множество навыков, используемых кибернетической системой*
- *степень конвергенции и интеграции различного вида моделей решения задач, используемых кибернетической системой*
- *качество организации взаимодействия процессов решения задач в кибернетической системе*
- *быстродействие решателя задач кибернетической системы*
- *способность кибернетической системы решать задачи, предполагающие использование информации, обладающей различного рода не-факторами*
- *многообразие и качество решения задач информационного поиска*
- *способность кибернетической системы генерировать ответы на вопросы*
- *способность кибернетической системы к рассуждениям различного вида*
- *самостоятельность целеполагания кибернетической системы*
- *качество реализации планов собственных действий*
- *способность кибернетической системы к локализации такой области информации, хранимой в ее памяти, которой достаточно для обеспечения решения заданной задачи*
- *способность кибернетической системы к выявлению существенного в информации, хранимой в ее памяти*
- *активность кибернетической системы*

}

общий объем задач, решаемых кибернетической системой, определяется *мощностью языка представления задач, решаемых кибернетической системой*. Мощность языка представления задач прежде всего определяется ***многообразием видов задач, решаемых кибернетической системой*** (многообразием видов описываемых действий). Каждая задача есть спецификация соответствующего (описываемого) действия. Поэтому рассмотрение *многообразия видов задач, решаемых кибернетической системой*, полностью соответствует многообразию видов деятельности, осуществляемой этой системой. Важно заметить, что есть виды *деятельности кибернетической системы*, которые определяют качество и, в частности, *уровень интеллекта кибернетической системы*.

способность кибернетической системы к анализу решаемых задач предполагает оценку задачи на предмет:
- сложности достижения;
- целесообразности достижения (нужности, важности, приоритетности);
- соответствия цели существующим нормам (правилам) соответствующей деятельности.

метод решения задач — это вид *знаний, хранимых в памяти кибернетической системы* и содержащих информацию, которой достаточно либо для сведения каждой задачи из соответствующего класса задач к полной системе подзадач, решение которых гарантирует решение исходной задачи, либо для окончательного решения этой задачи из указанного класса задач. *методами для решения задач* могут быть не только алгоритмы, но также и *функциональные программы*, *продукционные системы*, *логические исчисления*, *генетические алгоритмы*, *искусственные нейронные сети* различного вида и так далее. Задачи, для которых не находятся соответствующие им методы, решаются с помощью метаметодов (стратегий) решения задач, направленных:
- на генерацию нужных исходных данных (нужного контекста), необходимых для решения каждой задачи;
- на генерацию плана решения задачи, описывающего сведение исходной задачи к подзадачам (до тех подзадач, методы решения которых системы известны);
- на сужение области решения задачи (на сужения контекста задачи, достаточного для ее решения).

качество решения каждой задачи определяется:
- *временем ее решения* (чем быстрее задача решается, тем выше качество ее решения);
- *полнотой и корректностью результата решения задачи*;
- *затраченными для решения задачи ресурсами памяти* (объемом фрагмента хранимой информации, используемой для решения задачи);
- *затраченным для решения задачи ресурсами решателя задач* (количеством используемых внутренних агентов).

Таким образом, повышение *качества решения каждой конкретной задачи*, а также каждого *класса задач* (путем совершенствования соответствующего метода, в частности, алгоритма) является важным фактором повышения *качества решателя задач кибернетической системы* в целом.

Перспективным вариантом построения *решателя задач кибернетической системы* является реализация *агентно-ориентированной модели обработки информации*, то есть построение *решателя задач* в виде *многоагентной системы*, агенты которой осуществляют обработку *информации, хранимой в памяти кибернетической системы*, и управляются этой информацией (точнее, ее текущим состоянием). Особое место среди этих агентов занимают *сенсорные* (рецепторные) и *эффекторные агенты*, которые, соответственно, воспринимают информацию о текущем состоянии внешней среды и воздействуют на внешнюю среду, в частности, путем изменения состояния

физической оболочки кибернетической системы (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*).

Указанная агентно-ориентированная модель организации взаимодействия процессов решения задач в *кибернетической системе* по сути есть не что иное, как модель ситуационного управления процессами решения задач, решаемых *кибернетической системой* как в своей внешней среде, так и в своей памяти.

быстродействие решателя задач кибернетической системы сводится к скорости решения задач, быстродействию решателя задач, скорости реакции *кибернетической системы* на различные задачные ситуации. Во многом свойство определяется *быстродействием процессора кибернетической системы*.

способность кибернетической системы решать задачи, предполагающие использование информации, обладающей различного рода не-факторами включает:
- нечетко сформулированные задачи ("делай то, не знаю что");
- задачи, которые решаются в условиях неполноты, неточности, противоречивости исходных данных;
- задачи, принадлежащие классам задач, для которых практически невозможно построить соответствующие алгоритмы.

Примерами задач, предполагающих использование информации, обладающей различного рода *не-факторами*, являются задачи проектирования, распознавания, целеполагания, прогнозирования, и так далее. Для таких задач характерны:
- неточность и недостоверность исходных данных;
- отсутствие критерия качества результата;
- невозможность или высокая трудоемкость разработки алгоритма;
- необходимость учета контекста задачи.

способность кибернетической системы генерировать (порождать, строить, синтезировать, выводить) ответы на самые различные вопросы и, в частности, на *вопросы* типа “что это такое?”, на почему-вопросы, означает способность *кибернетической системы* объяснять (обосновывать корректность) своих действий (см. *Главу 3.4. Язык вопросов для ostis-систем*).

самостоятельность целеполагания кибернетической системы — способность *кибернетической системы* генерировать, инициировать и решать задачи, которые не являются подзадачами, инициированными внешними (другими) субъектами, а также способность на основе анализа своих возможностей отказаться от выполнения задачи, инициированной извне, переадресовав ее другой *кибернетической системе*, либо на основе анализа самой этой задачи обосновать ее нецелесообразность или некорректность. Повышение уровня самостоятельности существенно расширяет возможности *кибернетической системы*, то есть объем тех задач, которые она может решать не только в "идеальных" условиях, но и в реальных, осложненных обстоятельствах. Способность *кибернетической системы* адекватно расставлять приоритеты своим целям и не "распыляться" на достижение неприоритетных (несущественных) целей есть способность анализа целесообразности деятельности.

способность кибернетической системы к выявлению существенного в информации, хранимой в ее памяти — способность к выявлению (обнаружению, выделению) таких фрагментов *информации, хранимой в памяти кибернетической системы*, которые существенны (важны) для достижения соответствующих целей. Понятие *существенного (важного) фрагмента информации, хранимой в памяти кибернетической системы*, относительно и определяется соответствующей задачей. Тем не менее, есть важные перманентно решаемые задачи, в частности *задачи анализа качества информации, хранимой в памяти кибернетической системы*. Существенные фрагменты хранимой информации, выделяемые в процессе решения этих задач, являются относительными не столько по отношению к решаемой задаче, сколько по отношению к текущему состоянию хранимой информации.

Уровень ***активности кибернетической системы*** может быть разным для разных решаемых задач, для разных классов выполняемых действий, для разных *видов деятельности*. Чем выше *активность кибернетической системы*, тем (при прочих равных условиях) она больше успевает сделать, следовательно, тем выше ее качество (эффективность). Обратным свойством является понятие ***пассивности кибернетической системы***.

Пункт 1.1.1.7. Комплекс свойств, определяющих уровень обучаемости кибернетической системы

обучаемость кибернетической системы — способность *кибернетической системы* повышать свое качество, адаптируясь к решению новых задач, *качество внутренней информации модели своей среды*, *качество* своего *решателя задач* и даже *качество* своей *физической оболочки*.

Максимальный уровень ***обучаемости кибернетической системы*** — это ее способность эволюционировать (повышать уровень своего качества) максимально быстро и в любом направлении, то есть способность быстро и без каких-либо ограничений приобретать любые новые *знания* и *навыки*.

Реализация способности *кибернетической системы* обучаться, то есть решать перманентно инициированную сверхзадачу самообучения, накладывает дополнительные требования, предъявляемые к *информации, хранимой в памяти кибернетической системы*, к *решателю задач кибернетической системы*, а в перспективе также и к *физической оболочке кибернетической системы*.

Важнейшей характеристикой *кибернетической системы* является не только то, какой уровень интеллекта *кибернетическая система* имеет в текущий момент, какое множество действий (задач) она способна выполнять, но и то, насколько быстро этот уровень может повышаться.

обучаемость кибернетической системы
$\Rightarrow$ *свойство-предпосылка**:
- *гибкость кибернетической системы*
- *стратифицированность кибернетической системы*
- *рефлексивность кибернетической системы*
- *ограниченность обучения кибернетической системы*
- *познавательная активность кибернетической системы*
- *способность кибернетической системы к самосохранению*

Поскольку обучение всегда сводится к внесению тех или иных изменений в обучаемую *кибернетическую систему*, без высокого уровня гибкости этой системы не может быть высокого уровня ее обучаемости. ***гибкость кибернетической системы*** определяется простотой и многообразием возможных самоизменений *кибернетической системы*.

При наличии ***стратифицированности кибернетической системы*** появляется возможность четкого определения области действия различных изменений, вносимых в кибернетическую систему, то есть возможность четкого ограничения тех частей кибернетической системы, за пределы которых нет необходимости выходить для учета последствий внесенных в систему первичных изменений (осуществлять дополнительные изменения, являющиеся последствиями первичных изменений).

рефлексивность кибернетической системы есть способность *кибернетической системы* к самоанализу. Конструктивным результатом рефлексии *кибернетической системы* является генерация в ее памяти спецификации различных негативных или подозрительных особенностей, которые следует учитывать для повышения качества кибернетической системы. Такими особенностями (недостатками) могут быть выявленные противоречия (ошибки), выявленные пары *синонимичных знаков*, *омонимичные знаки*, *информационные дыры*.

ограниченность обучения кибернетической системы определяет границу между теми *знаниями* и *навыками*, которые соответствующая *кибернетическая система* принципиально может приобрести, и теми *знаниями* и *навыками*, которые указанная *кибернетическая система* не сможет приобрести никогда. Данное свойство определяет максимальный уровень потенциальных возможностей соответствующей кибернетической системы. Максимальная степень отсутствия ограничений в приобретении новых *знаний* и *навыков* — это полное отсутствие ограничений, то есть полная универсальность возможностей соответствующих кибернетических систем.

познавательная активность кибернетической системы — любознательность, активность и самостоятельность в приобретении новых знаний и навыков. Следует отличать способность приобретать новые *знания* и *навыки*, а также их совершенствовать, от желания это делать. Желание (целевая установка) научиться решать те или иные задачи может быть сформулировано *кибернетической системой* либо самостоятельно, либо извне (некоторым учителем).

познавательная активность кибернетической системы
$\Rightarrow$ *свойство-предпосылка**:
- *способность кибернетической системы к синтезу познавательных целей и процедур*
- *способность кибернетической системы к самоорганизации собственного обучения*
- *способность кибернетической системы к экспериментальным действиям*

способность кибернетической системы к синтезу познавательных целей и процедур является способностью планировать свое обучение и управлять процессом обучения, умение задавать *вопросы* или целенаправленные последовательности *вопросов*, способность генерировать четкую спецификацию своей информационной потребности. ***способность кибернетической системы к самоорганизации собственного обучения*** есть способность осуществлять управление своим обучением, способность кибернетической системы самой выполнять роль своего учителя. ***способность кибернетической системы к экспериментальным действиям*** — способность к отклонениям от составленных планов своих действий для повышения качества результата или сохранении целенаправленности этих действий, способность к импровизации.

Чем выше уровень *безопасности кибернетической системы*, тем выше уровень *обучаемости кибернетической системы*. ***способность кибернетической системы к самосохранению*** означает способность *кибернетической*

системы к выявлению и устранению угроз, направленных на снижение ее качества и даже на ее уничтожение, что означает полную потерю необходимого качества (см. *Главу 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*).

Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы

Исследователи *Искусственного интеллекта* определяют *интеллект* как неотъемлемое свойство машины (см. *Gao R..PerfoMfISaE-2002art*). Их целью является построение систем, которые демонстрируют весь спектр когнитивных способностей, которые мы обнаруживаем у людей (см. *Laird J.E..Claim aCiEHLIS-2009art*).

Основным свойством, характеристикой *кибернетической системы* является уровень ее интеллекта, который является интегральной характеристикой, определяющей уровень эффективности взаимодействия *кибернетической системы* со средой своего существования. Процесс эволюции *кибернетических систем* следует рассматривать как процесс повышения уровня их качества по целому ряду свойств и как процесс повышения уровня их интеллекта (см. *Zagorskiy A..Facto tDtLoIoCS-2022art*).

кибернетическая система может быть как *интеллектуальной*, так и *неинтеллектуальной*. В свою очередь, *интеллектуальные система* может быть как *слабоинтеллектуальной*, так и *высокоинтеллектуальной*.

уровень интеллекта кибернетической системы
⇒ *свойство-предпосылка**:
- *образованность кибернетической системы*
- *обучаемость кибернетической системы*
- *интероперабельность кибернетической системы*

образованность кибернетической системы — уровень *навыков*, а также иных *знаний*, приобретенных *кибернетической системой* к заданному моменту.

образованность кибернетической системы
⇒ *свойство-предпосылка**:
- *качество навыков, приобретенных кибернетической системой*
- *качество информации, хранимой в памяти кибернетической системы*

Примерами *образованной кибернетической системы* являются:
- *кибернетическая система, основанная на знаниях*;
- *кибернетическая система, управляемая знаниями*;
- *целенаправленная кибернетическая система*;
- *гибридная кибернетическая система*;
- *потенциально универсальная кибернетическая система*.

обучаемая кибернетическая система — *кибернетическая система*, способная познавать среду своего обитания, то есть строить и постоянно уточнять в своей памяти информационную модель этой среды, а также использовать эту модель для решения различных задач (для организации своей деятельности) в указанной среде.

Примерами *обучаемой кибернетической системы* являются:
- *кибернетическая система* с высоким уровнем стратифицированности своих знаний и навыков;
- *рефлексивная кибернетическая система*;
- *самообучаемая кибернетическая система*;
- *кибернетическая система* с высоким уровнем познавательной активности.

интеллект кибернетической системы, как и лежащий в его основе познавательный процесс, выполняемый кибернетической системой, имеет социальный характер, поскольку наиболее эффективно формируется и развивается в форме взаимодействия кибернетической системы с другими кибернетическими системами. **социально ориентированная *кибернетическая система*** имеет достаточно высокий уровень интеллекта, чтобы быть полезным членом различных, в том числе и человеко-машинных сообществ. Определенный уровень социально значимых качеств является необходимым условием *интеллектуальности кибернетической системы*. Примерами *социально ориентированной кибернетической системы* являются:
- *кибернетическая система*, способная устанавливать и поддерживать высокий уровень семантической совместимости и взаимопонимания с другими системами;
- *договороспособная кибернетическая система*.

Все свойства, присущие *кибернетическим системам*, в различных *кибернетических системах* могут иметь самый различный уровень. Более того, в некоторых *кибернетических системах* некоторые из этих свойств могут вообще

отсутствовать. При этом в *кибернетических системах*, которые условно будем называть *интеллектуальными системами*, все указанные выше свойства должны быть представлены в достаточно развитом виде.

§ 1.1.2. Понятие интеллектуальной многоагентной системы

⇒ *ключевое понятие**:
- *индивидуальная кибернетическая система*
- *многоагентная система*
  ⊃ *кибернетическая система*
- *иерархическая кибернетическая система*
- *агент*
- *многоагентная система*
  ≔ [самостоятельный компонент многоагентной системы]
  ≔ [кибернетическая система, являющаяся агентом по крайней мере одной многоагентной системы]
- *агент**
  ≔ [быть агентом заданной многоагентной системы*]
  ⇒ *второй домен**:
  агент
- *индивидуальный агент*
- *коллективный агент*
- *встроенный агент*
  ≔ [внутренний агент индивидуальной кибернетической системы]
- *специализированный агент*
  ≔ [интеллектуальный агент]
- *мультиагент*
  ≔ [кибернетическая система, являющаяся агентом более чем одной многоагентной системы]

⇒ *ключевое знание**:
- *Классификация многоагентных систем*
- *Обобщенная архитектура кибернетических систем*
- *Факторы, определяющие качество многоагентных систем*
- *Факторы, определяющие уровень интеллекта многоагентных систем*

⇒ *рассматриваемый вопрос**:
- *Почему переход от отдельных кибернетических систем к их коллективным, многоагентным системам, агентами которых являются заданные кибернетические системы, является одним из направлений повышения уровня интеллекта исходных кибернетических систем.*
- *Всегда ли объединение кибернетических систем в коллектив этих систем приводит у повышению уровня интеллекта.*
- *Почему не каждое соединение достаточно качественных кибернетических систем порождает качественную многоагентную систему.*

⇒ *библиографическая ссылка**:
- *Тарасов В.Б.оМногоСкИОФ-2002кн*
- *Варшавский В.А..ОркесИБДР-1984кн*
- *Hadzic M..OntolBMAS-2009bk*
- *Dorri A..MultiASaS-2018art*
- *Ferber J..fAgent tOaOVoM-2003art*
- *Balaji P.G..aIntro tMAS-2010art*

Переход от *кибернетических систем* к *коллективам* взаимодействующих между собой *кибернетических систем*, то есть к социальной организации *кибернетических систем*, является важнейшим фактором эволюции *кибернетических систем*. *кибернетическую систему*, представляющую собой *коллектив* взаимодействующих *кибернетических систем*, обладающих определенной степенью самостоятельности (самодостаточности, свободы выбора), будем называть ***многоагентной системой*** (см. *Dorri A..MultiASaS-2018art*).

кибернетическая система
⇒ *разбиение**:
{• *индивидуальная кибернетическая система*

- *кибернетическая система, являющаяся минимальным компонентом индивидуальной кибернетической системы*
- *кибернетическая система, являющаяся комплексом компонентов соответствующей индивидуальной кибернетической системы*
- *сообщество индивидуальных кибернетических систем*
  ⇒ *разбиение**:
  {• *простое сообщество индивидуальных кибернетических систем*
  • *иерархическое сообщество индивидуальных кибернетических систем*
  }

}

Теория многоагентных систем
⇒ *пояснение**:
[Теория многоагентных систем — это теория перехода количества *кибернетических систем* в *кибернетическую систему* нового качества, — это выявление принципов и методов, позволяющие множество *кибернетических систем соединить* в *коллективную кибернетическую систему*, обладающую существенно более высоким уровнем качества (в том числе интеллекта) по сравнению с *качеством кибернетической системы*, входящих в этот коллектив.]

Агенты *многоагентной системы* могут (но вовсе не обязательно должны) быть *интеллектуальными системами*. Так, например, агенты *интеллектуального решателя задач*, имеющего агентно-ориентированную архитектуру, не являются *интеллектуальными системами*. Агентом *иерархической многоагентной системы* может быть другая *многоагентная система* (см. *Ferber J..fAgent tOaOVoM-2003art*).

многоагентная система — коллектив взаимодействующих автономных *кибернетических систем*, имеющих общую среду обитания, жизнедеятельности (см. *Hadzic M..OntolBMAS-2009bk*).

Классификация *многоагентных систем* приведена ниже:

многоагентная система
⇒ *разбиение**:
{• *одноуровневая многоагентная система*
• *иерархическая многоагентная система*
}
⇒ *разбиение**:
{• *многоагентная система с централизованным управлением*
• *многоагентная система с децентрализованным управлением*
}

одноуровневая многоагентная система реализует либо одну модель параллельного (распределенного) решения задач соответствующего класса, либо комбинацию фиксированного числа разных и параллельно реализованных *моделей решения задач*. ***иерархическая многоагентная система*** состоит из агентов, которые могут быть *индивидуальными кибернетическими системами*, *коллективами индивидуальных кибернетических систем*, а также *коллективами, состоящими из индивидуальных кибернетических систем и коллективов индивидуальных кибернетических систем*.

В ***многоагентной системе с централизованным управлением*** специально выделяются агенты, которые принимают решения в определенной области деятельности *многоагентной системы* и обеспечивают выполнение этих решений путем управления деятельностью остальных агентов, входящих в состав этой системы.

В ***многоагентной системе с децентрализованным управлением*** решения принимаются коллегиально и "автоматически" (решения о признании новой кем-то предложенной информации — в том числе, об инициировании некоторой задачи, решения о коррекции (уточнении) уже ранее признанной, согласованной информации) на основе четко продуманной и постоянно совершенствуемой методики, а также на основе активного участия всех агентов в формировании новых предложений, подлежащих признанию или согласованию (см. *Balaji P.G..aIntro tMAS-2010art*). В такой *многоагентной системе* все агенты участвуют в управлении этой системы. Примером такой системы является оркестр, способный играть без дирижера.

Переход к *многоагентным системам* является важнейшим фактором повышения *качества* (и, в частности, уровня интеллекта) *кибернетических систем*, так как уровень *интеллекта многоагентной системы* может быть значительно выше уровня интеллекта каждого входящего в нее агента. Это происходит далеко не всегда, поскольку важнейшим фактором *качества многоагентных систем* является не только качество входящих в нее агентов, но и организация взаимодействия агентов и, в частности, переход от централизованного к децентрализованному управлению. Количество не всегда переходит в новое качество.

качество индивидуальных кибернетических систем определяется, кроме всего прочего, тем, насколько большой вклад *индивидуальная кибернетическая система* вносит в повышение качества тех коллективов, в состав которых она входит. Указанное свойство *индивидуальных кибернетических систем* будем называть уровнем их *интероперабельности* (см. *Ouksel A.M..SemanIiGIS-1999art*, *Главу 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*).

синергетическая кибернетическая система — *многоагентная система*, обладающая высоким уровнем коллективного интеллекта, атомарными агентами которой являются *индивидуальные интеллектуальные системы*, имеющие высокий уровень *интероперабельности* (см. *Lopes L.S..SemanIitIoT-2022art*, *Hamilton Inter..Intero aKEftG-2006art*). Примером *синергетической кибернетической системы* является *творческий коллектив*, реализующий сложный наукоемкий проект.

Эффективность *творческого коллектива* (например в области *научно-технической деятельности*) определяется:

- согласованностью мотивации, целевой установки всего коллектива и каждого его члена (не должно быть противоречий между целью коллектива и творческой самореализацией каждого его члена);
- эффективной организацией децентрализованного управления деятельностью членов сообщества;
- четкой, оперативной и доступной всем фиксацией документации текущего состояния содеянного и направлений его дальнейшего развития;
- уровнем трудоемкости оперативности фиксации индивидуальных результатов в рамках коллективно создаваемого общего результата;
- уровнем структурированности и, прежде всего, стратифицированности обобщенной документации (базы знаний);
- эффективностью ассоциативного доступа к фрагментам документации;
- гибкостью коллективно создаваемой базы;
- автоматизацией анализа содеянного и управления проектом.

Уровень *интеллекта многоагентной системы* может быть значительно ниже уровня интеллекта самого "глупого" члена этого коллектива, но может быть и значительно выше уровня интеллекта самого "умного" члена указанного коллектива. Для того, чтобы количество *интеллектуальных систем* переходило в существенно более интеллектуальное качество коллектива таких систем, все объединяемые в коллектив *интеллектуальные системы* должны иметь высокий уровень *интероперабельности*, что накладывает дополнительные требования, предъявляемые к *информации, хранимой в памяти интеллектуальных систем*, а также к *решателям задач интеллектуальных систем*, объединяемых в коллектив.

интероперабельность кибернетической системы — способность *кибернетической системы* взаимодействовать с другими кибернетическими системами в целях создания коллектива *кибернетических систем* (многоагентных систем), уровень качества и, в частности, уровень *интеллекта* которого выше уровня *качества* каждой *кибернетической системы*, входящей в состав этого коллектива.

Для того, чтобы количество членов *коллектива кибернетической системы* перешло в более высокое качество самого коллектива, члены коллектива должны обладать дополнительными способностями, которые будем называть свойствами *интероперабельности*. Основными такими свойствами являются способность устанавливать и поддерживать достаточный уровень *семантической совместимости* (взаимопонимания) с другими *кибернетическими системами* и *договороспособности* между ними (способность согласовывать свои действия с другими) (см. *Neiva F.W..TowarPItSC-2016art*).

Целенаправленный обмен информацией между *кибернетическими системами* существенно ускоряет процесс их обучения (процесс накопления знаний и навыков). Следовательно, способность эффективно использовать указанный канал накопления знаний и навыков существенно повышает уровень *обучаемости кибернетических систем*. Повышение уровня *интероперабельности кибернетической системы* является, с одной стороны, дополнительным повышением уровня *интеллекта* самой этой *кибернетической системы*, а также фактором повышения уровня *интеллекта* тех *коллективов*, тех *многоагентных систем*, в состав которых эта *кибернетическая система* входит.

интероперабельность кибернетической системы
⇒ *свойство-предпосылка**:
- *договороспособность кибернетической системы*
- *социальная ответственность кибернетической системы*
- *социальная активность кибернетической системы*

Свойства-предпосылки уровня *договороспособности кибернетической системы* представлены ниже:

договороспособность кибернетической системы
⇒ *свойство-предпосылка**:
- *способность кибернетической системы к пониманию принимаемых сообщений*

- *способность кибернетической системы к формированию передаваемых сообщений, понятных адресатам*
- *способность кибернетической системы к обеспечению семантической совместимости с партнерами*
- *коммуникабельность кибернетической системы*
- *способность кибернетической системы к обсуждению и согласованию целей и планов коллективной деятельности*
- *способность кибернетической системы брать на себя выполнение актуальных задач в рамках согласованных планов коллективной деятельности*

Понимание информации, поступающей извне, включает в себя:
- перевод этой информации на внутренний язык кибернетической системы;
- локальную верификацию вводимой информации;
- погружение (конвергенцию, размещение) текста, являющегося результатом указанного перевода в состав хранимой информации (в частности, в состав базы знаний).

Погружение вводимой информации в состав *базы знаний кибернетической системы* сводится к выявлению и устранению противоречий, возникающих между погружаемым текстом и текущего состояния *базы знаний*. Сложность проблемы понимания вводимой вербальной информации заключается не только в сложности непротиворечивого погружения вводимой информации в текущее состояние *базы знаний*, но и в сложности трансляции этой информации с *внешнего языка* на *внутренний язык кибернетической системы*, то есть в сложности генерации текста внутреннего языка, семантически эквивалентного вводимому тексту *внешнего языка*. Для *естественных языков* указанная трансляция является сложной задачей, так как в настоящее время проблема формализации *синтаксиса* и *семантики естественных языков* не решена.

семантическая совместимость двух заданных кибернетических систем определяется *согласованностью систем понятий*, используемых обеими взаимодействующими *кибернетическими системами*. Проблема обеспечения перманентной поддержки *семантической совместимости* взаимодействующих *кибернетических систем* является необходимым условием обеспечения высокого уровня *взаимопонимания кибернетических систем* и, как следствие, эффективного их взаимодействия.

коммуникабельность кибернетической системы — способность *кибернетической системы* к установлению взаимовыгодных контактов с другими *кибернетическими системами* (в том числе, с коллективами интеллектуальных систем) путем честного выявления взаимовыгодных общих целей (интересов).

Свойства-предпосылки уровня *социальной ответственности кибернетической системы* представлены ниже:

социальная ответственность кибернетической системы
⇒ *свойство-предпосылка**:
- *способность кибернетической системы выполнять качественно и в срок взятые на себя обязательства в рамках соответствующих коллективов*
- *способность кибернетической системы адекватно оценивать свои возможности при распределении коллективной деятельности*
- *альтруизм/эгоизм кибернетической системы*
- *отсутствие/наличие действий, которые по безграмотности кибернетической системы снижают качество коллективов, в состав которых она входит*
- *отсутствие/наличие "осознанных", мотивированных действий, снижающих качество коллективов, в состав которых кибернетическая система входит*

Свойства-предпосылки уровня *социальной активности кибернетической системы* представлены ниже:

социальная активность кибернетической системы
⇒ *свойство-предпосылка**:
- *способность кибернетической системы к генерации предлагаемых целей и планов коллективной деятельности*
- *активность кибернетической системы в экспертизе результатов других участников коллективной деятельности*
- *способность кибернетической системы к анализу качества всех коллективов, в состав которых она входит, а также всех членов этих коллективов*
- *способность кибернетической системы к участию в формировании новых коллективов*
- *количество и качество тех коллективов, в состав которых кибернетическая система входит или входила*

Формирование специализированного *коллектива кибернетических систем* сводится к тому, что в *памяти* каждой *кибернетической системы*, входящей в коллектив, генерируется спецификация этого коллектива, включающая в себя:

- перечень весь членов коллектива;
- способности каждого из членов коллектива;
- их обязанности в рамках коллектива;
- спецификацию всего множества задач (вида деятельности), для решения (выполнения) которых сформирован данный коллектив кибернетических систем.

Каждая *кибернетическая система* может входить в состав большого количества коллективов, выполняя при этом в разных коллективах в общем случае разные "должностные обязанности", разные "бизнес-процессы".

§ 1.1.3. Направления эволюции компьютерных систем

⇒ *эпиграф**:
[From data science to knowledge science.]

В эволюции компьютерных систем можно выделить два общих направления.

Первое направление — это

- **расширение множества и многообразия задач**, решаемых компьютерной системой;
- повышение **сложности этих задач** вплоть до трудно формализуемых (трудно решаемых) задач, интеллектуальных задач, решаемых в условиях неполноты, неточности, нечеткости и так далее;
- повышение **качества решения задач** либо путем более эффективного использования известных моделей решения задач (например, путем разработки более качественных алгоритмов), либо путем использования принципиально новых моделей решения задач;
- расширение **многообразия используемых видов информации** (знаний);
- расширение **многообразия используемых моделей решения задач**.

Очевидно, что расширение множества решаемых задач в условиях пусть и большой, но всегда конечной памяти компьютерной системы делает все более и более актуальным переход от частных методов и моделей решения задач к их обобщениям (или, как отмечал Д. А. Поспелов, от связки "ключей" к набору "отмычек").

Очевидно также, что многообразие видов задач, решаемых компьютерными системами, многообразие используемых моделей решения задач приводит:

- к интегрированным информационным ресурсам;
- к интегрированным решателям задач;
- к интегрированным компьютерным системам;
- к коллективам компьютерных систем.

Проблема здесь заключается не в самой интеграции, а в ее качестве. Интеграция может быть **эклектичной**, если не обеспечить совместимость интегрируемых компонентов, а в случае такой совместимости интеграция может привести к новому качеству, к дополнительному расширению множества решаемых задач. Это будет означать переход от эклектичности к гибридности, синергетичности.

Второе общее направление эволюции компьютерных систем — это повышение уровня их ***обучаемости*** и, как следствие, темпов их эволюции.

обучаемость компьютерной системы определяется:

- **трудоемкостью** и темпами приобретения (расширения) и совершенствования активно используемых знаний и навыков;
- **уровнем ограничений**, накладываемых на вид приобретаемых и используемых знаний и навыков (фактически, это ограничения на множество всех тех задач, которые принципиально могут быть решены данной компьютерной системой).

В свою очередь, **трудоемкость и темпы расширения и совершенствования** знаний и навыков компьютерной системы определяется:

- ***гибкостью*** — многообразием и трудоемкостью возможных изменений, вносимых в систему в процессе пополнения системы новыми знаниями и навыками и совершенствования уже приобретенных знаний и навыков;
- ***стратифицированностью*** — четким разделением системы на достаточно независящие друг от друга уровни иерархии, то есть возможностью локализации фрагментов компьютерной системы, не выходя за пределы которых, априори достаточно проводить анализ последствий тех или иных вносимых в систему изменений;
- ***рефлексивностью*** — способностью анализировать собственное состояние и свою деятельность;
- ***гибридностью*** — способностью приобретать и использовать широкое (а в идеале — неограниченное) многообразие знаний и навыков;

- ***уровнем самообучаемости*** — уровнем активности, самостоятельности, целеустремленности в процессе своего обучения, то есть уровнем способности к обучению без учителя, уровнем автоматизации приобретения новых знаний и навыков, а также совершенствования уже приобретенных знаний и навыков;
- ***совместимостью*** — трудоемкостью интеграции;
- ***способностью к постоянному мониторингу и поддержке своей совместимости*** как с другими компьютерными системами, так и со своими пользователями в условиях интенсивной эволюции этих компьютерных систем и их пользователей.

совместимость (трудоемкость интеграции) компьютерных систем может рассматриваться в двух аспектах:
- в аспекте **глубокой интеграции** компьютерных систем, что предполагает преобразование нескольких компьютерных систем в одну целостную компьютерную систему путем объединения информационных и функциональных ресурсов интегрируемых компьютерных систем;
- в аспекте преобразования нескольких компьютерных систем в **коллектив взаимодействующих компьютерных систем**, способных к совместному корпоративному решению сложных задач.

совместимость (трудоемкость интеграции) компьютерных систем определяется:
- совместимостью различного вида информации (знаний), хранимой в памяти компьютерной системы;
- совместимостью различных моделей решения задач;
- совместимостью встроенных (в том числе типовых) подсистем, входящих в состав компьютерных систем;
- совместимостью внешней информации, поступающей на вход компьютерной системе, с информацией, хранимой в памяти компьютерной системы (трудоемкостью понимания внешней информации — трансляции, погружения, выравнивания понятий);
- коммуникационной (в том числе семантической) совместимостью с пользователями и с другими компьютерными системами.

Важнейшая форма обучения компьютерной системы это приобретение новых знаний и навыков в "готовом" виде, то есть в виде некоторых знаковых конструкций, вводимых в память компьютерной системы, поскольку приобретение знаний и навыков из внешних достоверных источников требует существенно меньшего времени по сравнению с их приобретением собственными силами, на основе собственного опыта и собственных ошибок. Но для того, чтобы указанная форма обучения была эффективной, необходимо максимально возможным образом упростить и формализовать механизм (процедуру) погружения новых знаний в память компьютерной системы.

Для решения этой задачи ключевое значение имеет создание удобного для этой цели способа кодирования различного вида информации в памяти компьютерной системы.

Поскольку основным каналом обучения компьютерных систем является приобретение ими знаний и навыков от других субъектов — от других компьютерных систем и от пользователей (от разработчиков-учителей и от конечных пользователей), важнейшим фактором обучаемости компьютерной системы является превращение компьютерной системы в коммуникативную систему, способную эффективно общаться с внешними субъектами. Следовательно, уровень обучаемости компьютерных систем определяется также уровнем ее совместимости с самими этими внешними субъектами, с приобретаемыми ею знаниями и навыками, то есть степенью того, как компьютерная система вместе с теми субъектами, с которыми она обменивается информацией, решает проблему "вавилонского столпотворения".

Таким образом, этапы эволюции *традиционных компьютерных систем*, в основе которых лежит их интерпретация на *машинах фон Неймана*, направлены на повышение качества этих систем и, в частности, на повышение уровня их *интеллекта*.

В качестве примера рассмотрим эволюцию *языков программирования* компьютерных систем:
- Исходная особенность языков программирования заключается в том, что язык представления обрабатываемых программами данных (его синтаксис и денотационная семантика) не задается и фактически для любой программы или для семейства программ разрабатывается свой такой язык. (Языки программирования "хромают" на одну ногу.).
- Данные преобразуются в *базы данных*, которые становятся общими для программ заданного языка программирования и изменение которых не может быть обусловлено и предусмотрено каждой из этих программ. Такие языки становятся *языками программирования, ориентированными на обработку баз данных*, а базам данных ставится в соответствие общий язык представления баз данных (с соответствующим синтаксисом и денотационной семантикой).
- Разные *языки программирования* (с разной денотационной и операционной семантикой) ориентируются на обработку *баз данных*, которым соответствует один и тот же *язык представления баз данных* (т.е. языки становятся совместимыми по обрабатываемым данными).
- *Языки представления баз данных* становятся универсальными и "превращаются" в универсальные *языки представления баз знаний* (заметим, что продукционные и фреймовые языки представления знаний не являются универсальными).

- Разные *языки программирования*, ориентированные на обработку баз знаний, становятся подъязыками *универсального языка представления баз знаний*, т.е. становятся совместимыми не только по обрабатываемым базам знаний, но также и по своему *синтаксису*.
- Расширяется многообразие *языков программирования*, реализующих различные *модели решателей задач*:
  - алгоритмические языки программирования низкого и высокого уровня;
  - последовательные и параллельные процедурные языки программирования (синхронные и асинхронные);
  - функциональные языки программирования;
  - логические языки программирования;
  - продукционные языки программирования;
  - объектно-ориентированные языки программирования;
  - генетические алгоритмы.
- Создаются *языки семантической спецификации программ*, *языки формулировки задач* и стратегии поиска пути решения задач на основе заданного пакета программ различных языков программирования.

Эволюция языков программирования подробнее рассматривается в работах Ершова А.П., Капитоновой Ю.В., Летичевского А.А., Непейводы Н.Н., Мак-Карти Дж. (язык LISP), Ковальского Р. (язык Prolog) и других.

Глава 1.2.
Интеллектуальные компьютерные системы нового поколения

⇒ *автор**:
- *Голенков В. В.*
- *Шункевич Д. В.*
- *Ковалев М. В.*
- *Садовский М. Е.*

⇒ *аннотация**:
[В главе рассмотрены принципы построения *интеллектуальных компьютерных систем нового поколения*. В качестве ключевых свойств *интеллектуальных систем нового поколения* выделяются их *самообучаемость*, *интероперабельность* и *семантическая совместимость*. В главе рассматривается подход к обеспечению указанных свойств на основе *смыслового представления информации* и *многоагентных моделей обработки информации*.]

⇒ *подраздел**:
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*
- *§ 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения*
- *§ 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения*

⇒ *ключевой знак**:
- *Технология OSTIS*
- *УСК*

⇒ *ключевое понятие**:
- *интеллектуальная компьютерная система нового поколения*
- *интероперабельная интеллектуальная компьютерная система*
- *самообучаемая интеллектуальная компьютерная система*
- *семантическая сеть*
- *многоагентная система обработки информации в общей памяти*

⇒ *библиографическая ссылка**:
- *Yaghoobirafi K..aAppro fSIiADIS-2022art*
- *Ouksel A.M..SemanIiGIS-1999art*
- *Lanzenberger M..MakinOTKIitSW-2008art*
- *Neiva F.W..TowarPItSC-2016art*
- *Pohl J.Inter atNfISaHP-2004art*
- *Waters J..GlobaIUSS-2009art*
- *Lopes L.S..SemanIitIoT-2022art*
- *Hamilton Inter..Intero aKEftG-2006art*
- *Баринов И.И..ФормиСРКпИИ-2021ст*
- *Мартынов В.В.СемиоОИ-1974кн*
- *Мельчук И.А.ОпытТЛМСТ-1999кн*
- *Мельчук И.А.КакНМЛ-1998ст*
- *Харламов А.А..СеманСкФОР-2011ст*
- *Москин Н.Д.оПредсЗсПССвИ-2011ст*
- *Тимченко В.А.МетодПКСС-2013ст*
- *Массель Л.В..СеманТнОИОК-2013ст*
- *Ефименко И.В..УСКМТЛС-2014ст*
- *Голенков В.В.ред.ОткрыСТПИ-2014ст*
- *Голенков В.В..СтрукСП-2014ст*
- *Загорулько Ю.А..оФормаСОЗвИиИСнОО-2014ст*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Golenkov V.V..tStand oICSaaK-2020art*
- *Летичевский А.А..ИнсерП-2003ст*

Введение в Главу 1.2.

Важнейшим направлением повышения уровня *интеллекта индивидуальных интеллектуальных кибернетических систем* является переход к *коллективам индивидуальных интеллектуальных кибернетических систем* и далее к *иерархическим коллективам интеллектуальных кибернетических систем*, членами которых являются как *индивидуальные интеллектуальные кибернетические системы*, так и *коллективы индивидуальных интеллектуальных кибернетических систем*, а также *иерархические коллективы интеллектуальных кибернетических систем*.

Аналогичным образом необходимо повышать уровень интеллекта и *индивидуальных интеллектуальных компьютерных систем* (искусственных *кибернетических систем*). Но при этом надо помнить, что далеко не каждое объединение *интеллектуальных кибернетических систем* (в том числе и *компьютерных систем*) становится *интеллектуальным коллективом*. Для этого необходимо соблюдение дополнительных требований, предъявляемых ко всем членам *интеллектуальных коллективов*. Важнейшим из них является требование высокого уровня ***интероперабельности***, то есть способности к эффективному взаимодействию с другими членами коллектива. Переход от современных *интеллектуальных компьютерных систем* к *интероперабельным интеллектуальным компьютерным системам* является ключевым фактором перехода к ***интеллектуальным компьютерным системам нового поколения***, обеспечивающим существенное повышение уровня автоматизации человеческой деятельности.

Расширение областей применения *интеллектуальных компьютерных систем* требует перехода к решению комплексных задач — задач, решение которых невозможно с помощью одной модели решения задач, одного вида знаний, одной интеллектуальной компьютерной системы.

Для решения комплексных задач необходим:
- Переход к *гибридным* индивидуальным интеллектуальным компьютерным системам, в которых осуществляется конвергенция и интеграция различных моделей решения задач и различных видов знаний;
- Переход к коллективам семантически совместимых самостоятельных интеллектуальных компьютерных систем, в которых обеспечивается:
  - *интероперабельность* объединяемых интеллектуальных компьютерных систем,
  - конвергенция объединяемых интеллектуальных компьютерных систем при сохранении их самостоятельности.

§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения

⇒ *ключевое понятие**:
- *интеллектуальная компьютерная система нового поколения*
- *интероперабельная интеллектуальная компьютерная система*
- *гибридная интеллектуальная компьютерная система*

⇒ *ключевое отношение**:
- *соединение интеллектуальных компьютерных систем**
  ≔ [преобразование множества интеллектуальных компьютерных систем в коллектив, членами (агентами) которого являются эти системы*]
- *глубокая интеграция интеллектуальных компьютерных систем**
  ≔ [быть результатом преобразования множества индивидуальных интеллектуальных компьютерных систем в одну интегрированную индивидуальную интеллектуальную компьютерную систему*]

⇒ *ключевой параметр**:
- *интероперабельность интеллектуальных компьютерных систем^*
- *семантическая совместимость пар интеллектуальных компьютерных систем^*

⇒ *ключевое знание**:
- *Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *Принципы, лежащие в основе интеллектуальных компьютерных систем нового поколения*
- *Отличие данных от знаний*

⇒ *библиографическая ссылка**:
- *Yaghoobirafi K..aAppro fSIiADIS-2022art*
- *Ouksel A.M..SemanIiGIS-1999art*
- *Lanzenberger M..MakinOTKIitSW-2008art*
- *Neiva F.W..TowarPItSC-2016art*
- *Pohl J.Inter atNfISaHP-2004art*
- *Waters J..GlobaIUSS-2009art*

- *Lopes L.S..SemanIitIoT-2022art*
- *Hamilton Inter..Intero aKEftG-2006art*
- *Баринов И.И..ФормиСРКпИИ-2021ст*

Создание различных комплексов взаимодействующих *интеллектуальных компьютерных систем* требует повышения качества не только самих этих систем, но также и качества их взаимодействия. *Интеллектуальные компьютерные системы нового поколения* должны иметь высокий уровень ***интероперабельности***, то есть высокий уровень способности к эффективному, целенаправленному взаимодействию с себе подобными и с пользователями в процессе коллективного (распределенного) и децентрализованного решения сложных задач (см. *Yaghoobirafi K..aAppro fSIiADIS-2022art* , *Ouksel A.M..SemanIiGIS-1999art* , *Lanzenberger M..MakinOTKIitSW-2008art* , *Neiva F.W..TowarPItSC-2016art*, *Pohl J.Inter atNfISaHP-2004art*, *Waters J..GlobaIUSS-2009art*). Уровень ***интероперабельности*** *интеллектуальных компьютерных систем* — это, образно говоря, уровень их "социализации", полезности в рамках различных априори неизвестных сообществ (коллективов) *интеллектуальных систем*.

Уровень ***интероперабельности*** *интеллектуальных компьютерных систем* — это уровень их коммуникационной (социальной) совместимости, позволяющей им самостоятельно формировать коллективы *интеллектуальных компьютерных систем* и их пользователей, а также самостоятельно согласовывать и координировать свою деятельность в рамках этих коллективов при решении сложных задач в частично предсказуемых условиях. Повышение уровня *интероперабельности* интеллектуальных компьютерных систем определяет переход к ***интеллектуальным компьютерным системам нового поколения***, без которых невозможна реализация таких проектов, как *интеллектуальное-предприятие*, *интеллектуальная-больница*, *интеллектуальная-школа*, *интеллектуальный-университет*, *интеллектуальная-кафедра*, *интеллектуальный-дом*, *интеллектуальный-город*, *интеллектуальное-общество* (см. *Lopes L.S..SemanIitIoT-2022art*, *Hamilton Inter..Intero aKEftG-2006art*).

интеллектуальная компьютерная система
:= [интеллектуальная искусственная кибернетическая система]
⇒ *разбиение**:
{• *индивидуальная интеллектуальная компьютерная система*
• *интеллектуальный коллектив интеллектуальных компьютерных систем*
:= [интеллектуальная *многоагентная система*, агенты которой являются *интеллектуальными компьютерными системами*]
⇒ *примечание**:
[Не каждый *коллектив интеллектуальных компьютерных систем* может оказаться интеллектуальным, поскольку уровень интеллекта такого коллектива определяется не только уровнем интеллекта его членов, но также и эффективностью (качеством) их взаимодействия.]
⇒ *разбиение**:
{• *интеллектуальный коллектив индивидуальных интеллектуальных компьютерных систем*
• *иерархический интеллектуальный коллектив интеллектуальных компьютерных систем*
:= [*интеллектуальный коллектив интеллектуальных компьютерных систем*, по крайней мере одним из членов которого является *интеллектуальный коллектив интеллектуальных компьютерных систем*]
}
}

интеллектуальные компьютерные системы нового поколения
⇒ *предъявляемые требования**:
- высокий уровень *интероперабельности*
- высокий уровень *обучаемости*
- высокий уровень *гибридности*
- высокий уровень способности решать *интеллектуальные задачи* (то есть *задачи*, *методы* решения которых и/или требуемая для их решения исходная информация априори неизвестны)
- высокий уровень *синергетичности*

интероперабельность^
:= [способность к эффективному (целенаправленному) взаимодействию с другими самостоятельными субъектами]
:= [способность к партнерскому взаимодействию в решении *комплексных задач*, требующих *коллективной деятельности*]
:= [способность работать в коллективе (в команде)]
:= [уровень социализации]
:= [social skills]

высокий уровень интероперабельности
⇒ *обеспечивается**:
- высоким уровнем *взаимопонимания*
  ⇒ *обеспечивается**:
  - высоким уровнем ***семантической совместимости*** заданного субъекта с другими субъектами заданного коллектива
  - высоким уровнем *способности понимать* сообщения и поведение партнеров
  - высоким уровнем *способности быть понятной* для партнеров:
    - способности понятно и обоснованно формулировать свои предложения и информацию, полезную для решения текущих задач
    - способности действовать и комментировать свои действия так, чтобы они и их мотивы были понятны партнерам
  - высоким уровнем *способности к повышению уровня семантической совместимости* со своими партнерами
- высоким уровнем *договороспособности*, то есть способности согласовывать с партнерами свои планы и намерения в целях своевременного обеспечения высокого качества коллективного результата
- высоким уровнем *способности к децентрализованной координации* своих действий с действиями партнеров в непредсказуемых (нештатных) обстоятельствах
- высоким уровнем способности разделять ответственность с партнерами
- высоким уровнем *способности к минимизации негативных последствий конфликтных ситуаций* с другими субъектами
  ⇒ *обеспечивается**:
  - высоким уровнем *способности к предотвращению возникновения конфликтных ситуаций*
  - *соблюдением этических норм* и правил, препятствующих возникновению разрушительных последствий конфликтных ситуаций
  - высоким уровнем *способности разделять ответственность* с партнерами за своевременное и качественное достижение общей цели

семантическая совместимость^
:= [степень согласованности (совпадения) систем *понятий* и других *ключевых знаков*, используемых заданными взаимодействующими субъектами]
⇒ *примечание**:
[Обеспечение *семантической совместимости* требует формализации *смыслового представления информации*.]

способность разделять ответственность с партнерами, являющаяся необходимым условием децентрализованного управления коллективной деятельностью
⇒ *обеспечивается**:
- *способностью к мониторингу* и анализу коллективно выполняемой деятельности
- *способностью оперативно информировать партнеров* о неблагоприятных ситуациях, событиях, тенденциях, а также инициировать соответствующие коллективные действия

высокий уровень обучаемости интеллектуальной компьютерной системы нового поколения
⇒ *пояснение**:
[Важнейшим направлением повышения уровня автоматизации человеческой деятельности является повышение уровня автоматизации не только проектирования интеллектуальной компьютерной системы, но и комплексной поддержки всех остальных этапов жизненного цикла *интеллектуальной компьютерной системы*. В частности, это касается модернизации (совершенствования, реинжиниринга) интеллектуальной компьютерной системы непосредственно в ходе их эксплуатации. Для того, чтобы обеспечить высокий уровень автоматизации такой модернизации, необходимо существенно повысить ***уровень самообучаемости*** *интеллектуальной компьютерной системы* для того, что они сами (самостоятельно) могли себя модернизировать (самосовершенствовать) входе своего целевого функционирования.]

высокий уровень обучаемости
⇒ *обеспечивается**:
- высоким уровнем *гибкости информации*, хранимой в памяти интеллектуальной системы
- высоким уровнем *качества стратификации информации*, хранимой в памяти интеллектуальной системы (стратифицированностью *базы знаний*)
- высоким уровнем *рефлексивности* интеллектуальной системы
- высоким уровнем *способности исправлять свои ошибки* (в том числе устранять противоречия в своей *базе знаний*)

- высоким уровнем *познавательной активности*
- низким уровнем *ограничений на вид приобретаемых знаний и навыков* (отсутствие таких ограничений означает потенциальную *универсальность* интеллектуальной системы и предполагает высокий уровень ее гибридности)

обучаемость^
:= [способность быстро и качественно приобретать новые *знания* и *навыки*, а также совершенствовать уже приобретенные *знания* и *навыки*]

гибридность^
:= [степень многообразия используемых *видов знаний* и *моделей решения задач* и уровень эффективности их совместного использования]
:= [индивидуальная способность решать *комплексные задачи*, требующие использования различных *видов знаний*, а также различных комбинаций различных *моделей решения задач*]

высокий уровень гибридности
⇒ *обеспечивается**:
- высокой степенью многообразия используемых *видов знаний* и *моделей решения задач*
- высокой степенью *конвергенции* и глубокой *интеграции* (степенью взаимопроникновения) различных *видов знаний* и *моделей решения задач*
- способностью неограниченно расширять уровень своей *гибридности*

Подчеркнем, что *гибридность* и *интероперабельность интеллектуальных компьютерных систем нового поколения* предполагает отказ от известной парадигмы "черных ящиков", поскольку:
- все многообразие моделей решения задач *гибридной интеллектуальной компьютерной системы* должно интерпретироваться на одной общей *универсальной платформе*;
- доступность информации о том, как устроен каждый используемый метод, модель решения задач, каждый субъект существенно повышает качество их *координации* при *совместном решении комплексных задач*;
- появляется возможность некоторые методы, модели решения задач и целые субъекты (например, *интеллектуальные компьютерные системы*) использовать для совершенствования (повышения качества) других методов, моделей и субъектов.

Особо необходимо отметить следующие характеристики *интеллектуальных компьютерных систем нового поколения*:
- ***степень конвергенции***, унификации и стандартизации *интеллектуальных компьютерных систем* и их компонентов и соответствующая этому ***степень интеграции*** (глубина интеграции) *интеллектуальных компьютерных систем* и их компонентов;
- ***семантическая совместимость*** между *интеллектуальными компьютерными системами* в целом и *семантическая совместимость* между компонентами каждой *интеллектуальной компьютерной системы* (в частности, совместимость между различными *видами знаний* и различными *моделями обработки знаний*), которые являются основными показателями степени ***конвергенции*** (сближения) между *интеллектуальными компьютерными системами* и их компонентами.

Особенность указанных характеристик *интеллектуальных компьютерных систем* их компонентов заключается в том, что они играют важную роль при решении всех ключевых задач современного этапа развития *Искусственного интеллекта* и тесно связаны друг с другом.

Заметим также, что перечисленные требования, предъявляемые к *интеллектуальным компьютерным системам нового поколения*, направлены на преодоление проклятия *вавилонского столпотворения* как внутри *интеллектуальных компьютерных систем нового поколения* (между внутренними *информационными процессами* решения различных задач), так и между взаимодействующими самостоятельными *интеллектуальными компьютерными системами нового поколения* в процессе коллективного решения *комплексных задач*.

На современном этапе эволюции *интеллектуальных компьютерных систем* для существенного расширения областей их применения и качественного повышения уровня автоматизации человеческой деятельности:
- необходим переход к созданию семантически совместимых ***интеллектуальных компьютерных систем нового поколения***, ориентированных не только на индивидуальное, но и на коллективное (совместное) решение *комплексных задач*, требующих скоординированной деятельности нескольких самостоятельных интеллектуальных компьютерных систем и использования различных моделей и методов в непредсказуемых комбинациях, что необходимо для существенного расширения сфер применения *интеллектуальных компьютерных систем*, для перехода от автоматизации локальных видов и областей *человеческой деятельности* к комплексной автоматизации более крупных (объединенных) видов и областей этой деятельности;
- необходима разработка ***Общей формальной теории и стандарта интеллектуальных компьютерных систем нового поколения***;

- необходима разработка ***Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения***, которая включает в себя поддержку *проектирования* этих систем (как начального этапа их жизненного цикла) и обеспечение их *совместимости* на всех этапах их жизненного цикла;
- необходима ***конвергенция*** и ***унификация*** *интеллектуальных компьютерных систем нового поколения* и их компонентов;
- необходима реализация "бесшовной", "диффузной", взаимопроникающей, ***глубокой интеграции семантически смежных компонентов интеллектуальных компьютерных систем***, то есть интеграции, при которой отсутствуют четкие границы ("швы") интегрируемых (соединяемых) компонентов, и которая может осуществляться автоматически. Это означает переход к ***гибридным интеллектуальным компьютерным системам***;
- необходимо соблюдение ***Принципа бритвы Оккама*** — максимально возможное структурное упрощение *интеллектуальных компьютерных систем нового поколения*, исключение эклектичных решений;
- необходима ориентация на потенциально ***универсальные*** (то есть способные быстро приобретать любые знания и навыки), ***синергетические*** *интеллектуальные компьютерные системы* с "сильным" интеллектом.

интеллектуальная компьютерная система нового поколения

⇒ *принципы, лежащие в основе**:

- *смысловое представление знаний* в памяти *интеллектуальных компьютерных систем*, предполагающее отсутствие *омонимических знаков*, которые в разных контекстах обозначают разные сущности, а также отсутствие *синонимии*, то есть пар синонимичных *знаков*, которые обозначают одну и ту же сущность
- смысловое представление информационной конструкции в общем случае имеет нелинейный (графовый) характер представления информации, который является *рафинированной семантической сетью*
- фрактальный характер (масштабируемое самоподобие) структуризации представляемых знаний в базах знаний
- использование общего для всех интеллектуальных компьютерных систем *универсального языка смыслового представления знаний* в памяти *интеллектуальных компьютерных систем*, обладающий максимально простым *синтаксисом*, обеспечивающий представление любых *видов знаний* и имеющий неограниченные возможности перехода от *знаний* к *метазнаниям*. Простота синтаксиса *информационных конструкций* указанного *языка* позволяет называть эти конструкции *рафинированными семантическими сетями*
- *структурно-перестраиваемая (графодинамическая) организация памяти* интеллектуальных компьютерных систем, при которой обработка знаний сводится не столько к изменению состояния хранимых *знаков*, сколько к изменению конфигурации связей между этими *знаками*
- *семантически неограниченный ассоциативный доступ к информации*, хранимой в памяти *интеллектуальных компьютерных систем*, по заданному образцу произвольного размера и произвольной конфигурации
- универсальная ситуационная многоагентная модель обработки знаний, ориентированная на обработку смыслового представления информации в ассоциативной графодинамической памяти, *децентрализованное ситуационное управление информационными процессами* в памяти *интеллектуальных компьютерных систем*, реализованное с помощью *агентно-ориентированной модели обработки баз знаний*, в котором *инициирование* новых *информационных процессов* осуществляется не путем передачи управления соответствующим априори известным процедурам, а в результате возникновения соответствующих *ситуаций* или *событий в памяти интеллектуальной компьютерной системы*, поскольку «основная проблема компьютерных систем состоит не в накоплении знаний, а в умении активизировать нужные знания в процессе решения задач» (Поспелов Д. А.). Такой многоагентный процесс обработки информации представляет собой *деятельность*, выполняемую некоторым коллективом самостоятельных *информационных агентов* (агентов обработки информации), условием инициирования каждого из которых является появление в текущем состоянии *базы знаний* соответствующей этому агенту *ситуации* и/или *события*. «Выбор многоагентных технологий объясняется тем, что в настоящее время любая сложная производственная, логистическая или другая система может быть представлена набором взаимодействий более простых систем до любого уровня детальности, что обеспечивает фрактально-рекурсивный принцип построения многоярусных систем, построенных как открытые цифровые колонии и экосистемы ИИ. В основе многоагентных технологий лежит распределенный или децентрализованный подход к решению задач, при котором динамически обновляющаяся информация в распределенной сети интеллектуальных агентов обрабатывается непосредственно у агентов вместе с локально доступной информацией от "соседей". При этом существенно сокращаются как ресурсные и временные затраты на коммуникации в сети, так и время на обработку и принятие решений в центре системы (если он все-таки есть).»

  ⇐ *цитата**:

  Баринов И.И..ФормиСРКпИИ-2021ст стр. 270
- агентно-ориентированная модель обработки знаний в памяти интеллектуальной компьютерной системы, обеспечивающая высокую степень *интероперабельности* между внутренними агентами индивидуальной

интеллектуальной компьютерной системы, взаимодействующими через общую память (это, фактически, "внутренняя" интероперабельность интеллектуальной компьютерной системы нового поколения)

- Переход к *семантическим моделям решения задач*, в основе которых лежит учет не только синтаксических (структурных) аспектов обрабатываемой информации, но также и семантических (смысловых) аспектов этой информации — “From data science to knowledge science”
- ***онтологическая модель баз знаний*** *интеллектуальных компьютерных систем*, то есть онтологическая структуризация всей информации, хранимой в памяти *интеллектуальной компьютерной системы*, предполагающая четкую *стратификацию базы знаний* в виде иерархической системы *предметных областей* и соответствующих им *онтологий*, каждая из которых обеспечивает семантическую *спецификацию* всех *понятий*, являющихся ключевыми в рамках соответствующей *предметной области*
- ***онтологическая локализация решения задач*** в *интеллектуальных компьютерных системах*, предполагающая локализацию *области действия* каждого хранимого в памяти *метода* и каждого *информационного агента* в соответствии с *онтологической моделью* обрабатываемой *базы знаний*. Чаще всего, такой *областью действия* является одна из *предметных областей* либо одна из *предметных областей* вместе с соответствующей ей *онтологии*
- ***онтологическая модель интерфейса*** *интеллектуальной компьютерной системы,* в состав которой входит:
  - онтологическое описание *синтаксиса* всех *языков*, используемых *интеллектуальной компьютерной системой* для *общения* с внешними *субъектами*
  - онтологическое описание *денотационной семантики* каждого *языка*, используемого *интеллектуальной компьютерной системой* для *общения* с внешними *субъектами*
  - семейство *информационных агентов*, обеспечивающих *синтаксический анализ*, *семантический анализ* (перевод на внутренний смысловой язык) и *понимание* (погружение в *базу знаний*) любого введенного *сообщения*, принадлежащего любому *внешнему языку*, полное онтологическое описание которого находится в базе знаний *интеллектуальной компьютерной системы*
  - семейство *информационных агентов*, обеспечивающих *синтез сообщений*, которые (1) адресуются внешним субъектам, с которыми общается *интеллектуальная компьютерная система*, (2) *семантически эквивалентны* заданным *фрагментам базы знаний* интеллектуальной компьютерной системы, определяющим *смысл* передаваемых *сообщений*, (3) принадлежат одному из *внешних языков*, полное онтологическое описание которого находится в *базе знаний* интеллектуальной компьютерной системы
- *семантически дружественный характер пользовательского интерфейса*, обеспечиваемый (1) формальным описание в базе знаний средства управления пользовательским интерфейсом и (2) введением в состав *интеллектуальной компьютерной системы* соответствующих help-подсистем, обеспечивающих существенное снижение языкового барьера между пользователями и *интеллектуальными компьютерными системами*, что существенно повысит эффективность *эксплуатации интеллектуальных компьютерных систем*
- *минимизация негативного влияния человеческого фактора* на эффективность *эксплуатации интеллектуальных компьютерных систем* благодаря реализации интероперабельного (партнерского) стиля взаимодействия не только между самими *интеллектуальными компьютерными системами*, но также и между *интеллектуальными компьютерными системами* и их пользователями. Ответственность за качество совместной деятельности должно быть распределено между всеми партнерами
- ***мультимодальность*** (гибридный характер) *интеллектуальной компьютерной системы*, что предполагает:
  - многообразие *видов знаний*, входящих в состав *базы знаний* интеллектуальной компьютерной системы
  - многообразие *моделей решения задач,* используемых *решателем задач* интеллектуальной компьютерной системы
  - многообразие *сенсорных каналов*, обеспечивающих *мониторинг* состояния *внешней среды* интеллектуальной компьютерной системы
  - многообразие *эффекторов*, осуществляющих *воздействие* на *внешнюю среду*
  - многообразие *языков общения* с другими субъектами (с пользователями, с интеллектуальными компьютерными системами)
- ***внутренняя семантическая совместимость*** между компонентами *интеллектуальной компьютерной системы* (то есть максимально возможное введение общих, совпадающих *понятий* для различных фрагментов хранимой *базы знаний*), являющаяся формой ***конвергенции*** и *глубокой интеграции* внутри *интеллектуальной компьютерной системы* для различного вида *знаний* и различных *моделей решения задач*, что обеспечивает эффективную реализацию *мультимодальности интеллектуальной компьютерной системы*
- ***внешняя семантическая совместимость*** между различными *интеллектуальными компьютерными системами*, выражающаяся не только в общности используемых *понятий*, но и в общности базовых

знаний и являющаяся необходимым условием обеспечения высокого уровня *интероперабельности* интеллектуальных компьютерных систем

- ориентация на использование *интеллектуальных компьютерных систем* как *когнитивных агентов* в составе ***иерархических многоагентных систем***
- фрактальный характер (масштабируемое самоподобие) структуризации иерархических коллективов интеллектуальных компьютерных систем нового поколения
- ***платформенная независимость* интеллектуальных компьютерных систем**, предполагающая:
  - четкую *стратификацию* каждой *интеллектуальной компьютерной системы* (1) на *логико-семантическую модель*, представленную ее *базой знаний*, которая содержит не только *декларативные знания*, но и знания, имеющие *операционную семантику*, и (2) на *платформу*, обеспечивающую *интерпретацию* указанной *логико-семантической модели*
  - универсальность указанной *платформы* интерпретации *логико-семантической модели интеллектуальной компьютерной системы*, что дает возможность каждой такой *платформе* обеспечивать интерпретацию любой *логико-семантической модели интеллектуальной компьютерной системы*, если эта модель представлена на том же *универсальном языке смыслового представления информации*
  - многообразие вариантов реализации *платформ интерпретации логико-семантических моделей интеллектуальных компьютерных систем* — как вариантов, программно реализуемых на *современных компьютерах*, так и вариантов, реализуемых в виде *универсальных компьютеров нового поколения*, ориентированных на использование в *интеллектуальных компьютерных системах* нового поколения (такие компьютеры мы назвали *ассоциативными семантическими компьютерами*)
  - легко реализуемую возможность переноса (переустановки) логико-семантической модели (*базы знаний*) любой *интеллектуальной компьютерной системы* на любую другую *платформу интерпретации логико-семантических моделей*
- изначальная ориентация *интеллектуальных компьютерных систем нового поколения* на использование ***универсальных ассоциативных семантических компьютеров*** (компьютеров нового поколения) в качестве *платформы интерпретации логико-семантических моделей* (баз знаний) *интеллектуальных компьютерных систем*

В настоящее время разработано большое количество различного вида *моделей решения задач*, моделей представления и обработки знаний различного вида. Но в разных *интеллектуальных компьютерных системах* могут быть востребованы разные комбинации этих моделей. При разработке и реализации различных *интеллектуальных компьютерных систем* соответствующие методы и средства должны гарантировать *логико-семантическую совместимость* разрабатываемых компонентов и, в частности, их способность использовать общие *информационные ресурсы*. Для этого, очевидно, необходима *унификация* указанных моделей.

Многообразие различных видов интеллектуальных компьютерных систем и, соответственно, многообразие используемых ими комбинаций моделей представления знаний и решения задач определяется:
- многообразием назначения интеллектуальных компьютерных систем и вида окружающей их среды;
- многообразием различных видов хранимых знаний;
- многообразием моделей обработки знаний и решений задач;
- многообразием различных видов сенсорных и эффекторных подсистем.

Следует выделить следующие аспекты *совместимости* моделей представления и обработки знаний в *интеллектуальных компьютерных системах*:
- синтаксический;
- семантический (согласованность систем понятий, их денотационной семантики);
- функциональный (операционный).

Следует также отличать:
- *совместимость* между компонентами *интеллектуальных компьютерных систем*;
- *совместимость* между верхним логико-семантическим уровнем используемых моделей представления и обработки знаний и различными уровнями их интерпретации вплоть до аппаратного уровня;
- *совместимость* между индивидуальными интеллектуальными компьютерными системами;
- *совместимость* между индивидуальными интеллектуальными компьютерными системами и их пользователями;
- *совместимость* между коллективами интеллектуальных компьютерных системам.

следует отличать*
∋ {• *данные*
:= [информационная конструкция, обрабатываемая с помощью программы традиционного языка программирования]
• *знание*

:= [семантически целостный фрагмент базы знаний]

}

⇒ *отличие**:

[Для каждого знания всегда известен язык, на котором это знание представлено и денотационная семантика которого задана. При этом указанный язык имеет достаточно большую семантическую мощность, а в идеале является универсальным языком. В отличие от этого структуризация данных для традиционных программ осуществляется в целях упрощения самих этих программ и, следовательно, для разных программ в общем случае осуществляется по-разному. Таким образом, при разработке традиционных программ представление обрабатываемых данных осуществляется в общем случае на разных языках, денотационная семантика которых нигде не документируется и известна только разработчикам программ. Другими словами, данные для разных программ имеют денотационную семантику не только разную, но еще и априори неизвестную. По сути это форма проявления *вавилонского столпотворения* в традиционных языках программирования, которые образно говоря "хромают на одну ногу", формализуя методы обработки информации, но не формализуя семантику обрабатываемой информации.]

§ 1.2.2. Принципы, лежащие в основе смыслового представления информации

⇒ *ключевое понятие**:

- *смысловое представление информации*
  := [смысл]
- *семантическая сеть*
- *рафинированная семантическая сеть*
- *граф знаний*
  := [представление сложноструктурированного знания в виде графовой структуры]
- *универсальный язык семантических сетей*
  := [универсальный язык, информационными конструкциями которого являются семантические сети]

⇒ *ключевое знание**:

- *Принципы, лежащие в основе смыслового представления информации*

⇒ *библиографическая ссылка**:

- *Мартынов В.В.СемиоОИ-1974кн*
- *Мельчук И.А.ОпытТЛМСТ-1999кн*
- *Мельчук И.А.КакНМЛ-1998ст*
- *Харламов А.А..СеманСкФОР-2011ст*
- *Москин Н.Д.оПредсЗсПССвИ-2011ст*
- *Тимченко В.А.МетодПКСС-2013ст*
- *Массель Л.В..СеманТнОИОК-2013ст*
- *Ефименко И.В..УСКМТЛС-2014ст*
- *Голенков В.В.ред.ОткрыСТПИ-2014ст*
- *Голенков В.В..СтрукСП-2014ст*
- *Загорулько Ю.А..оФормаСОЗвИиИСнОО-2014ст*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Golenkov V.V..tStand oICSaaK-2020art*

Предлагаемый подход к решению проблем, препятствующих дальнейшей эволюции компьютерных систем и технологий — стандартизация моделей представления и обработки информации

Анализ проблем эволюции компьютерных систем разного уровня сложности, разного уровня обучаемости и интеллектуальности, разного назначения показывает, что проклятие "вавилонского столпотворения" и, как следствие, несовместимость, дублирование и субъективизм согласовываемых информационных ресурсов и моделей их обработки нас преследует везде:

- и в развитии традиционных компьютерных систем;
- и в развитии технологий искусственного интеллекта;
- и в развитии методов и средств информатизации научной и инженерной деятельности.

Рассматривая проблему обеспечения совместимости информационных ресурсов и моделей их обработки, следует говорить о разных аспектах решения этой проблемы:

- об обеспечении совместимости между различными компонентами компьютерных систем, а также между целостными компьютерными системами, входящими в коллективы компьютерных систем;
- об обеспечении совместимости, то есть высокого уровня взаимопонимания между различными компьютерными системами и их пользователями;

- об обеспечении междисциплинарной совместимости, то есть конвергенции различных областей знаний;
- о методах и средствах постоянного мониторинга и восстановления совместимости в условиях интенсивной эволюции компьютерных систем и их пользователей, которая часто нарушает достигнутую совместимость (согласованность) и требует дополнительных усилий на ее восстановление.

Суть предлагаемого нами подхода к решению проблем эволюции компьютерных систем заключается, во-первых, в объединении всех указанных выше направлений эволюции компьютерных систем (как общих направлений, так и частных) и, во-вторых, в трактовке проблемы обеспечения **совместимости** различных видов знаний, различных моделей решения задач, различных компьютерных систем как **ключевой проблемы** эволюции компьютерных систем, решение которой существенно упростит решение и многих других проблем.

Так, например, без обеспечения совместимости информационных ресурсов, используемых в разных компьютерных системах, а также информационных ресурсов, представляющих знания различного семантического вида невозможно:

- ни создавать **коллективы компьютерных систем**, способные координировать свои действия при кооперативном расширении сложных задач;
- ни создавать **гибридные компьютерные системы**, которые способны при решении сложных комплексных задач использовать всевозможные сочетания разных видов знаний и разных моделей решения задач;
- ни использовать **компонентную методику проектирования** компьютерных систем **на всех уровнях** иерархии проектируемых систем.

О какой информационной совместимости и взаимопонимании (в том числе между специалистами) можно говорить при наличии ужасающей понятийной и терминологической неряшливости, терминологического псевдотворчества, в том числе, в области информатики.

Говоря о **совместимости** компьютерных систем и их компонентов, а также совместимости компьютерных систем с пользователями, следует отметить неоднозначность трактовки термина “совместимость”. В этой связи следует отличать:

- совместимость как один из факторов обучаемости, как **способность** к быстрому повышению уровня согласованности (интеграции, взаимопонимания). Сравните обучаемость как **способность** к быстрому расширению знаний и навыков, но никак не характеристика объема и качества приобретенных знаний и навыков;
- совместимость как характеристика достигнутого уровня согласованности (интеграции, взаимопонимания).

Аналогичным образом интеллект компьютерной системы с одной стороны можно трактовать как **уровень** (объем и качество) приобретенных знаний и навыков, а с другой стороны как **способность** к быстрому расширению и совершенствованию знаний и навыков, то есть как **скорость** повышения уровня знаний и навыков.

Кроме того, следует говорить не только о **способности** к быстрому повышению уровня согласованности и не только о достигнутом уровне согласованности, но и о самом **процессе** повышения уровня согласованности и, прежде всего, о перманентном процессе восстановления (поддержки, сохранения) достигнутого уровня согласованности, поскольку в ходе эволюции компьютерных систем и их пользователей (то есть в ходе расширения и повышения качества их знаний и навыков) уровень их согласованности может понижаться.

Главным фактором обеспечения совместимости различных видов знаний, различных моделей решения задач и различных компьютерных систем в целом является

- унификация (стандартизация) представления информации в памяти компьютерных систем;
- унификация принципов организации обработки информации в памяти компьютерных систем.

Унификация представления информации, используемой в компьютерных системах, предполагает:

- синтаксическую унификацию используемой информации — унификацию формы представления (кодирования) этой информации. При этом следует отличать:
  - кодирование информации в памяти компьютерной системы (внутреннее представление информации);
  - внешнее представление информации, обеспечивающее однозначность интерпретации (понимания, трактовки) этой информации разными пользователями и разными компьютерными системами;
- семантическую унификацию используемой информации в основе которой лежит согласование и точная спецификация всех (!) используемых понятий (концептов) с помощью иерархической системы формальных онтологий.

Важно отметить, что грамотная унификация (стандартизация) должна не ограничивать творческую свободу разработчика, а гарантировать ***совместимость*** его результатов с результатами других разработчиков. Подчеркнем также, что текущая версия любого ***стандарта*** — это не догма, а только опора для дальнейшего его совершенствования.

Целью качественного *стандарта* является не только обеспечения совместимости технических решений, но и минимализация дублирования (повторения) таких решений. Один из важных критериев качества *стандарта* — ничего лишнего.

стандарт
:= [знания о структуре и принципах функционирования искусственных систем соответствующего класса]
:= [онтология искусственных систем некоторого класса]
:= [теория искусственных систем некоторого класса]

Стандарты, как и другие важные для человечества знания, должны быть формализованы и должны постоянно совершенствоваться с помощью специальных интеллектуальных компьютерных систем, поддерживающих процесс эволюции стандартов путем согласования различных точек зрения.

смысловое представление информации
:= [запись (представление) информационной конструкции на смысловом уровне]
:= [информационная конструкция синтаксическая структура которой близка ее смыслу, то есть близка описываемой конфигурации связей между описываемыми сущностями]
:= [смысловое представление информационной конструкции]
⊃ *семантическая сеть*
⊃ *рафинированная семантическая сеть*

рафинированная семантическая сеть
⇒ *принципы, лежащие в основе**:
- Каждый элемент (синтаксически атомарный фрагмент) рафинированной семантической сети является знаком одной из описываемых сущностей
- Каждая сущность, описываемая рафинированной семантической сетью, должна быть представлена своим знаком, который является элементом этой сети
- В рамках каждой отдельной рафинированной семантической сети отсутствует синонимия разных знаков, а также отсутствуют омонимичные знаки
- Многообразие сущностей, описываемых рафинированными семантическими сетями, ничем не ограничивается. Соответственно этому, семантическая типология элементов рафинированных семантических сетей является весьма богатой
- Особым видом элементов рафинированных семантических систем являются знаки связей между другими элементами этих сетей. При этом, связываемыми элементами (то есть элементами, которые инцидентны указанным знакам связей) могут быть также и знаки других связей. Чаще всего знак связи между элементами рафинированной семантической сети является отражением связи между сущностями, которые обозначаются указанными элементами. Но в некоторых случаях знак связи между элементами рафинированной семантической сети может быть отражением, например, связи между одной описываемой сущностью и знаком другой описываемой сущности

Объективным ориентиром для **унификации представления информации** в памяти компьютерных систем и ключом к решению многих проблем эволюции компьютерных систем и технологий является **формализация смысла представляемой информации**.

Согласно В. В. Мартынову (см. *Мартынов В.В.СемиоОИ-1974кн*), «фактически всякая мыслительная деятельность человека (не только научная), как полагают многие ученые, использует внутренний семантический код, на который переводят с естественного языка и с которого переводят на естественный язык. Поразительная способность человека к идентификации огромного множества структурно различных фраз с одинаковым смыслом и способность **запомнить смысл вне этих фраз** убеждает нас в этом.»

Приведем также слова И.А. Мельчука (см. *Мельчук И.А.ОпытТЛМСТ-1999кн*):

«Идея была следующая – язык надо описывать следующим образом: надо уметь записывать смыслы фраз. Не фразы, а их смыслы, что отдельно. Плюс построить систему, которая по смыслу строит фразу. Это та область или тот поворот исследований, при котором интуиция способного лингвиста работает лучше всего: как выразить на данном языке данный смысл. Это – то, для чего лингвистов учат.

Лингвистический смысл научного текста – это совсем не то, что ты, читая его, из него извлекаешь. Это, очень грубо говоря, инвариант синонимических парафраз. Ты можешь один и тот же смысл выразить очень многими. Когда ты говоришь, то можешь сказать по-разному: “Сейчас я налью тебе вина”, или: “Дай, я тебе предложу вина”, или: “Не выпить ли нам по бокалу?”, – все это имеет один и тот же смысл. И вот можно придумать, как записывать этот смысл. Именно его. Не фразу, а смысл. И работать надо от этого смысла к реальным фразам. Синтаксис там по дороге тоже нужен, но он нужен именно по дороге, он не может быть ни конечной целью, ни начальной точкой. Это – промежуточное дело.» (см. *Мельчук И.А.КакНМЛ-1998ст*).

Уточнение принципов **смыслового представления информации** основано, во-первых, на четком противопоставление **внутреннего языка компьютерной системы**, используемого для хранения информации в памяти компьютера, и **внешних языков компьютерной системы**, используемых для общения (обмена сообщениями) компьютерной системы с пользователями и другими компьютерными системами (смысловое представление используется

исключительно для **внутреннего представления** информации в памяти компьютерной системы), и, во-вторых, на максимально возможном упрощении синтаксиса внутреннего языка компьютерной системы при обеспечении универсальности путем исключения из такого внутреннего универсального языка средств, обеспечивающих коммуникационную функцию языка (то есть обмен сообщениями).

Так, например, для внутреннего языка компьютерной системы излишними являются такие коммуникационные средства языка, как союзы, предлоги, разделители, ограничители, склонения, спряжения и другие.

Внешние языки компьютерной системы могут быть как близки ее внутреннему языку, так и весьма далеки от него (как, например, естественные языки).

смысл — это **абстрактная** знаковая конструкция, принадлежащая внутреннему языку компьютерной системы, являющаяся **инвариантом** максимального класса семантически эквивалентных знаковых конструкций (текстов), принадлежащих самым разным языкам, и удовлетворяющая следующим требованиям:

- **универсальность** — возможность представления любой информации;
- **отсутствие синонимии знаков** (многократного вхождения знаков с одинаковыми денотатами);
- **отсутствие дублирования информации** в виде семантически эквивалентных текстов (не путать с логической эквивалентностью);
- **отсутствие омонимичных знаков** (в том числе местоимений);
- **отсутствие у знаков внутренней структуры** (атомарный характер знаков);
- **отсутствие склонений, спряжений** (как следствие отсутствия у знаков внутренней структуры);
- **отсутствие фрагментов** знаковой конструкции, **не являющихся знаками** (разделителей, ограничителей, и так далее);
- **выделение знаков связей**, компонентами которых могут быть любые знаки, с которыми знаки связей связываются синтаксически задаваемыми отношениями инцидентности.

Следствием указанных принципов смыслового представления информации в памяти компьютерной системы является то, что знаки сущностей, входящие в смысловое представление информации, **не являются именами** (терминами) и, следовательно, не привязаны ни к какому естественному языку и не зависят от субъективных терминотворческих пристрастий различных авторов. Это значит, что при коллективной разработке смыслового представления каких-либо информационных ресурсов терминологические споры исключены.

Следствием указанных принципов смыслового представления информации является также то, что эти принципы приводят к нелинейным знаковым конструкциям (к графовым структурам), что усложняет реализацию памяти компьютерных систем, но существенно упрощает ее логическую организацию (в частности, ассоциативный доступ).

Нелинейность смыслового представления информации обусловлена тем, что:

- каждая описываемая сущность, то есть сущность, имеющая соответствующий ей знак, может иметь неограниченное число связей с другими описываемыми сущностями;
- каждая описываемая сущность в смысловом представлении имеет единственный знак, так как синонимия знаков здесь запрещена;
- все связи между описываемыми сущностями описываются (отражаются, моделируются) связями между знаками этих описываемых сущностей.

Суть **универсального смыслового представления информации** можно сформулировать в виде следующих положений:

- Смысловая знаковая конструкция трактуется как множество знаков, взаимно-однозначно обозначающих различные сущности (денотаты этих знаков) и множество связей между этими знаками;
- Каждая связь между знаками трактуется, с одной стороны, как множество знаков, связываемых этой связью, а, с другой стороны, как описание (отражение, модель) соответствующей связи, которая связывает денотаты указанных знаков или денотаты одних знаков непосредственно с другими знаками, или сами эти знаки. Примером первого вида связи между знаками является связь между знаками материальных сущностей, одна из которых является частью другой. Примером второго вида связи между знаками является связь между знаком множества знаков и одним из знаком, принадлежащих этому множеству, а также связь между знаком и знаком файла, являющегося электронным отражением структуры представления указанного знака во внешних знаковых конструкциях. Примерами третьего вида связи между знаками является связь между синонимичными знаками;
- Денотатами знаков могут быть (1) не только конкретные (константные, фиксированные), но и произвольные (переменные, нефиксированные) сущности, "пробегающие" различные множества знаков (возможных значений), (2) не только реальные (материальные), но и абстрактные сущности (например, числа, точки различных абстрактных пространств), (3) не только "внешние", но и "внутренние" сущности, являющиеся множествами знаков, входящих в состав той же самой знаковой конструкции.

Ключевым свойством языка смыслового представления информации является однозначность представления информации в памяти каждой компьютерной системы, то есть отсутствие семантически эквивалентных знаковых конструкций, принадлежащих смысловому языку и хранимых в одной смысловой памяти. При этом логическая

эквивалентность таких знаковых конструкций допускается и используются, например, для компактного представления некоторых знаний, хранимых в смысловой памяти.

Тем не менее, логической эквивалентностью хранимых в памяти знаковых конструкций увлекаться не следует, так как **логически эквивалентные** знаковые конструкции – это представление одного и того же знания, но с помощью **разных наборов понятий**. В отличие от этого **семантически эквивалентные** знаковые конструкции – это представление одного и того же знания с помощью одних и тех же понятий. Очевидно, что многообразие возможных вариантов представления одних и тех же знаний в памяти компьютерной системы существенно усложняет решение задач. Поэтому, полностью исключив **семантическую эквивалентность** в смысловой памяти, необходимо стремиться к минимизации **логической эквивалентности**. Для этого необходимо грамотное построение системы используемых понятий в виде иерархической системы формальных онтологий (см. *Главу 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*).

Важным этапом создания универсального формального способа смыслового кодирования знаний был разработанный В.В. Мартыновым Универсальный Семантический Код (УСК) (см. *Мартынов В.В.СемиоОИ-1974кн*).

В качестве **стандарта** универсального смыслового представления информации **в памяти компьютерных систем** нами предложен ***SC-код*** (Semantic Computer Code). В отличие от УСК В.В. Мартынова он, во-первых, носит нелинейный характер и, во-вторых, специально ориентирован на кодирование информации в памяти компьютеров нового поколения, ориентированных на разработку семантически совместимых интеллектуальных систем и названных нами **семантическими ассоциативными компьютерами**. Таким образом, основным лейтмотивом предлагаемого нами смыслового представления информации является ориентация на формальную модель памяти нефоннеймановского компьютера, предназначенного для реализации интеллектуальных систем, использующих смысловое представление информации. Особенностями такого представления являются следующие:

- ассоциативность;
- вся информация заключена в конфигурации связей, то есть переработка информации сводится к реконфигурации связей (к графодинамическим процессам);
- прозрачная семантическая интерпретируемость и, как следствие, семантическая совместимость.

Неявная привязка к фон-неймановским компьютерам присутствует во всех известных моделях представления знаний. Одним из примеров такой зависимости, является, например, обязательность именования описываемых объектов.

§ 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения

⇒ *ключевое понятие**:
- *смысловая память*
- *графодинамическая память*
- *ассоциативная память с информационным доступом по образцу произвольного размера и конфигурации*
- *система ситуационного децентрализованного управления информационными процессами*
- *многоагентная система обработки информации в общей памяти*
- *язык смыслового представления задач*
- *универсальный язык смыслового представления знаний*
- *язык смыслового представления методов*
  := [интегрированный язык смыслового представления различного вида программ]
- *инсерционная программа*

⇒ *ключевое знание**:
- *Принципы, лежащие в основе решателей задач индивидуальных интеллектуальных компьютерных систем нового поколения*

⇒ *библиографическая ссылка**:
- *Летичевский А.А..ИнсерП-2003ст*

решатель задач интеллектуальных компьютерных систем нового поколения

⇒ *предъявляемые требования**:
- решатель задач интеллектуальных компьютерных систем нового поколения должен уметь решать интеллектуальные задачи, к числу которых относятся следующие виды задач:
  - некачественно сформулированная задача
    := [задача, формулировка которой содержит различные не-факторы (неполнота, нечеткость, противоречивость (некорректность) и так далее)]

- задача, для решения которой, кроме самой формулировки задачи и соответствующего метода ее решения необходима дополнительная, но априори неизвестно какая информация об объектах, указанных в формулировке (постановке) задачи. При этом указанная дополнительная информация может присутствовать, а может и отсутствовать в текущем состоянии базы знаний интеллектуальных компьютерных систем. Кроме того, для некоторых задач может быть задана (указана) та область базы знаний, использования которой достаточно для поиска или генерации (в частности, логического вывода) указанной дополнительной требуемой информации. Такую область базы знаний будем называть областью решения соответствующей задачи
- задача, для которой соответствующий метод ее решения в текущий момент не известен. Для решения такой задачи можно:
  - переформулировать задачу, то есть сгенерировать (логически вывести) логически эквивалентную формулировку исходной задачи, для которой метод ее решения в текущий момент является известным
  - свести исходную задачу к семейству подзадач, для которых методы их решения в текущий момент известны.

- процесс решения задач в интеллектуальных компьютерных системах нового поколения реализуется коллективом информационных агентов, обрабатывающих базу знаний интеллектуальных компьютерных систем
- управление информационными процессами в памяти интеллектуальных компьютерных систем нового поколения осуществляется децентрализованным образом по принципам ситуационного управления

ситуационное управление
:= [ситуационно-событийное управление]
⇒ *пояснение**:
[управление последовательностью выполнения действий, при котором условием ("триггером") инициирования указанных действий является:
- возникновение некоторых ситуаций (условий, состояний);
- и/или возникновение некоторых *событий*.

]

ситуация
:= [структура, описывающая некоторую временно существующую конфигурацию связей между некоторыми сущностями]
:= [описание временно существующего состояния некоторого фрагмента (некоторой части) некоторой динамической системы]

событие
⊃ *возникновение временной сущности*
:= [появление, рождение, начало существования некоторой временной сущности]
⊃ *исчезновение временной сущности*
:= [прекращение, завершение существования некоторой временной сущности]
⊃ *переход от одной ситуации к другой*
⇒ *примечание**:
[Здесь учитывается не только факт возникновения новой ситуации, но и ее предыстория — то есть та ситуация, которая ей непосредственно предшествует. Так, например, реагируя на аномальное значение какого-либо параметра, нам важно знать:
- какова динамика изменения этого параметра (увеличивается он или уменьшается и с какой скоростью);
- какие меры были предприняты ранее для ликвидации этой аномалии.

]

решатель задач индивидуальной интеллектуальной компьютерной системы нового поколения
⇒ *принципы, лежащие в основе**:
- смысловое представление обрабатываемых знаний
- семантически неограниченный ассоциативный доступ к различным фрагментам знаний, хранимым в памяти интеллектуальных компьютерных систем нового поколения (доступ по заданному образцу произвольного размера и произвольной конфигурации)
- графодинамический характер обработки знаний в памяти, при котором обработка знаний сводится не только к изменению состояния атомарных фрагментов (ячеек) памяти, но и к изменению конфигурации связей между этими атомарными фрагментами

- ситуационное децентрализованное управление процессом обработки знаний, а также процессом организации взаимодействия интеллектуальных компьютерных систем с внешней средой
- использование семантически мощного языка задач, обеспечивающего представление формулировок самых различных задач, которые могут решаться либо в рамках памяти интеллектуальной компьютерной системы, либо во внешней среде и которые осуществляют инициирование соответствующих процессов решения задач
- многоагентный характер реализации процессов решения инициированных задач, в основе которого лежит иерархическая система агентов, каждый из которых активизируются при возникновении в памяти интеллектуальной компьютерной системы соответствующий ситуации или соответствующего события

§ 1.2.4. Принципы, лежащие в основе онтологических моделей мультимодальных интерфейсов интеллектуальных компьютерных систем нового поколения

⇒ *ключевое понятие**:
- *мультимодальный интерфейс*
- *вербальный интерфейс*
- *естественно-языковой интерфейс*
- *внешний язык*
  ≔ [язык обмена сообщениями]
- *внутренний язык*
  ≔ [язык представления информации в памяти кибернетической системы]
- *синтаксис внешнего языка*
- *денотационная семантика внешнего языка*
- *интерфейсная задача*
- *понимание сообщения*
- *синтез сообщения*
- *невербальный интерфейс*
- *сенсор*
  ≔ [рецептор]
- *сенсорная подсистема*
- *мультисенсорная подсистема*
- *сенсорная информация*
- *эффектор*
- *мультиэффекторная подсистема*
- *сенсо-моторная координация*

⇒ *ключевое знание**:
- *Принципы, лежащие в основе интерфейсов интеллектуальных компьютерных систем нового поколения*

интерфейс интеллектуальной компьютерной системы нового поколения

⇒ *принципы, лежащие в основе**:
- интерфейс *интеллектуальной компьютерной системы нового поколения* рассматривается как решатель задач частного вида — *интерфейсных задач*, основными из которых являются:
  - задачи понимания вербальной информации, приобретаемой интеллектуальной компьютерной системой (синтаксический анализ, семантический анализ и погружение в базу знаний интеллектуальной компьютерной системы)
  - задачи понимания невербальной информации, воспринимаемой сенсорными подсистемами интеллектуальной компьютерной системы (анализ изображений, анализ аудио-сигналов, погружение результатов анализа в базу знаний интеллектуальной компьютерной системы)
  - задачи синтеза сообщений, адресуемых внешним субъектам (кибернетическим системам)
- тот факт, что интерфейс *интеллектуальной компьютерной системы нового поколения* является решателем частного вида *задач интеллектуальной компьютерной системы нового поколения*, свойства, лежащие в основе решателей *задач интеллектуальной компьютерной систем нового поколения*, наследуются интерфейсами *интеллектуальной компьютерной систем нового поколения*. Из этого следует, что в основе интерфейса *интеллектуальной компьютерной систем нового поколения* лежит:
  - смысловое представление накапливаемых (приобретаемых знаний)
  - трактовка семантического анализа приобретаемой вербальной информации как процесса перевода этой информации на внутренний язык смыслового представления знаний с последующим погруже-

нием (вводом, интеграцией) результата этого перевода в состав текущего состояния базы знаний *интеллектуальной компьютерной системы нового поколения*
- трактовка синтеза сообщений, адресуемых внешними субъектами как процесса обратного перевода некоторого фрагмента базы знаний с внутреннего языка смыслового представления информации на внешний язык, используемый для общения с заданным субъектом
- агентно-ориентированная организация решения интерфейсных задач, реализуемая соответствующим коллективов внутренних агентов интерфейса *интеллектуальных компьютерных систем нового поколения*, взаимодействующих через общедоступную для них базу знаний *интеллектуальной компьютерной системы нового поколения*

- интерфейс *интеллектуальной компьютерной системы нового поколения* трактуется как специализированная встроенная *интеллектуальная компьютерная система нового поколения*, входящая в состав указанной выше интеллектуальной компьютерной системы, база знаний которой включает в себя:
  - онтологию синтаксиса внутреннего языка смыслового преставления информации
  - онтологию денотационной семантики внутреннего языка смыслового представления информации
  - онтологию синтаксиса всех внешних языков, используемых для общения с внешними субъектами
  - онтологии денотационной семантики всех внешних языков, используемых для общения с внешними субъектами (каждая такая онтология с формальной точки зрения является описанием соответствия между текстами внешних языков и семантически эквивалентными им текстами внутреннего языка смыслового представления информации.

  Подчеркнем при этом, что все указанные онтологии, входящие в состав базы знаний интерфейса интеллектуальных компьютерных систем нового поколения, как и вся остальная информация, входящая в состав этой базы знаний, представляется на внутреннем языке смыслового представления информации, который, соответственно используется в данном случае как метаязык.

интерфейс индивидуальной интеллектуальной компьютерной системы нового поколения
⇒ *принципы, лежащие в основе**:
- интерфейс индивидуальной интеллектуальной компьютерной системы нового поколения является специализированным компонентом решателя задач интеллектуальной компьютерной системы нового поколения, то есть специализированной встроенной (в индивидуальную интеллектуальную компьютерную систему нового поколения) интеллектуальной компьютерной системой нового поколения, ориентированной на решение интерфейсных задач, к которым относятся:
  - понимание принятых сообщений (их перевод на язык внутреннего смыслового представления информации и погружения в текущее состояние базы знаний)
  - синтез передаваемых сообщений (перевод сформированного сообщения с внутреннего языка смыслового представления на используемый внешний язык)
  - первичный анализ приобретаемой сенсорной информации, предполагающий распознавание некоторого семейства первичных образов и сцен
  - сенсомоторная координация действий, выполняемых эффекторами интеллектуальной компьютерной системы
- мультимодальный характер интерфейса — многообразие внешних языков, видов сенсоров и эффекторов
- формальное онтологическое описание на языке внутреннего смыслового представления информации
  - синтаксиса и денотационной семантики всех используемых внешних языков
  - первичных образов и сцен (ситуаций), являющихся результатом первичного анализа приобретаемой сенсорной информации
  - методов низкого уровня, непосредственно интерпретируемых эффекторами интеллектуальной компьютерной системы

Разговоры о дружественном и, в частности, адаптивном *пользовательском интерфейсе* ведутся давно, но это, чаще всего, касается формы ("синтаксической" стороны) *пользовательского интерфейса*, а не смыслового содержания взаимодействия с пользователями. В настоящее время *пользовательские интерфейсы* компьютерных систем (в том числе и *интеллектуальных компьютерных систем*) для широкого контингента пользователей не являются семантически (содержательно) дружественными (семантически комфортными). Организация взаимодействия пользователей с компьютерными системами (в том числе и с *интеллектуальными компьютерными системами*) является "узким местом", оказывающим существенное влияние на эффективность *автоматизации человеческой деятельности*. В основе современной организации взаимодействия пользователя с компьютерной системой лежит парадигма грамотного пользователя, который знает, чего он хочет от используемого им инструмента и несет полную ответственность за качество взаимодействия с этим инструментом. Эта парадигма лежит в основе деятельности лесоруба во взаимодействии с топором, всадника во взаимодействии с лошадью, автоводителя, летчика во взаимодействии с соответствующим транспортным средством, оператора атомной электростанции, железнодорожного диспетчера и так далее.

На современном этапе развития *Искусственного интеллекта* для повышения эффективности взаимодействия необходим переход от парадигмы грамотного управления используемым инструментом к парадигме равноправного сотрудничества, партнерскому взаимодействию *интеллектуальной компьютерной системы* со своим пользователем. *Интеллектуальная компьютерная система* должна повернуться "лицом" к пользователю.

Семантическая дружественность пользовательского интерфейса должна заключаться в адаптивности к особенностям и квалификации пользователя, исключении любых проблем для пользователя в процессе диалога с *интеллектуальной компьютерной системой*, в перманентной заботе о совершенствовании коммуникационных навыков пользователя.

При организации взаимодействия пользователя с *Глобальной сетью* компьютерным системам необходимо перейти от парадигмы "многооконного" интерфейса, в каждом "окне" которого свои "правила игры", к парадигме "одного окна". Пользователь не должен знать, какое "окно" ему надо "открыть" (в какую систему ему надо войти) для удовлетворения той или иной его потребности.

Пользователь не должен знать, какая конкретно система будет решать его задачу. Пользователь должен уметь с помощью универсальных средств сформулировать свою задачу, а соответствующая компьютерная система, входящая в *Глобальную сеть* и способная решить эту задачу, должна сама инициироваться, реагируя на факт появления указанной задачи. Таким образом пользовательский интерфейс должен быть интерфейсом пользователя не с конкретной компьютерной системой, а в целом со всей *Глобальной сетью компьютерных систем*.

Заключение к Главе 1.2. Достоинства предлагаемых принципов, лежащих в основе интеллектуальных компьютерных систем нового поколения

смысловое представление информации в памяти *интеллектуальных компьютерных систем* обеспечивает устранение дублирования информации, хранимой в памяти *интеллектуальной компьютерной системы*, то есть устранение многообразия форм представления одной и той же информации, запрещение появления в одной памяти *семантически эквивалентных информационных конструкций* и, в том числе, синонимичных *знаков*. Это существенно снижает сложность и повышает качество:

- разработки различных *моделей обработки знаний* (так как нет необходимости учитывать многообразие форм представления одного и того же знания);
- *семантического анализа* и *понимания* информации, поступающей (передаваемой) от различных внешних субъектов (от пользователей, от разработчиков, от других *интеллектуальных компьютерных систем*);
- *конвергенции* и *интеграции* различных видов знаний в рамках каждой *интеллектуальной компьютерной системы*;
- обеспечения *семантической совместимости* и *взаимопонимания* между различными *интеллектуальными компьютерными системами*, а также между *интеллектуальными компьютерными системами* и их пользователями.

Понятие *семантической сети* нами рассматривается не как красивая метафора сложноструктурированных *знаковых конструкций*, а как формальное уточнение понятия *смыслового представления информации*, как принцип представления информации, лежащей в основе нового поколения *компьютерных языков* и самих *компьютерных систем — графовых языков* и *графовых компьютеров*. *Семантическая сеть* — это нелинейная (графовая) *знаковая конструкция*, обладающая следующими свойствами:

- все элементы (то есть синтаксически элементарные фрагменты) этой *графовой структуры* (узлы и связки) являются *знаками* описываемых сущностей и, в частности, *знаками связей* между этими сущностями;
- все *знаки*, входящие в эту *графовую структуру*, не имеют *синонимов* в рамках этой структуры;
- "внутреннюю" структуру (строение) *знаков*, входящих в *семантическую сеть* не требуется учитывать при ее *семантическом анализе* (понимании);
- смысл *семантической сети* определяется *денотационной семантикой* всех входящих в нее *знаков* и конфигурацией *связей инцидентности* этих знаков;
- из двух *инцидентных знаков*, входящих в *семантическую сеть*, по крайней мере один является знаком связи.

рафинированная семантическая сеть — это *семантическая сеть*, имеющая максимально простую *синтаксическую структуру*, в которой, в частности,

- используется конечный *алфавит* элементов *семантической сети*, то есть конечное число синтаксически выделяемых типов (синтаксических меток), приписываемых этим элемента;
- внешние идентификаторы (в частности, имена), приписываемые элементам *семантической сети* используются только для ввода/вывода информации.

агентно-ориентированная модель обработки информации в сочетании с *децентрализованным ситуационным управлением процессом обработки информации*, а также со *смысловым представлением информации* в памяти *интеллектуальной компьютерной системы* существенно снижает сложность и повышает качество интеграции

    - обеспечивает автоматизацию решения сложных комплексных задач, для которых требуется создание временных или постоянных коллективов;
    - превращает *интеллектуальные компьютерные системы* в самостоятельные активные *субъекты*, способные инициировать различные комплексные задачи и, собственно, инициировать для этого работоспособные коллективы, состоящие из людей и *интероперабельных интеллектуальных компьютерных систем* требуемой квалификации.

- коллективы, состоящие из самостоятельных *интероперабельных интеллектуальных компьютерных систем* и людей, имеют хорошие перспективы стать *синергетическими* системами.

- *интероперабельность интеллектуальных компьютерных систем* обеспечивается:
    - высоким уровнем взаимопонимания и, соответственно, семантической совместимостью;
    - высоким уровнем договороспособности, то есть способности предварительно согласовывать свои действия с действиями других субъектов;
    - высоким уровнем способности оперативно координировать свои действия с действиями других субъектов в ходе их выполнения (см. *Главу 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*).

- к числу принципов, лежащих в основе построения *интероперабельных интеллектуальных компьютерных систем*, относятся:
    - смысловое представление знаний в памяти *интеллектуальных компьютерных систем* в виде рафинированных семантических сетей;
    - использование универсального языка внутреннего смыслового представления знаний (см. *Главу 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*);
    - графодинамическая организация обработки знаний;
    - агентно-ориентированные модели решения задач (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*);
    - структуризация и стратификация баз знаний в виде иерархической системы формальных онтологий (см. *Главу 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*);
    - семантически дружественный пользовательский интерфейс (см. *Главу 4.1. Общие принципы организации интерфейсов ostis-систем*).

- для разработки большого количества интероперабельных семантически совместимых *интеллектуальных компьютерных систем*, обеспечивающих переход на принципиально новый уровень автоматизации *человеческой деятельности*, необходимо создание технологии, обеспечивающей массовое производство таких *интеллектуальных компьютерных систем*, участие в котором доступно широкому контингенту разработчиков (в том числе разработчиков средней квалификации и начинающих разработчиков). Основными положениями такой технологии являются
    - стандартизация интероперабельных *интеллектуальных компьютерных систем*;
    - широкое использование *компонентного проектирования* на основе мощной библиотеки семантически совместимых многократно используемых (типовых) компонентов *интероперабельных интеллектуальных компьютерных систем* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*).

- эффективная эксплуатация *интероперабельных интеллектуальных компьютерных систем* требует создания не только *технологии проектирования* таких систем, но также и семейства технологий поддержки всех остальных этапов их жизненного цикла. Особенно это касается технологии перманентной поддержки *семантической совместимости всех* взаимодействующих *интероперабельных интеллектуальных компьютерных систем* в ходе их эксплуатации (см. *Главу 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*).

Глава 1.3.

Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения

⇒ *автор**:
- *Голенков В. В.*
- *Шункевич Д. В.*
- *Ковалев М. В.*
- *Садовский М. Е.*

⇒ *аннотация**:
[В главе рассмотрены принципы построения комплексной технологии разработки и поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения — *Технологии OSTIS*.]

⇒ *подраздел**:
- *§ 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)*
- *§ 1.3.2. Семантически совместимые ostis-системы*

⇒ *ключевой знак**:
- *Технология OSTIS*
- *Стандарт ostis-систем*
- *Метасистема OSTIS*
- *Стандарт OSTIS*
- *Экосистема OSTIS*

⇒ *ключевое понятие**:
- *ostis-система*

⇒ *ключевое знание**:
- *Обобщенный жизненный цикл ostis-систем*
- *Принципы, лежащие в основе Технологии OSTIS*

⇒ *библиографическая ссылка**:
- *Голенков В.В..СтандОТОП-2021кн*

§ 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)

жизненный цикл интеллектуальной компьютерной системы нового поколения

⇒ *включает в себя**:
- проектирование интеллектуальной компьютерной системы нового поколения
  ⇒ *включает в себя**:
  - проектирование базы знаний интеллектуальной компьютерной системы нового поколения
  - проектирование решателя задач интеллектуальной компьютерной системы нового поколения
  - проектирование интерфейса интеллектуальной компьютерной системы нового поколения
- реализацию интеллектуальной компьютерной системы нового поколения
- начальное обучение интеллектуальной компьютерной системы нового поколения
- мониторинг качества интеллектуальной компьютерной системы нового поколения
- поддержка требуемого уровня интеллектуальной компьютерной системы нового поколения
- реинжиниринг интеллектуальной компьютерной системы нового поколения
- обеспечение безопасности интеллектуальной компьютерной системы нового поколения
- эксплуатация интеллектуальной компьютерной системы нового поколения

Построение *Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* предполагает:
- четкое описание текущей версии *стандарта интеллектуальных компьютерных систем нового поколения*, обеспечивающего семантическую совместимость разрабатываемых систем;
- создание мощных библиотек семантически совместимых и многократно (повторно) используемых компонентов разрабатываемых *интеллектуальных компьютерных систем*;

- уточнение требований, предъявляемых к создаваемой комплексной технологии и обусловленных особенностями *интеллектуальных компьютерных систем нового поколения*, разрабатываемых и эксплуатируемых с помощью указанной технологии.

Создание инфраструктуры, обеспечивающей интенсивное перманентное развитие *Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* предполагает:

- обеспечение низкого порога вхождения в *технологию проектирования интеллектуальных компьютерных систем* как для пользователей технологии (то есть разработчиков прикладных или специализированных интеллектуальных компьютерных систем), так и для разработчиков самой технологии;
- обеспечение высоких темпов развития *технологии* за счет учета опыта разработки различных приложений путем активного привлечения авторов приложений к участию в развитии (совершенствовании) *технологии*.

В основе создания предлагаемой нами ***технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*** лежат следующие положения:

- реализация предлагаемой *технологии* разработки и сопровождения *интеллектуальных компьютерных систем нового поколения* в виде ***интеллектуальной компьютерной метасистемы***, которая полностью соответствует *стандартам* предлагаемых *интеллектуальных компьютерных систем нового поколения*, разрабатываемым по предлагаемой *технологии*. В состав такой *интеллектуальной компьютерной метасистемы*, реализующей предлагаемую технологию входит:
  - формальное онтологическое описание текущей версии *стандарта интеллектуальных компьютерных систем нового поколения*;
  - формальное онтологическое описание текущей версии *методов и средств проектирования, реализации, сопровождения, реинжиниринга и эксплуатации интеллектуальных компьютерных систем нового поколения*.

  Благодаря этому технология проектирования и реинжиниринга интеллектуальных компьютерных систем нового поколения и технология проектирования и реинжиниринга самой указанной технологии (то есть интеллектуальной компьютерной метасистемы) суть одно и тоже;

- ***унификация*** и ***стандартизация*** **интеллектуальных компьютерных систем нового поколения**, а также *методов* их *проектирования, реализации, сопровождения, реинжиниринга и эксплуатации*;
- перманентная эволюция ***стандарта интеллектуальных компьютерных систем нового поколения***, а также *методов* их *проектирования, реализации, сопровождения, реинжиниринга и эксплуатации;*
- ***онтологическое проектирование*** **интеллектуальных компьютерных систем нового поколения**, предполагающее:
  - четкое согласование и оперативную формализованную фиксацию (в виде *формальных онтологий*) утвержденного *текущего состояния* иерархической системы всех *понятий*, лежащих в основе перманентно эволюционируемого *стандарта интеллектуальных компьютерных систем нового поколения*, а также в основе каждой разрабатываемой *интеллектуальной компьютерной системы*;
  - достаточно полное и оперативное документирование текущего состояния каждого проекта;
  - использование *методики проектирования "сверху-вниз"*.
- ***компонентное проектирование*** **интеллектуальных компьютерных систем нового поколения**, то есть проектирование, ориентированное на сборку *интеллектуальных компьютерных систем* из готовых компонентов на основе постоянно расширяемых библиотек *многократно используемых компонентов*;
- ***комплексный характер*** предлагаемой *технологии*, осуществляющей:
  - поддержку *проектирования* не только *компонентов интеллектуальных компьютерных систем нового поколения* (различных *фрагментов баз знаний, баз знаний* в целом, различных *методов решения задач*, различных *внутренних информационных агентов, решателей задач* в целом, формальных онтологических описаний различных *внешних языков*, *интерфейсов* в целом), но также и *интеллектуальных компьютерных систем* в целом как самостоятельных *объектов проектирования* с учетом специфики тех классов, которым принадлежат проектируемые *интеллектуальных компьютерных системы*;
  - поддержку не только *комплексного проектирования интеллектуальных компьютерных систем нового поколения*, но также и поддержку их реализации (сборки, воспроизводства), сопровождения, реинжиниринга в ходе эксплуатации и непосредственно самой эксплуатации.

Для создания *технологии* комплексного проектирования и комплексной поддержки последующих этапов жизненного цикла *интеллектуальных компьютерных систем нового поколения* необходимо:

- Унифицировать формализацию различных моделей представления различного вида используемой информации, хранимой в памяти *интеллектуальных компьютерных систем* и различных моделей решения интеллектуальных задач для обеспечения *семантической совместимости* и простой автоматизируемой интегрируемости различных видов *знаний* и *моделей решения задач* в *интеллектуальных компьютерных системах*. Для этого необходимо разработать базовую *универсальную* абстрактную модель представления и обработки знаний, обеспечивающую реализацию всевозможных моделей решения задач.

- Унифицировать структуризацию *баз знаний* интеллектуальных компьютерных систем в виде иерархической системы онтологий разного уровня (см. *Главу 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*).
- Унифицировать систему используемых *понятий*, специфицируемых соответствующими *онтологиями* для обеспечения *семантической совместимости* и *интероперабельности* различных *интеллектуальных компьютерных систем* (см. *Главу 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*).
- Унифицировать архитектуру *интеллектуальных компьютерных систем*,
  - обеспечивающую *семантическую совместимость*:
    - между *интеллектуальными компьютерными системами* и их пользователями
    - между *индивидуальными интеллектуальными компьютерными системами*;
    - между *коллективными интеллектуальными компьютерными системами*,
  - а также обеспечивающую *интероперабельность* сообществ, состоящих из:
    - *индивидуальных интеллектуальных компьютерных систем*;
    - *коллективных интеллектуальных компьютерных систем*;
    - пользователей интеллектуальных компьютерных систем.

- Разработать *базовую модель интерпретации* всевозможных формальных моделей решения задач в интеллектуальных компьютерных системах с ориентацией на максимально возможное упрощение такой интерпретации в *компьютерах нового поколения*, которые специально предназначены для реализации индивидуальных *интеллектуальных компьютерных систем*.
- Разработать *компьютеры нового поколения*, принципы функционирования которых максимально близки к базовой абстрактной модели, обеспечивающей интеграцию всевозможных моделей представления знаний и моделей решения задач. При этом базовая машина обработки информации, лежащая в основе указанных компьютеров, должна существенно отличаться от фон-Неймановской машины и должна быть близка к базовой модели решения задач в интеллектуальных компьютерных системах для того, чтобы существенно снизить сложность интерпретации всего многообразия моделей решения задач в интеллектуальных компьютерных системах (см. *Главу 6.2. Ассоциативные семантические компьютеры для ostis-систем*).

Реализация всех перечисленных этапов развития *технологий Искусственного интеллекта* представляет собой переход на принципиально новый технологический уклад, обеспечивающий существенное повышение эффективности практического использования результатов работ в области *Искусственного интеллекта* и существенное повышение уровня автоматизации *человеческой деятельности.*

Предложенную нами *технологию комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения* мы назвали ***Технологией OSTIS*** (Open Semantic Technology for Intelligent Systems). Соответственно этому *интеллектуальные компьютерные системы нового поколения*, разрабатываемые по этой технологии называются ***ostis-системами***. Сама *Технология OSTIS* реализуется нами в форме специальной *ostis-системы*, которая названа нами ***Метасистемой OSTIS*** и *база знаний* которой содержит:

- Формальную теорию *ostis-систем*;
- Стандарт *ostis-систем*
  - Стандарт баз знаний *ostis-систем*
    - Стандарт внутреннего универсального языка смыслового представления знаний в памяти *ostis-систем*
    - Стандарт внутреннего представления онтологий верхнего уровня в памяти *ostis-систем*
    - Стандарт представления исходных текстов баз знаний *ostis-систем*
  - Стандарт решателей задач *ostis-систем*
    - Стандарт базового языка программирования *ostis-систем*
    - Стандарт языков программирования высокого уровня для *ostis-систем*
    - Стандарт представления искусственных нейронных сетей в памяти *ostis-систем*
    - Стандарт внутренних информационных агентов в *ostis-систем*
  - Стандарт интерфейсов *ostis-систем*
    - Стандарт внешних языков *ostis-систем*, близких к внутреннему универсальному языку смыслового представления знаний

- Стандарт методик и средств поддержки жизненного цикла *ostis-систем* (см. *Голенков В.В..СтандОТОП-2021кн*):
  - Ядро Библиотеки многократно используемых компонентов *ostis-систем* (***Библиотеки OSTIS***);
  - Методики *поддержки жизненного цикла ostis-систем* и их компонентов;
  - Инструментальные средства поддержки жизненного цикла *ostis-систем*.

Технология OSTIS
:= [Open Semantic Technology for Intelligent Systems]
:= [Открытая семантическая технология комплексной поддержки жизненного цикла ***семантически совместимых*** интеллектуальных компьютерных систем нового поколения]
:= [Модели, методики, методы и средства комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения]
:= [Теория интеллектуальных компьютерных систем нового поколения и практика компьютерной поддержки их жизненного цикла]
:= [Технологический комплекс (моделей, методик, автоматизированных методов и средств), соответствующий интеллектуальным компьютерным системам нового поколения (интероперабельным и семантически совместимым компьютерным системам)]
:= [Предлагаемая нами комплексная технология поддержки всех этапов жизненного цикла всех компонентов для всех классов (видов) интеллектуальных компьютерных систем нового поколения при перманентной поддержке их семантической совместимости]
⇒ *принципы, лежащие в основе**:
- комплексный характер технологии, заключающийся в том, что осуществляется поддержка:
  - всех этапов жизненного цикла создаваемых продуктов
  - для всех компонентов интеллектуальных компьютерных систем нового поколения
  - для всех классов интеллектуальных компьютерных систем нового поколения
- обеспечивается перманентная поддержка семантической совместимости между всеми создаваемыми интеллектуальными компьютерными системами нового поколения
- ориентация на комплексную автоматизацию всего многообразия человеческой деятельности
- реализация технологии и, соответственно, комплексная автоматизация поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения (со всеми их компонентами и классами) осуществляется в виде семейства интеллектуальных компьютерных систем нового поколения, построенных по той же технологии

Стандарт OSTIS
:= [Стандарт Технологии OSTIS]
:= [Основная часть базы знаний Метасистемы OSTIS]
⇒ *включает в себя**:
- *Стандарт ostis-систем*
- *Стандарт методик и средств поддержки жизненного цикла ostis-систем*

§ 1.3.2. Семантически совместимые ostis-системы

база знаний ostis-системы
:= [sc-конструкция, которая в текущий момент времени хранится в памяти ostis-системы]
⊃ *база знаний индивидуальной ostis-системы*
  ⊃ *база знаний корпоративной ostis-системы*
⊃ *распределенная база знаний коллектива ostis-систем*
  ∋ *База знаний Экосистемы OSTIS*
⇒ *примечание**:
[Каждый *sc-элемент* (знак, хранимый в базе знаний ostis-системы) по отношению к базе знаний *ostis-системы* считается временной сущностью, поскольку каждый *sc-элемент* в какой-то момент вводится в состав *базы знаний* и в какой-то момент может быть из нее удален, но при этом следует отличать временный характер самого *sc-элемента* от временного или постоянного характера обозначаемой им сущности]

Полная документация *ostis-системы*, включающая в себя и руководство пользователя, и руководство разработчика, является неотъемлемой частью базы знаний каждой *ostis-системы* и, соответственно, обеспечивает всестороннюю информационную поддержку деятельности пользователя, а также обеспечивает персонифицированную адаптацию к каждому пользователю и повышение уровня его квалификации по использованию этой интеллектуальной компьютерной системы.

ostis-система
:= [интеллектуальная компьютерная система нового поколения, построенная по *Технологии OSTIS*]
:= [предлагаемое нами уточнение понятия интеллектуальной компьютерной системы нового поколения]
⇒ *разбиение**:

{• *ostis-субъект*
:= [самостоятельная *ostis-система*]
:= [интероперабельная ostis-система]
⇒ *разбиение**:
{• *индивидуальная ostis-система*
• *коллективная ostis-система*
}
• *встроенная ostis-система*
:= [*ostis-система*, являющаяся частью некоторой *индивидуальной ostis-системы*]
}

интеллектуальная компьютерная система
⊃ *интероперабельная интеллектуальная компьютерная система*
:= [интеллектуальная компьютерная система нового поколения]
⊃ *ostis-субъект*
:= [предлагаемый нами вариант построения интероперабельных интеллектуальных компьютерных систем]

индивидуальная ostis-система
:= [минимальная самостоятельная *ostis-система*]
⇒ *разбиение**:
{• *персональный ostis-ассистент*
:= [*ostis-система*, осуществляющая комплексное адаптивное обслуживание конкретного пользователя по *всем* вопросам, касающимся его взаимодействия с любыми другими *ostis-системами*, а также представляющая интересы этого пользователя во всей глобальной сети *ostis-систем*]
• *корпоративная ostis-система*
:= [*ostis-система*, осуществляющая координацию совместной деятельности *ostis-систем* в рамках соответствующего коллектива *ostis-систем*, осуществляющая мониторинг и реинжиниринг соответствующего множества *ostis-систем* и представляющая интересы этого коллектива в рамках других коллективов *ostis-систем*]
• *индивидуальная ostis-система, не являющаяся ни персональным ostis-ассистентом, ни корпоративной ostis-системой*
}

коллективная ostis-система
:= [многоагентная система, представляющая собой коллектив индивидуальных и коллективных *ostis-систем*, деятельность которого координируется соответствующей корпоративной *ostis-системой*]
⇒ *примечание**:
[В состав коллектива *ostis-систем* могут входить индивидуальные *ostis-системы* могут входить индивидуальные *ostis-системы* любого вида — в том числе, корпоративные *ostis-системы*, представляющие интересы других коллективов *ostis-систем*]

Для *Технологии OSTIS* в рамках множества всевозможных *ostis-системы* особое место занимают следующие *ostis-системы*:

• *Метасистема OSTIS (см. Главу 7.2. Метасистема OSTIS)*
:= [Индивидуальная ostis-система, являющаяся реализацией Ядра Технологии OSTIS]
:= [Интеллектуальная компьютерная система нового поколения, построенная по *Технологии OSTIS* и обеспечивающая автоматизацию компьютерную поддержку жизненного цикла интеллектуальной компьютерной системы нового поколения, создаваемых также по *Технологии OSTIS*]
∈ *ostis-система*
⇐ *предлагаемая форма реализации**:
Технология OSTIS

• *Экосистема OSTIS (см. Главу 7.3. Структура Экосистемы OSTIS)*
:= [Коллективная ostis-система, представляющая собой глобальный коллектив всех *ostis-систем*, взаимодействующих между собой и осуществляющих комплексную автоматизацию человеческой деятельности]
:= [Глобальная сеть взаимодействующих *ostis-систем*, ориентированная на перманентно расширяемую комплексную автоматизацию самых различных видов и областей человеческой деятельности]
⇒ *примечание**:
[Это основной продукт *Технологии OSTIS*, который можно рассматривать как предлагаемый нами подход к реализации *Общества 5.0*, *Науки 5.0*, *Индустрии 5.0*]

Заключение к Главе 1.3.

Для разработки большого количества интероперабельных семантически совместимых *интеллектуальных компьютерных систем*, обеспечивающих переход на принципиально новый уровень автоматизации *человеческой деятельности* необходимо создание технологии, обеспечивающей массовое производство таких *интеллектуальных компьютерных систем*, участие в котором должно быть доступно широкому контингенту разработчиков (в том числе разработчиков средней квалификации и начинающих разработчиков). Основными положениями такой технологии являются:

- стандартизация интероперабельных *интеллектуальных компьютерных систем*;
- широкое использование *компонентного проектирования* на основе мощной библиотеки семантически совместимых многократно используемых (типовых) компонентов *интероперабельных интеллектуальных компьютерных систем* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*).

Эффективная эксплуатация *интероперабельных интеллектуальных компьютерных систем* требует создания не только *технологии проектирования* таких систем, но также и семейства технологий автоматизации всех остальных этапов их жизненного цикла. Особенно это касается технологии перманентной поддержки *семантической совместимости* всех взаимодействующих *интероперабельных интеллектуальных компьютерных систем*, а также технологии перманентной эволюции (модернизации) этих систем в ходе их эксплуатации.

Часть 2.

Смысловое представление и онтологическая систематизация знаний в интеллектуальных компьютерных системах нового поколения

⇒ *аннотация**:

[В части рассмотрено описание видов языков и информационных конструкций в рамках соответствующей онтологии, уточнение модели смыслового представления знаний, использующей понятие смыслового пространства, способы внутреннего и внешнего представления информации ostis-систем, онтологическая систематизация фактографической и логической информации, средства описания структуры баз знаний, а также средства описания (спецификации) денотационной семантики и синтаксиса естественных языков в ostis-системах. Данная часть имеет важное значение для обеспечения семантической совместимости ostis-систем.**]**

⇒ *подраздел**:

- *Глава 2.1. Информационные конструкции и языки*
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Глава 2.1.
Информационные конструкции и языки

⇒ *автор**:
- *Никифоров С.А.*
- *Гойло А.А.*

⇒ *подраздел**:
- *§ 2.1.1. Формализация понятия информационной конструкции*
- *§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*

⇒ *ключевое понятие**:
- *знак*
- *знаковая конструкция*
- *информационная конструкция*
- *дискретная информационная конструкция*
- *алфавит*
- *язык*
- *язык ostis-системы*
- *смысловое представление информации*
- *синтаксис языка*
- *денотационная семантика языка*
- *операционная семантика языка*
- *идентификатор*
- *файл*
- *файл ostis-системы*

⇒ *ключевой класс параметров**:
- *параметр, заданный на множестве знаковых конструкций*
- *параметр, заданный на множестве дискретных информационных конструкций*
- *параметр, заданный на множестве дискретных информационных конструкций*
- *параметр, заданный на множестве дискретных информационных конструкций*
- *параметр, заданный на множестве языков*

⇒ *ключевой класс отношений**:
- *отношение, заданное на множестве знаков*
- *отношение, заданное на множестве знаковых конструкций*
- *отношение, заданное на множестве элементов дискретных информационных конструкций*
- *отношение, заданное на множестве дискретных информационных конструкций*
- *отношение, заданное на множестве языков*
- *отношение, заданное на естественно-языковых файлах*

⇒ *библиографическая ссылка**:
- *Goylo A..Means oFDoSaD-2022art*
- *Голенков В.В..ОнтолПГСС-2019ст*
- *Голенков В.В..СтрукСП-2014ст*
- *Голенков В.В..СтандОТОП-2021кн*
- *Golenkov V.V..Metho aTfECoC-2019art*

§ 2.1.1. Формализация понятия информационной конструкции

В начале определим понятия знака и знаковой конструкции. Ранее данные понятия рассматривались в работах (см. *Goylo A..Means oFDoSaD-2022art*).

знак

⇒ *разбиение**:
- {• *знак, являющийся элементом дискретной информационной конструкции*

- *знак, являющийся неатомарным фрагментом дискретной информационной конструкции*

}

Под ***знаком*** понимается фрагмент *информационной конструкции*, который условно представляет (изображает) некоторую описываемую сущность, которую называют ***денотатом знака***. При этом отсутствие *знака*, обозначающего некоторую сущность, не означает отсутствие самой этой сущности. Это означает только то, что мы даже не догадываемся о ее существовании и, следовательно, не приступили к ее исследованию.

Поскольку все *знаки* являются *дискретными информационными конструкциями*, множество *знаков* является областью задания всех *отношений, заданных на множестве дискретных информационных конструкций*. Тем не менее есть как минимум одно *отношение*, специфичное для множества *знаков*.

отношение, заданное на множестве знаков
∋ *синонимия знаков**

знаки являются ***синонимичными*** в том и только в том случае, если они обозначают одну и ту же сущность. При этом *синонимичные знаки* могут быть *синтаксически эквивалентными*, а могут и не быть таковыми.

знаковая конструкция
⊂ *дискретная информационная конструкция*

знаковая конструкция — *дискретная информационная конструкция* которая в общем случае представляет собой конфигурацию *знаков* и специальных фрагментов *информационной конструкции*, обеспечивающих структуризацию конфигурации *знаков* — различного вида ***разделителей*** и ***ограничителей***. Для некоторых *знаковых конструкций* и даже для некоторых *языков* необходимость в *разделителях* и *ограничителях* может отсутствовать.

отношение, заданное на множестве знаковых конструкций
∋ *знак**
∋ *разделитель знаковой конструкции**
∋ *разделители знаковой конструкции**
∋ *ограничитель знаковой конструкции**
∋ *ограничители знаковой конструкции**
∋ *семантическая смежность знаковых конструкций**
∋ *конкатенация знаковых конструкций**
≔ [декомпозиция заданной знаковой конструкции на последовательность знаковых конструкций*]

знак* — *бинарное ориентированное отношение*, связывающее *знаковую конструкцию* со множеством всех *знаков*, входящих в ее состав.

семантическая смежность знаковых конструкций* — *бинарное отношение*, связывающее *семантически смежные знаковые конструкции*. При этом *знаковые конструкции* $\boldsymbol{Ti}$ и $\boldsymbol{Tj}$ являются *смежными* в том и только в том случае, если существуют *синонимичные знаки* $\boldsymbol{Ti}$ и $\boldsymbol{Tj}$, один из которых входит в состав $\boldsymbol{Ti}$, а второй — в состав $\boldsymbol{Tj}$

класс знаковых конструкций^
∋ *семантически элементарная знаковая конструкция*
∋ *семантически связная знаковая конструкция*

семантически элементарная знаковая конструкция — *знаковая конструкция*, описывающая некоторую (одну) связь между некоторыми *знаками* сущностей.

семантически связная знаковая конструкция — *знаковая конструкция*, которую можно представить в виде конкатенации *семантически элементарных знаковых конструкций*, каждая из которых *семантически смежна* предшествующей и последующей *семантически элементарной знаковой конструкции*.

параметр, заданный на множестве знаковых конструкций^
∋ *семантическая связность знаковых конструкций^*
  ∋ *семантически связная знаковая конструкция*
  ∋ *семантически несвязная знаковая конструкция*
∋ *наличие разделителей и ограничителей^*
  ∋ *знаковая конструкция, содержащая разделители и/или ограничители*
  ∋ *знаковая конструкция без разделителей и ограничителей*

информационная конструкция — *знаковая конструкция* (структура), содержащая некоторые сведения о некоторых сущностях. Форма представления ("изображения", "материализации"), форма структуризации (синтаксическая структура), а также *смысл** (денотационная семантика) *информационных конструкций* могут быть самыми различными.

информационная конструкция
⇒ *разбиение**:
{• *sc-конструкция*
:= [информационная конструкция, принадлежащая внутреннему языку ostis-системы]
• *информационна конструкция, не являющаяся sc-конструкцией*
:= [инородная (внешняя) для sc-памяти информационная конструкция]
:= [информационная конструкция, представленная на внешнем языке ostis-систем]
⇒ *разбиение**:
{• *файл*
• *информационная конструкция. не являющаяся ни sc-конструкцией, ни файлом*
}
}

дискретная информационная конструкция — *информационная конструкция*, *смысл* которой задается:
- множеством элементов (синтаксически атомарных фрагментов) этой *информационной конструкции*,
- *алфавитом* этих элементов — семейством классов *синтаксически эквивалентных элементов информационной конструкции*,
- принадлежностью каждого элемента *информационной конструкции* соответствующему классу *синтаксически эквивалентных элементов информационной конструкции*,
- конфигурацией связей инцидентности между элементами *информационной конструкции*.

Следствием этого является то, что форма представления элементов *дискретной информационной конструкции* для анализа ее смысла не требует уточнения. Главным является:
- наличие простой процедуры выделения (сегментации) элементов *дискретной информационной конструкции*,
- наличие простой процедуры установления синтаксической эквивалентности разных элементов *дискретной информационной конструкции*,
- наличие простой процедуры установления принадлежности каждого элемента *дискретной информационной конструкции* соответствующему классу синтаксически эквивалентных элементов (то есть соответствующему элементу алфавита).

элементом дискретной информационной конструкции является синтаксически атомарный фрагмент (символ), входящий в состав *дискретной информационной конструкции*. Поскольку *дискретные информационные конструкции* могут иметь общие элементы (атомарные фрагменты) и даже некоторые из них могут быть частями других *информационных конструкций*, элемент *дискретной информационной конструкции* может входить в состав сразу нескольких *информационных конструкций*.

Далее перейдем к рассмотрению понятий ***синтаксиса информационной конструкции*** и ***денотационной семантики***.

синтаксис
⊃ *синтаксис информационной конструкции*
:= [описание структуры информационной конструкции, являющееся минимально достаточным для ее семантически эквивалентное смысловое представление]
⊃ *синтаксис языка*
:= [синтаксические правила языка]

денотационная семантика
⊃ *денотационная семантика информационной конструкции*
:= [отображение синтаксической структуры информацинной конструкции в ее семантически эквивалентное смысловое представление]
⊃ *денотационная семантика языка*
:= [объединение семантических окрестностей основных понятий заданного языка и конъюнкция кванторных высказываний, являющихся семантическими правилами заданного языка, то есть правилами, которым должны удовлетворять семантически правильные смысловые информационные конструкции, соответствующие (семантические эквивалентные) синтаксически правильным конструкциям (текстам) заданного языка]

понимание
⊂ *действие*
:= [трансляция информационной конструкции на язык смыслового представления знаний]
:= [процесс построения семантически эквивалентного данной информационной конструкции смыслового представления]
:= [описание структуры информационной конструкции, являющееся минимально достаточным для ее семантически эквивалентное смысловое представление]

семантический анализ
⊂ *действие*
:= [построение синтаксического описания]

трансляция
⊂ *действие*
:= [перевод]
:= [построение семантически эквивалентной информационной конструкции]

Далее рассмотрим отношения, заданные на множестве элементов *дискретных информационных конструкций*.

отношение, заданное на множестве элементов дискретных информационных конструкций
∋ *элемент дискретной информационной конструкции**
∋ *синтаксическая эквивалентность элементов дискретных информационных конструкций**
∋ *инцидентность элементов дискретных информационных конструкций**

элемент дискретной информационной конструкции* — *бинарное ориентированное отношение*, каждая пара которого связывает (1) *знак* некоторой *дискретной информационной конструкции* и (2) *знак* одного из элементов этой *дискретной информационной конструкции**.

синтаксическая эквивалентность элементов дискретных информационных конструкций* — *отношение*, связывающее *синтаксически эквивалентные элементы* (атомарные фрагменты) одной и той же или разных *дискретных информационных конструкций*, то есть элементы, принадлежащие одному и тому же классу *синтаксически эквивалентных элементов дискретных информационных конструкций**.

инцидентность элементов дискретных информационных конструкций* для *линейных информационных конструкций* — это последовательность *элементов* (символов), входящих в состав этих *информационных конструкций*. Для *дискретных информационных конструкций*, конфигурация которых имеет нелинейный характер, *отношение инцидентности* их элементов может быть разбито на несколько частных *отношений инцидентности*, каждое из которых является подмножеством объединенного *отношения инцидентности*. Например, для двухмерных дискретных *информационных конструкций* это (1) *инцидентность элементов информационных конструкций* "по горизонтали" и (2) *инцидентность элементов информационных конструкций* "по вертикали".

Далее рассмотрим отношения, заданные на множестве *дискретных информационных конструкций*.

отношение, заданное на множестве дискретных информационных конструкций
∋ *неэлементарный фрагмент дискретной информационной конструкции**
∋ *алфавит дискретной информационной конструкции**
∋ *первичная синтаксическая структура дискретной информационной конструкции**
∋ *синтаксическая эквивалентность дискретных информационных конструкций**
∋ *копия дискретной информационной конструкции**
∋ *семантическая эквивалентность дискретных информационных конструкций**
∋ *семантическое расширение дискретной информационной конструкции**
∋ *синтаксис информационной конструкции**
∋ *смысл**
∋ *операционная семантика информационной конструкции**

неэлементарный фрагмент дискретной информационной конструкции* — *бинарное ориентированное отношение*, связывающее заданную *дискретную информационную конструкцию* с *дискретной информационной конструкцией*, которая является подструктурой для нее, в состав которой входит (1) подмножество элементов заданной *информационной конструкции* и, соответственно, (2) подмножество пар инцидентности *элементов заданной информационной конструкции*.

алфавит дискретной информационной конструкции* — *бинарное отношение*, связывающее *дискретную информационную конструкцию* с семейством попарно непересекающихся *классов синтаксически эквивалентных элементов* заданной *дискретной информационной конструкции**

первичная синтаксическая структура дискретной информационной конструкции* — *бинарное ориентированное отношение*, связывающее *дискретную информационную конструкцию* с *графовой структурой*, которая полностью описывает ее конфигурацию и которая включает: (1) *знаки* всех тех классов *синтаксически эквивалентных элементов*, которым принадлежат элементы описываемой *дискретной информационной конструкции*, (2) *знаки* всех *элементов* (атомарных фрагментов) описываемой *информационной конструкции*, (3) пары, описывающие *инцидентность элементов* описываемой *информационной конструкции*, (4) пары, описывающие принадлежность *элементов* описываемой *информационной конструкции* соответствующим *классам синтаксически эквивалентных элементов* этой *информационной конструкции*.

синтаксическая эквивалентность дискретных информационных конструкций*: *дискретные информационные конструкции* Ti и Tj являются синтаксически эквивалентными в том и только в том случае, если между конструкцией Ti и конструкцией Tj существует изоморфизм, в рамках которого каждому элементу конструкции Ti соответствует *синтаксически эквивалентный элемент информационной конструкции* Tj, то есть элемент, принадлежащий тому же классу *синтаксически эквивалентных элементов дискретных информационных конструкций*. Обратное утверждение также является верным.

копия дискретной информационной конструкции* — *бинарное ориентированное отношение*, которое связывает *дискретную информационную конструкцию* с *дискретной информационной конструкцией*, которая не только *синтаксически эквивалентна* ей, но и содержит информацию о форме представления *элементов* данной копируемой *информационной конструкции**.

копия дискретной информационной конструкции*
$\subset$ *синтаксическая эквивалентность дискретных информационных конструкций**

семантическая эквивалентность дискретных информационных конструкций*: *информационная конструкция* Ti и *информационная конструкция* Tj являются семантически эквивалентными в том и только в том случае, если каждая сущность (в том числе, и каждая связь между сущностями), описываемая в *информационной конструкции* Ti описывается также и в *информационной конструкции* Tj. Обратное утверждение также является верным.

семантическое расширение дискретной информационной конструкции*: *информационная конструкция* Tj является *семантическим расширением дискретной информационной конструкции* Ti в том и только в том случае, если каждая сущность, описываемая в Ti, описывается также и в Tj, но обратное неверно.

синтаксис информационной конструкции* — описание того, из каких частей состоит заданная *информационная конструкция* и как эти части (фрагменты) связаны между собой.

смысл* (*денотационная семантика информационной конструкции**) — *бинарное ориентированное отношение*, каждая пара которого связывает некоторую *информационную конструкцию* с явным (формальным) представлением того, какие сущности описывает данная *информационная конструкция* и как эти сущности связаны между собой.

операционная семантика информационной конструкции* — *бинарное ориентированное отношение*, каждая пара которого связывает *знак* некоторой *информационной конструкции* со множеством правил ее трансформации — описанием того, на основании каких правил можно осуществлять действия по преобразования (обработке, трансформации) заданной *информационной конструкции*, оставляя ее в рамках класса синтаксически и семантически правильных *информационных конструкций*.

операционная семантика информационной конструкции*
$\Rightarrow$ *второй домен**:
операционная семантика информационной конструкции

Далее рассмотрим заданные на множестве дискретных *информационных конструкций* соответствия.

соответствие, заданное на множестве дискретных информационных конструкций
$\ni$ *соответствие между синтаксической структурой информационной конструкции и смыслом этой конструкции**
$\subset$ *соответствие**

соответствие, заданное на множестве дискретных информационных конструкций — множество ориентированных пар, первым компонентом которых является ориентированная пара, состоящая из (1) *знака синтаксической структуры* некоторой *информационной конструкции* и (2) *знака* смысловой структуры этой *конструкции*, а вторым компонентом которых является множество ориентированных пар, связывающих фрагменты синтаксической структуры заданной *информационной конструкции* (которые описывают либо структуру фрагментов заданной конструкции, либо связи между фрагментами этой конструкции) с теми фрагментами смысловой структуры за-

данной *информационной конструкции*, которые семантически эквивалентны либо синтаксически представленным фрагментам заданной *информационной конструкции*, либо синтаксически представленным связям между такими фрагментами.

параметр, заданный на множестве дискретных информационных конструкций^
- ∋ *размерность дискретных информационных конструкций^*
  - ≔ [типология дискретных *информационных конструкций*, определяемая их размерностью]
  - ∋ *линейная информационная конструкция*
  - ∋ *двухмерная информационная конструкция*
  - ∋ *трехмерная информационная конструкция*
  - ∋ *четырехмерная информационная конструкция*
  - ∋ *графовая информационная конструкция*

линейная информационная конструкция — *дискретная информационная конструкция*, каждый *элемент* которой может иметь не более двух инцидентных ему элементов (один слева, другой справа).

двухмерная информационная конструкция — *дискретная информационная конструкция*, каждый *элемент* которой может иметь не более четырех инцидентных ему элементов (слева-справа, сверху-снизу).

трехмерная информационная конструкция — *дискретная информационная конструкция*, каждый *элемент* которой может иметь не более шести инцидентных ему элементов (слева-справа, сверху-снизу, сзади-спереди).

четырехмерная информационная конструкция — *дискретная информационная конструкция*, каждый *элемент* которой может иметь не более восьми инцидентных ему элементов (например, слева-справа, сверху-снизу, сзади-спереди, раньше-позже).

графовая информационная конструкция — *дискретная информационная конструкция*, множество элементов которой разбивается на два подмножества — связки и узлы. При этом узлы могут иметь неограниченное количество инцидентных им связок. В некоторых *графовых информационных конструкциях* и связки могут иметь неограниченное количество инцидентных им других связок.

параметр, заданный на множестве дискретных информационных конструкций^
- ∋ *типология дискретных информационных конструкций, определяемая их носителем^*
  - ∋ *некомпьютерная форма представления дискретных информационных конструкций*
    - ⊃ *аудио-сообщение*
    - ⊃ *информационная конструкция, представленная на языке жестов*
    - ⊃ *информационная конструкция, представленная в письменной форме*
  - ∋ *файл*

Представление *информационных конструкций* в виде *файлов* ориентировано на представление *дискретных* (!) *информационных конструкций*. Поэтому "файловое" представление недискретных *информационных конструкций* (например, различного рода сигналов) предполагает "дискретизацию" таких конструкций, то есть преобразование их в дискретные. Так преобразуются аудио-сигналы (в частности, речевые сообщения), изображения, видео-сигналы и другие.

параметр, заданный на множестве дискретных информационных конструкций^
- ∋ *уровень унификации представления синтаксически эквивалентных элементов дискретных информационных конструкций^*
  - ∋ *дискретная информационная конструкция с низким уровнем унификации представления элементов*
    - ⊃ *аудио-сообщение*
    - ⊃ *информационная конструкция, представленная на языке жестов*
    - ⊃ *рукопись или ее копия*
  - ∋ *дискретная информационная конструкция с высоким уровнем унификации представления элементов*
    - ⊃ *печатный текст*
    - ⊃ *файл*

Уровень ***унификации представления синтаксически эквивалентных элементов дискретных информационных конструкций^*** — уровень "членораздельности" *дискретных информационных конструкций*.

Чем выше *уровень унификации представления элементов дискретных информационных конструкций*, тем проще реализуется:
- процедура выделения (сегментации) элементов *дискретной информационной конструкции*,
- процедура установления синтаксической эквивалентности этих элементов,
- процедура их распознавания, то есть процедура установления их принадлежности соответствующим классам синтаксически эквивалентных элементов.

Уточнив понятия *знака*, *знаковой конструкции*, *информационной конструкции*, *дискретной информационной конструкции* и рассмотрев соответствующие отношения, можно перейти к формализации понятия *язык*.

язык — класс *знаковых конструкций*, для которого существуют:
- общие правила их построения,
- общие правила их соотнесения с теми сущностями и конфигурациями сущностей, которые описываются (отражаются) указанными *знаковыми конструкциями*.

язык
⇒ *разбиение**:
{• *язык, у которого все знаки, входящие в состав его знаковых конструкций, являются элементарными фрагментами этих конструкций*
• *язык, у которого знаки, входящие в состав его знаковых конструкций, в общем случае не являются элементарными фрагментами этих конструкций*
⇒ *разбиение**:
{• *язык, знаковые конструкции которого содержат разделители и ограничители*
• *язык, знаковые конструкции которого не содержат разделителей и ограничителей*
}
}

Для *языков, у которого все знаки, входящие в состав его знаковых конструкций, являются элементарными фрагментами этих конструкций* существенно упрощаются методы обработки их текстов.

язык, знаковые конструкции которого не содержат разделителей и ограничителей — *язык*, *знаковые конструкции* такого языка состоят только из *знаков*.

отношение, заданное на множестве языков — *отношение*, область определения которого включает в себя множество всевозможных *языков*.

текст заданного языка* — *бинарное отношение*, связывающее *язык* и синтаксически правильную (правильно построенную) *знаковую конструкцию* данного языка. ***синтаксически корректная знаковая конструкция для заданного языка**** — *бинарное отношение*, связывающее *язык* и *знаковую конструкцию*, не содержащая синтаксических ошибок для данного *языка*.

отношение, заданное на множестве языков
:= **[**отношение, область определения которого включает в себя множество всевозможных языков**]**
∋ *текст заданного языка**
= (*синтаксически корректная знаковая конструкция для заданного языка** ∩ *синтаксически целостная знаковая конструкция для заданного языка**)
∋ *синтаксически корректная знаковая конструкция для заданного языка**
∋ *синтаксически целостная знаковая конструкция для заданного языка**
∋ *синтаксически неправильная знаковая конструкция для заданного языка**
= (*синтаксически некорректная знаковая конструкция для заданного языка** ∪ *синтаксически нецелостная знаковая конструкция для заданного языка**)
⊃ *синтаксически некорректная знаковая конструкция для заданного языка**
⊃ *синтаксически нецелостная знаковая конструкция для заданного языка**
∋ *знание, представленное на заданном языке**
:= **[**семантически правильный текст заданного языка***]**
= *(семантически корректный текст заданного языка** ∩ *семантически целостный текст заданного языка*)*
:= **[**истинный текст заданного языка***]**
:= **[**истинное высказывание, представленное на заданном языке***]**
∋ *семантически корректный текст заданного языка**
:= **[**текст заданного языка, не содержащий семантических ошибок, противоречащих признанным закономерностям и фактам***]**
∋ *семантически целостный текст заданного языка**
:= **[**текст заданного языка, содержащий достаточную информацию для установления его истинности***]**
∋ *семантически неправильный текст для заданного языка**
= *(семантически некорректный текст для заданного языка** ∪ *семантически нецелостный текст для заданного языка*)*
⊃ *семантически некорректный текст для заданного языка**
⊃ *семантически нецелостный текст для заданного языка**
∋ *алфавит**
:= **[**алфавит заданной информационной конструкции или заданного языка***]**

:= [семейство классов синтаксически эквивалентных элементов (элементарных фрагментов) заданной информационной конструкции или *информационных конструкций* заданного языка*]

∋ *семейство классов синтаксически эквивалентных разделителей**

:= [семейство классов синтаксически эквивалентных разделителей, входящих в состав заданной информационной конструкции или в состав *информационных конструкций* заданного языка*]

∋ *семейство классов синтаксически эквивалентных ограничителей**

:= [семейство классов синтаксически эквивалентных ограничителей, входящих в состав заданной информационной конструкции или в состав *информационных конструкций* заданного языка*]

∋ *синтаксис языка**

:= [быть теорией правильно построенных *информационных конструкций*, принадлежащих заданному языку*]

:= [определение понятия правильно построенной информационной конструкции заданного языка*]

:= [синтаксические правила заданного языка*]

:= [быть синтаксическими правилами заданного языка*]

:= [*бинарное ориентированное отношение*, каждая пара которого связывает *знак* некоторого *языка* с описанием синтаксически выделяемых классов фрагментов конструкций заданного языка, с описанием отношений, заданных на этих классах и с конъюнкцией кванторных высказываний, являющихся *синтаксическими правилами заданного языка*, то есть правилами, которым должны удовлетворять все *синтаксические правильные (правильно построенные) конструкции (тексты)* указанного языка*]

⇒ *примечание**:

[При представлении синтаксиса (синтаксических правил) заданного *языка* используются все те понятия, которые вводятся для представления синтаксических структур *информационных конструкций*, принадлежащих указанному *языку*. Это и синтаксически выделяемые классы фрагментов указанных *информационных конструкций*, и *отношения*, заданные на множестве таких фрагментов.]

⇒ *второй домен**:

синтаксис языка

∋ *описание синтаксических понятий языка**

:= [описание синтаксически выделяемых классов фрагментов конструкций заданного языка*]

⇒ *второй домен**:

описание синтаксических понятий языка

⇐ *обобщенное включение**:

синтаксис языка

∋ *синтаксические правила языка**

:= [синтаксические правила заданного языка*]

⇒ *второй домен**:

синтаксические правила языка

∋ *денотационная семантика языка**

:= [быть теорией морфизмов, связывающих правильно построенные *информационные конструкции* заданного языка с описываемыми конфигурациями описываемых сущностей*]

∋ *денотационная семантика языка**

:= [семантические правила заданного языка*]

:= [быть семантическими правилами заданного языка*]

:= [*бинарное ориентированное отношение*, каждая пара которого связывает *знак* некоторого *языка* с описанием основных семантических понятий заданного *языка* и конъюнкцией кванторных высказываний, являющихся *семантическими правилами* заданного *языка*, то есть правилами, которым должны удовлетворять семантически правильные смысловые *информационные конструкции*, соответствующие (*семантические эквивалентные*) синтаксически правильным конструкциям (текстам) заданного *языка**]

⇒ *примечание**:

[При формулировке семантических правил заданного *языка* используются понятия, введенные в рамках базовых онтологий (онтологий самого высокого уровня, в которых рассматриваются самые общие классы описываемых сущностей, самые общие *отношения* и *параметры*).]

⇒ *второй домен**:

денотационная семантика языка

∋ *описание семантических понятий языка**

⇒ *второй домен**:

описание семантических понятий языка

∋ *семантические правила языка**

⇒ *второй домен**:

семантические правила языка

∋ *семантическая эквивалентность языков**

:= [быть семантически эквивалентными языками*]

⇒ *определение**:
[*язык* $\boldsymbol{Li}$ и *язык* $\boldsymbol{Lj}$ будем считать *семантически эквивалентными языками** в том и только в том случае, если для каждого текста, принадлежащего $\boldsymbol{Li}$, существует *семантически эквивалентный текст**, принадлежащий $\boldsymbol{Lj}$, и наоборот.]

∋ *семантическое расширение языка**
⇔ *обратное отношение**:
*семантическое сужение языка**
⇒ *определение**:
[*язык* $\boldsymbol{Lj}$ будем считать *семантическим расширением* языка* $\boldsymbol{Li}$ в том и только в том случае, есть ли для каждого текста, принадлежащего $\boldsymbol{Li}$, существует *семантически эквивалентный текст**, принадлежащий $\boldsymbol{Lj}$, но обратное неверно.]

∋ *синтаксическое расширение языка**
≔ [быть семантически эквивалентным надмножеством заданного языка*]
⇒ *определение**:
[*язык* $\boldsymbol{Lj}$ будем считать *синтаксическим расширением* языка* $\boldsymbol{Li}$ в том и только в том случае, если:

- $\boldsymbol{L_j} \supset \boldsymbol{Li}$ (то есть все тексты языка $\boldsymbol{Li}$ являются также и текстами языка $\boldsymbol{Lj}$, но обратное неверно);
- *язык* $\boldsymbol{Lj}$ и *язык* $\boldsymbol{Li}$ являются *семантически эквивалентными языками**.

]

∋ *синтаксическое ядро языка**
≔ [быть синтаксическим ядром заданного языка*]
≔ [быть семантически эквивалентным подмножеством заданного *языка*, имеющим минимальную синтаксическую сложность*]

∋ *направление синтаксического расширения ядра заданного языка**
≔ [быть правилом трансформации *информационных конструкций*, принадлежащих заданному *языку*, которое описывает одно из направлений перехода от множества конструкций ядра этого языка ко множеству всех принадлежащих ему *информационных конструкций**]

∋ *операционная семантика языка**
≔ [*бинарное ориентированное отношение*, каждая пара которого связывает знак некоторого *языка* со множеством правил трансформации текстов этого языка*]
⇒ *второй домен**:
операционная семантика языка

∋ *внутренний язык**
≔ [быть внутренним *языком* для заданной системы, основанной на обработке информации, или заданного множества таких систем*]
≔ [быть *языком* внутреннего представления информации в памяти заданной системы, основанной на обработке информации или заданного класса таких систем*]

∋ *внешний язык**
≔ [быть внешним *языком* для заданной системы, основанной на обработке информации, или заданного множества таких систем*]
≔ [быть *языком*, используемым для обмена информацией заданной системы, основанной на обработке информации, или заданного множества таких систем с другими системами, основанными на обработке информации, (в том числе, и с себе подобными)*]

∋ *используемый язык**
= (*внутренний язык** ∪ *внешний язык**)
≔ [*язык*, используемый заданной системой, основанной на обработке информации или заданного множества таких систем*]
≔ [*язык*, носителем которого является (которым владеет) данная система, основанная на обработке информации]

параметр, заданный на множестве языков^ — семейство классов эквивалентности языков, трактуемой в контексте того или иного свойства (характеристики), присущего *языкам*.

параметр, заданный на множестве языков^

∋ *семантическая мощность языка^*
≔ [класс *языков*, семантически эквивалентных друг другу]
∋ *универсальный язык*
≔ [класс всевозможных универсальных языков]
⇒ *примечание**:

[Очевидно, что все универсальные *языки* (если они действительно таковыми являются, а не только претендуют на это) семантически эквивалентны друг другу, то есть имеют одинаковую семантическую мощность.]

∋ *уровень синтаксической сложности представления знаков в текстах языка*^
- ∋ *язык, в текстах которого все знаки представлены синтаксически элементарными фрагментами*
- ∋ *язык, в текстах которого знаки в общем случае представлены синтаксически неэлементарными фрагментами*

∋ *использование разделителей и ограничителей в текстах языка*^
- ∋ *язык, в текстах которого не используются разделители и ограничители*
- ∋ *язык, в текстах которого используются разделители и ограничители*

∋ *уровень сложности процедуры установления синонимии знаков в текстах языка*^
- ∋ *язык, в рамках каждого текста которого синонимичные знаки отсутствуют*
  - ⇒ *пояснение**:
    [В текстах такого языка знак каждой описываемой сущности входит однократно.]
- ∋ *язык, в рамках которого синонимичные знаки представлены синтаксически эквивалентными фрагментами текстов*
- ∋ *флективный язык*
  - ≔ [язык, в рамках которого синонимичные знаки могут быть представлены синтаксически неэквивалентными фрагментами, но фрагментами, являющимися модификациями некоторого "ядра" этих фрагментов (при склонении и спряжении этих знаков).]
- ∋ *язык, в рамках которого синонимичные знаки в общем случае могут быть представлены синтаксически неэквивалентными текстовыми фрагментами, структура которых носит непредсказуемый характер*

∋ *наличие омонимии в текстах языка*^
- ∋ *язык, в текстах которого присутствует омонимия знаков*
  - ≔ [язык, в текстах которого присутствуют синтаксически эквивалентные, не синонимичные знаки]
- ∋ *язык, в текстах которого омонимия знаков отсутствует*

семантически выделяемый класс дискретных информационных конструкций

∋ *синтаксическая структура информационной конструкции*
- ⇐ *второй домен**:
  *синтаксис информационной конструкции**
- ⊃ *первичная синтаксическая структура информационной конструкции*
- ⊃ *вторичная синтаксическая структура информационной конструкции*

∋ *синтаксис языка*

∋ *описание синтаксических понятий языка*

∋ *синтаксические правила языка*

∋ *денотационная семантика языка*

∋ *описание семантических понятий языка*

∋ *семантические правила языка*

∋ *операционная семантика языка*

∋ *смысловое представление информации*
- ⇐ *второй домен**:
  *смысл**

смысловое представление информации — явное (формальное) представление описываемых сущностей и связей между ними (см. *Голенков В.В..ОнтолПГСС-2019ст*, *Голенков В.В..СтрукСП-2014ст*). Для явного представления описываемых сущностей и связей между ними требуется существенное упрощение синтаксической структуры *информационных конструкций* (см. *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*).

язык ostis-системы — *язык* представления *информационных конструкций* в ostis-системах.

язык ostis-системы

⊂ *формальный язык*

⊂ *универсальный язык*

⇐ *используемые языки**:
ostis-система

∋ *SC-код*
- ⇐ *внутренний язык**:
  ostis-система
- ∈ *универсальный язык*

Для формального описания различного рода *языков* (включая *SCg-код*, *SCs-код*, *SCn-код*, рассматриваемые в Главе 2.3.) используется целый ряд метаязыковых понятий.

Перечислим некоторые из них: *идентификатор*, *класс синтаксически эквивалентных идентификаторов*, *имя*, *простое имя*, *выражение*, *внешний идентификатор**, *алфавит**, *разделители**, *ограничители**, *предложения**

Синтаксис *языков представления знаний в ostis-системах* может быть формально описан различными способами. Так, например, можно использовать метаязык Бэкуса-Наура для описания синтаксиса *SCs-кода* или его расширение для описания синтаксиса *SCn-кода*. Однако значительно более логично и целесообразно описывать синтаксис всех форм внешнего отображения sc-текстов с помощью самого *SC-кода*. Такой подход позволит *ostis-системам* самостоятельно понимать, анализировать и генерировать тексты указанных *языков* на основе принципов, общих для любых форм внешнего представления информации, в том числе нелинейных.

алфавит* — *бинарное отношение*, связывающее множество текстов с семейством максимальных множеств синтаксически однотипных элементарных (атомарных) фрагментов текстов, принадлежащих заданному множеству текстов.

ограничители*
:= [*отношение*, связывающее заданный класс *информационных конструкций* с соответствующим классом их *ограничителей*]
:= [быть *ограничителями*, используемыми в заданном множестве *информационных конструкций**]

разделители*
:= [быть *разделителями*, используемыми в заданном множестве *информационных конструкций**]
⇒ *второй домен**:
разделитель

идентификатор — структурированный *знак* соответствующей (обозначаемой) сущности, который чаще всего представляет собой строку (цепочку символов), которую будем называть именем соответствующей сущности. В формальных текстах (в том числе текстах *SC-кода*, *SCg-кода*, *SCs-кода*, *SCn-кода*) основные используемые *идентификаторы* не должны быть омонимичными, то есть должны однозначно соответствовать идентифицируемым сущностям. Следовательно, каждая пара идентификаторов, имеющих одинаковую структуру, должны обозначать одну и ту же сущность.

имя
⊂ *идентификатор*
:= [строковый идентификатор]
:= [идентификатор, представляющий собой строку (цепочку) символов]
⇒ *декомпозиция действия**:
{• *простое имя*
:= [атомарное имя]
:= [имя, в состав которого другие имена не входят]
• *выражение*
:= [неатомарное имя]
}

внешний идентификатор* — *бинарное ориентированное отношение*, каждая связка (sc-дуга) которого связывает некоторый элемент с *файлом*, содержимым которого является *внешний идентификатор* (чаще всего, имя), соответствующий указанному элементу. Понятие внешнего идентификатора является понятием относительным и важным для ostis-систем, поскольку внутреннее для *ostis-систем* представление информации (в виде текстов *SC-кода*) оперирует не идентификаторами описываемых сущностей, а *знаками*, структура которых никакого значения не имеет.

предложения*
:= [быть множеством всех предложений заданного текста, не являющихся встроенными предложениями, то есть предложениями, входящими в состав других предложений*]
⇒ *второй домен**:
предложение

предложение
⇒ *пояснение**:
[минимальный семантически целостный фрагмент текста, представляющий собой конфигурацию *знаков*, входящих в этот фрагмент и связываемых между собой *отношениями инцидентности* (в частности, отношением

непосредственной последовательности в строке), а также различного вида *разделителями* и *ограничителями*]

§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем

Для представления *баз знаний ostis-систем* используется целый ряд как *универсальных языков*, так и *специализированных языков*, как *формальных языков*, так и *естественных языков*, как *внутренних языков*, обеспечивающих представление информации в памяти *ostis-систем*, так и *внешних языков*, обеспечивающих представление информации, вводимой в память *ostis-систем*, либо выводимой из этой памяти. *Естественные языки* используются исключительно для представления *файлов*, хранимых в памяти *ostis-системы* и формально специфицируемых в рамках *базы знаний* этой *ostis-системы*.

Для эксплуатации *интеллектуальных компьютерных систем*, построенных на основе *SC-кода* (см. *Главу 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*, а также *Голенков В.В..СтандОТОП-2021кн*, *Golenkov V.V..Metho aTfECoC-2019art*), кроме способа абстрактного внутреннего представления баз знаний (*SC-кода*) потребуются несколько способов внешнего изображения абстрактных *sc-текстов*, удобных для пользователей и используемых при оформлении исходных текстов *баз знаний* указанных интеллектуальных компьютерных систем и исходных текстов фрагментов этих *баз знаний*, а также используемых для отображения пользователям различных фрагментов *баз знаний* по пользовательским запросам. В качестве таких способов внешнего отображения *sc-текстов* и применяются рассмотренные в *Главе 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний* внешние языки ostis-систем (*SCg-код*, *SCs-код* и *SCn-код*).

SCg-код *— внешний язык* ostis-систем*, тексты которого представляют собой графовые структуры общего вида с точно заданной *денотационной семантикой**.

SCg-код
- ∈ *язык ostis-системы*
  - ≔ [Semantic Code graphical]
  - ⇐ *внешний язык**:
    ostis-система
  - ∈ *универсальный язык*

SCs-код *— внешний язык* ostis-систем*, тексты которого представляют собой строки (цепочки) символов.

SCs-код
- ∈ *язык ostis-системы*
  - ≔ [Semantic Code string]
  - ⇐ *внешний язык**:
    ostis-система
  - ∈ *универсальный язык*

SCn-код *— внешний язык* ostis-систем*, тексты которого представляют собой двухмерные матрицы символов, являющиеся результатом форматирования, двухмерной структуризации текстов *SCs-кода*.

SCn-код
- ∈ *язык ostis-системы*
  - ≔ [Semantic Code natural]
  - ⇐ *внешний язык**:
    ostis-система
  - ∈ *универсальный язык*

Все основные внешние *формальные языки*, используемые *ostis-системами* (*SCg-код*, *SCs-код*, *SCn-код*) являются различными вариантами внешнего представления текстов *внутреннего языка ostis-систем — SC-кода*. Указанные языки являются универсальными и, следовательно, *семантически эквивалентными языками**.

При этом, каждая *ostis-система* может приобрести способность использовать любой *внешний язык* (как универсальный, так и специализированный, как естественный, так и искусственный), если *синтаксис* и *денотационная семантика* этого *языка* будут описаны в памяти *ostis-системы* на ее *внутреннем языке* (*SC-коде*).

файл *— информационная конструкция*, представленная в "цифровой"форме, хранимой в какой-либо компьютерной памяти, но не являющийся *sc-конструкцией*, хранимой в памяти *ostis-системы*.

файл
⇒ *разбиение**:
{• *файл ostis-системы*
:= [файл, хранимый в памяти ostis-системы]
⇒ *разбиение**:
{• *сформированный файл ostis-системы*
• *несформированный файл ostis-системы*
}
• *файл компьютерной системы, не являющейся ostis-системой*
}
⊃ *текстовый файл*
⊃ *естественно-языковой файл*
⇒ *разбиение**:
{• *размеченный естественно-языковой файл*
• *неразмеченный естественно-языковой файл*
}
⊃ *русскоязычный естественно-языковой файл*
⊃ *англоязычный естественно-языковой файл*
⊃ *файл изображения*
⊃ *видео-файл*
⊃ *аудио-файл*

файл ostis-системы — инородная для *sc-кода информационная конструкция*, которая может как храниться в памяти *ostis-системы*, так и вне ее.

файл ostis-систем
⇒ *разбиение**:
{• *хранимый файл ostis-систем*
• *пустой файл ostis-систем*
}
⇒ *разбиение**:
{• *файл-экземпляр*
:= [обозначение одного из вхождений (однорго из экземпляров) информационной конструкции]]
• *файл-класс*
:= [обозначение всевозможнных *информационных конструкций*, каждая из которых эквивалентна той, что представлена содержимым данного sc-узла]
}
⊃ *sc.g-файл*
⇒ *примечание**:
[См. *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*.]
⊃ *sc.s-файл*
⇒ *примечание**:
[См. *§ 2.3.2. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*.]
⊃ *sc.n-файл*
⇒ *примечание**:
[См. *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*.]
⊃ *естественно-языковой файл ostis-системы*
⊂ *естественно-языковой файл*
:= [естественно-языковой файл, в котором выделены фрагменты, являющиеся sc-идентификаторами и, соответственно, ссылками на соответствующие им sc-элементы]

файл-образец
:= [файл-эталон]
:= [файл-оригинал]

файл-образец — *файл*, объявленный как оригинал в рамках *экоситсемы OSTIS*. ***файл-копия*** — *файл*, являющийся *копией* соответствующего образца и хранимый в одном месте. *файл-копия* может быть *копией* как и *файла-класса*, так и *файла-экземпляра*. ***файл-копия*** — *файл*, являющийся *копией* соответствующего образца и хранимый в одном месте. *файл-копия* может быть *копией* как и *файла-класса*, так и *файла-экземпляра*.

файл-копия

⇒ *разбиение**:

{• *файл-копия, образец которого хранится в памяти ostis-системы*

• *файл-копия, образец которого хранится в памяти компьютерной системы, которая ostis-системой не является*

}

файлом ostis-системы может стать любой *файл* современной компьютерной системы. Некоторые *текстовые файлы* (в первую очередь *естественно-языковые файлы* — тексты *естественных языков*) могут быть некоторым образом размечены путем указания связей размеченного *файла* с другими *файлами ostis-системы*, а также с различными *sc-элементами*. Так, например:

- в любом месте размеченного текста можно сделать ссылку на другой *файл*, трактуемый как примечание к данному месту размеченного текста,
- в размеченном тексте можно выделить используемые в нем *основные идентификаторы* и *неосновные идентификаторы* (курсивом),
- для выделенных *неосновных sc-идентификаторов* можно выделить соответствующие им *основные sc-идентификаторы*.

Если в естественно-языковом тексте (например в цитате) используется *неосновной sc-идентификатор*, то при разметке этого текста после указанного *sc-идентификатора* приводится соответствующий (синонимичный) основной *sc-идентификатор*. Если в естественно-языковом тексте приводится подряд несколько *основных sc-идентификаторов*, выделенных курсивом, то при разметке этого текста после указанных идентификаторов приводится перечень (через точку с запятой) соответствующих *sc-идентификаторов*.

В состав *текстового файла ostis-систем* могут входить выделенные курсивом *основные sc-идентификаторы*, являющиеся *внешними sc-идентификаторами* соответствующих *sc-элементов*. В частности, это могут быть *sc-идентификаторы*, обозначающие другие текстовые *файлы*, являющиеся примечаниями и(или) пояснениями к соответствующим местам заданного *файла*, а также библиографические ссылки. Такого рода *sc-идентификаторы* в исходном тексте *файла ограничителями* ссылок на *информационные конструкции* (квадратными скобками).

отношение, заданное на естественно-языковых файлах

∋ *ссылка**

:= [*бинарное ориентированное отношение*, любая пара которого связывает *естественно-языковой файл* с другим *файлом*, на который указанный *файл* ссылается. При этом (1) второй *файл* может быть не только *естественно-языковым файлом* (2) в тексте первого *файла* может быть явно указано место с которого ссылка осуществляется, (3) частным видом ссылки может быть *примечание*.]

∋ *ключевой знак**

∋ *семантическая смежность**

∋ *перевод**

∋ *авторская ссылка**

∋ *авторское примечание**

Глава 2.2.

Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)

⇒ *автор**:
- *Голенков В. В.*
- *Ивашенко В. П.*

⇒ *подраздел**:
- *§ 2.2.1. Базовая денотационная семантика SC-кода*
- *§ 2.2.2. Синтаксис SC-кода*
- *§ 2.2.3. Смысловое пространство ostis-систем*

⇒ *ключевое понятие**:
- *sc-элемент*
- *обозначение sc-множества*
- *обозначение sc-связки*
- *обозначение sc-класса*
- *обозначение sc-структуры*
- *обозначение внешней сущности*
- *sc-константа*
- *sc-переменная*
- *sc-множество*
- *sc-связка*
- *sc-класс*
- *sc-структура*
- *внешняя сущность*

⇒ *ключевое знание**:
- *Базовая денотационная семантика SC-кода*

⇒ *библиографическая ссылка**:
- *Нариньяни А.С.нФактоНиНРиВ-2000ст*
- *Ivashenko V.P.GenerPSRLaSS-2022art*
- *Ivashenko V.P.StrinPMfKDS-2020art*
- *Collatz L.FunctAaNM-1966bk*
- *Ивашенко В.П.ОнтолМПВОСиЯвПОЗ-2017ст*
- *Ивашенко В.П.Модел иАИЗнООСС-2014дс*
- *Bohm D.tUndivUaOIoQT-1993bk*
- *Bohm D.Whole atIO-2002bk*
- *Налимов В.В..РеальНВМБ-1995кн*
- *Налимов В.В.СпонтСВТСиСАЛ-1989кн*
- *Налимов В.В.ВероятМЯоСЕиИЯ-1979кн*
- *Мартынов В.В.Язык вПиВкПГС-2004кн*
- *Manin Y.I.SemanS-2016art*
- *Melcuk I.Langu fMtT-2016bk*
- *Harris J.AlgebGaFC-1992bk*
- *Alt H.Compu tFDbTPC-1995art*
- *Study E.KürzeWiKG-1905art*
- *Kostrikin A..LineaAaG-1997bk*
- *Lowe W.Towar aToSS-2001art*
- *Manin Y.I.ZipfsLaLLPD-2014art*
- *Мартынов В.В.вЦентрСЧ-2009кн*
- *Гордей А.Н.ТеориАПАЗ-2014ст*

Введение в Главу 2.2.

В *§ 2.2.1. Базовая денотационная семантика SC-кода* и в *§ 2.2.2. Синтаксис SC-кода* приведено формальное описание *Денотационной семантики и Синтаксиса SC-кода*. Сделано это будет следующим образом. Поскольку все элементы *информационных конструкций* являются обозначениями описываемых сущностей и, в том числе, обозначениями различных выделяемых классов *sc-элементов*, можно явно ввести различные семантически значимые и синтаксически выделяемые классы *sc-элементов* и на основе этого явно описать средствами *SC-кода Базовую денотационную семантику* и *Синтаксис SC-кода*.

Синтаксис SC-кода задается семейством классов синтаксический выделяемых *sc-элементов*. Элементы, принадлежащие каждому синтаксически выделяемому классу *sc-элементов* должны иметь одинаковые синтаксические признаки (синтаксические метки). При этом очевидно, что *Синтаксис SC-кода* существенно упростится, если синтаксически выделяемые классы sc-элементов будут одновременно иметь и четкую семантическую интерпретацию. Таким образом, формализацию *Синтаксиса SC-кода* целесообразно осуществлять после формализации *Базовой денотационной семантики SC-кода*. Путем синтаксического выделения тех семантически выделенных классов *sc-элементов*, которые, во-первых, необходимы для кодирования sc-конструкций в памяти ostis-систем (в sc-памяти) и во-вторых, обеспечивают максимально возможное упрощение обработки sc-конструкций (например, упрощение анализа часто проверяемых семантических характеристик обрабатываемых *sc-элементов*).

SC-код является одним из возможных вариантов *смыслового представления знаний* (см. *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*)

В основе *SC-кода* лежат следующие понятия:

- ***sc-элемент***
  ≔ [элементарный (атомарный) фрагмент информационной конструкции, принадлежащей SC-коду]
  ≔ [обозначение одной из описываемых сущностей]

- ***sc-множество***
  ≔ [sc-конструкция]
  ≔ [множество sc-элементов]
  ≔ [информационная конструкция SC-кода]

- ***sc-структура***
  ≔ [sc-множество, содержащее связки (знаки связей) между элементами этого множества]

- ***sc-текст***
  ≔ [связная sc-структура, являющаяся семантически корректной в рамках Базовой денотационной семантики SC-кода, а также синтаксически корректной в рамках соответствующей синтаксической модификации SC-кода]

- ***sc-знание***
  ≔ [sc-текст, обладающий дополнительным свойством иметь истинное значение по отношению к соответствующей предметной области]

- ***файл***
  ≔ [информационная конструкция, которая не является sc-конструкцией и которая может храниться в файловой памяти ostis-системы]

- ***sc-идентификатор***
  ⊂ *файл*
  ≔ [файл, являющийся внешним идентификатором (в частности, именем) соответствующего sc-элемента, хранимого в sc-памяти ostis-системы]

- ***основной sc-идентификатор***
  ≔ [sc-идентификатор, который взаимно однозначно соответствует идентифицируемому sc-элементу]

§ 2.2.1. Базовая денотационная семантика SC-кода

⇒ *подраздел**:

Пункт 2.2.1.1. Семантическая классификация sc-элементов по базовым признакам

К числу базовых признаков классификации *sc-элементов* относятся:
- структурный признак;
- логико-семантический признак;
- темпоральная характеристика сущностей, обозначаемых *sc-элементами*, которая, в свою очередь, включает в себя:
  - постоянство или временность существования обозначаемой сущности;
  - статичность (стационарность) или динамичность (изменчивость) обозначаемой сущности.

Структурная классификация sc-элементов

=
{

sc-элемент
⇒ *разбиение**:
Структурный признак классификации sc-элементов
= {• *обозначение sc-множества*
⇒ *разбиение**:
{• *обозначение sc-связки*
⇒ *разбиение**:
{• *обозначение sc-синглетона*
• *обозначение sc-пары*
⇒ *разбиение**:
{• *обозначение неориентированной sc-пары*
• *обозначение ориентированной sc-пары*
⇒ *разбиение**:
{• *обозначение sc-пары принадлежности*
⇒ *разбиение**:
{• *обозначение sc-пары нечеткой принадлежности*
• *обозначение sc-пары позитивной принадлежности*
• *обозначение sc-пары негативной принадлежности*
}
• *обозначение ориентированной sc-пары, не являющейся парой принадлежности*
}
}
• *обозначение sc-связки, не являющейся ни синглетоном, ни парой*
}
• *обозначение sc-класса*
• *обозначение sc-структуры*
}
• *обозначение внешней сущности*
⇒ *разбиение**:
{• *обозначение файла*
• *обозначение информационной конструкции, не являющейся ни sc-множеством, ни файлом*
• *обозначение внешней сущности, не являющейся информационной конструкцией*
}
}
} /* Завершили представление *Структурной классификации sc-элементов* */

Логико-семантическая классификация sc-элементов

=
{

sc-элемент
⇒ *разбиение**:
{• *sc-константа*
:= [sc-элемент, логико-семантическим значением которого является он сам]
• *sc-переменная*
⇒ *разбиение**:
{• *sc-переменная 1-го уровня*
:= [sc-элемент, областью возможных значений которого является множество sc-констант]
• *sc-переменная 2-го уровня*
:= [sc-элемент, возможными значениями которого являются переменные 1-го уровня]
⇒ *примечание**:
[такие переменные (метапеременные) необходимы для описания логических языков]
• *sc-переменная универсального типа*
:= [sc-переменная, на значения которой не накладывается никаких ограничений]
}
}
} /* Завершили представление *Логико-семантической классификации sc-элементов* */

Классификация sc-элементов по темпоральным характеристикам обозначаемых ими сущностей

=
{

sc-элемент
⇒ *разбиение**:
Признак постоянства существования сущностей, обозначаемых sc-элементами
= {• *обозначение постоянной сущности*
:= [обозначение постоянно существующей сущности]
• *обозначение временной сущности*
⇒ *разбиение**:
{• *обозначение внешней временной сущности*
⊃ *обозначение внешней ситуации*
⊃ *обозначение внешнего события*
⊃ *обозначение внешнего процесса*
• *обозначение временной сущности в sc-памяти*
⇒ *разбиение**:
{• *обозначение ситуации в sc-памяти*
:= [обозначение ситуации, которая возникла или возникает в процессе обработки информации в sc-памяти]
• *обозначение события в sc-памяти*
:= [обозначение события, которое произошло или произойдет в процессе обработки информации в sc-памяти]
• *обозначение информационного процесса в sc-памяти*
:= [обозначение внутреннего процесса в sc-памяти, который происходит, произошел или будет происходить]
}
}
}
⇒ *разбиение**:
Признак статичности сущностей, обозначаемых sc-элементами
= {• *обозначение статической сущности*
:= [обозначение статичной сущности]
:= [обозначение стационарной сущности]
:= [обозначение сущности, изменения которой в рамках соответствующего интервала времени ее существования считаются несущественными]
⊃ *обозначение статического sc-множества*
• *обозначение динамической сущности*

:= [обозначение сущности изменяющейся во времени]
⊃ *обозначение динамического sc-множетсва*
}
} /* Завершили представление *Классификации sc-элементов по темпоральным характеристикам обозначаемых ими сущностей* */

Когда речь идет о темпоральных свойствах *sc-элементов*, следует четко отличать:
- временный характер присутствия любого *sc-элемента* в составе той *базы знаний* (в той *sc-памяти*) *ostis-системы*, в которой он находится (когда-то он появляется, когда-то может быть удален из *sc-памяти*);
- временный характер присутствия в *sc-памяти* всей заданной *sc-конструкции* (заданного множества sc-элементов) — такую *sc-конструкцию* будем называть *ситуацией в sc-памяти*;
- временный характер существования *внешней сущности*, которую *sc-элемент* обозначает;
- статичный или динамичный (изменчивый) характер *внешней сущности*, обозначаемой *sc-элементом*. Динамический характер внешней сущности, предполагает наличие в *sc-памяти* описания процесса изменения состояния или конфигурации указанной *внешней сущности*;
- *динамическое sc-множество* (динамическая sc-конструкция), являющееся отражением (динамической моделью) соответствующего внешнего процесса (процесса, происходящего во внешней среде);
- *динамическое sc-множество* (динамическая sc-конструкция), являющееся отражением (динамической моделью) соответствующего внутреннего процесса (информационного процесса, происходящего в той же *sc-памяти*, в которой находится *sc-элемент*, обозначающий указанное динамическое *sc-множество*)

Структурная классификация sc-констант

⇒ *примечание**:
[Данная классификация полностью аналогична *Структурной классификации sc-элементов*, в отличие от которой она описывает структурную классификацию только константных sc-элементов (*sc-констант*)]
∈ *sc-структура*
⇔ *аналог**:
Структурная классификация sc-элементов
=
{

sc-константа
⇒ *разбиение**:
{• *sc-множество*
⇒ *разбиение**:
{• *sc-связка*
⇒ *разбиение**:
{• *sc-синглетон*
• *sc-пара*
⇒ *разбиение**:
{• *неориентированная sc-пара*
• *ориентированная sc-пара*
⇒ *разбиение**:
{• *sc-пара принадлежности*
⇒ *разбиение**:
{• *sc-пара нечеткой принадлежности*
• *sc-пара позитивной принадлежности*
⊃ *sc-пара постоянной позитивной принадлежности*
⇐ *пересечение множеств**:
{• *sc-константа*
• *постоянная сущность*
• *статическая сущность*
• *sc-пара позитивной принадлежности*
}
⊃ *sc-пара временной позитивной принадлежности*
⇐ *пересечение множеств**:
{• *sc-константа*
• *временная сущность*
• *динамическая сущность*
• *sc-пара позитивной принадлежности*
}
• *sc-пара негативной принадлежности*

}
- *ориентированнная sc-пара, не являющаяся парой принадлежности*

}
}
- *sc-связка, не являющаяся ни синглетоном, ни парой*

}
- *sc-класс*
- *sc-структура*

}
- *внешняя сущность*
  - := [sc-элемент, являющийся знаком внешней сущности]
  - := [знак внешней сущности]
  - := [знак сущности, не являющейся sc-множеством (sc-конструкцией)]
  - ⇒ *разбиение**:
    {• *файл*
    • *информационная конструкция, не являющаяся ни sc-множеством, ни файлом*
    • *внешняя сущность, не являющаяся информационной конструкцией*
    }

}
} /* Завершили представление *Структурной классификации sc-констант* */

Пункт 2.2.1.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами

Перейдем к детальному рассмотрению смысла классов *sc-элементов* (sc-классов), введенных в представленных выше классификациях.

Указанные *sc-классы* рассматриваются в порядке их введения в представленных выше классификациях *sc-элементов*.

Сначала поясним смысл понятий (sc-классов), введенных в *Структурной классификации sc-элементов* и в *Структурной классификации sc-констант*.

sc-элемент
:= [обозначение множества]
:= [sc-обозначение множества, представимого в SC-коде]
⇒ *разбиение**:
{• *обозначение sc-множества*
  := [обозначение множества *sc-элементов*]
  := [обозначение множества, все элементы которого являются *sc-элементами*]
  := [обозначение внутренней для sc-памяти сущности, то есть сущности, хранимой в sc-памяти]
• *обозначение внешней сущности*
  := [обозначение синглетона внешней сущности]
  := [терминальный *sc-элемент*]
}
⇒ *примечание**:
- [Каждый *sc-элемент* является обозначением соответствующего множества.]
- [Ко множествам, представимым в *SC-коде*, относятся либо *sc-множества*, элементами которых являются *sc-элементы*, либо синглетоны, элементами которых являются сущности, не являющиеся *sc-элементами* (синглетоны внешних сущностей). Таким образом, строго говоря, не каждое множество может быть обозначено соответствующим *sc-элементом* и представлено в SC-коде. Но каждое множество, не являющееся *sc-множеством* или синглетоном указанного выше вида может быть однозначно преобразовано в *sc-множество* и описано средствами *SC-кода*. При этом теоретико-множественные свойства "нестандартных" для *SC-кода* множеств совпадают со свойствами соответствующих им "стандартных" для *SC-кода* множеств.]
- [Тот факт, что каждый *sc-элемент* является обозначением соответствующего множества (частным случае которых являются синглетоны внешних описываемых сущностей), означает то, что базовым видом объектов, которыми оперирует *SC-код* на синтаксическом, семантическом и логическом уровне являются

множества знаков, обозначающих различные множества. В этом смысле *SC-код* имеет базовую теоретико-множественную основу.]

⇒ *правила построения внешних идентификаторов sc-элементов заданного класса**:
Общие правила построения внешних идентификаторов sc-элементов
:= [Общие правила идентификации *sc-элементов*]
= {• [Принадлежность идентифицируемого *sc-элемента* некоторым классам *sc-элементов* (sc-классам) явно указывается во внешнем идентификаторе этого *sc-элемента* (в *sc-идентификаторе*) с помощью соответствующих условных признаков:
- если первым символом *sc-идентификатора* является знак подчеркивания, то идентифицируемый *sc-элемент* принадлежит Классу *sc-переменных*. По умолчанию считается, что идентифицируемый *sc-элемент* принадлежит Классу *sc-констант*;
- если последним символом *sc-идентификатора* является символ “звездочка”, то идентифицируемый *sc-элемент* принадлежит Классу обозначений *неролевых отношений*;
- если последним символом *sc-идентификатора* является апостроф, то идентифицируемый *sc-элемент* принадлежит Классу обозначений *ролевых отношений*, каждое из которых является подмножеством Отношения принадлежности, то есть Класса всех *константных позитивных sc-пар принадлежности*;
- если последним символом *sc-идентификатора* является символ “^”, то идентифицируемый *sc-элемент* принадлежит Классу обозначений *параметров*.

]
• [Слово “обозначение” в *sc-идентификаторе* означает то, что обозначаемая сущность может быть как константной, так и переменной.]
• [В *sc-идентификаторах* можно делать следующие сокращения:
- “sc-элемент, обозначающий . . . ” — “обозначение”
- “обозначение константного” — “знак константного”
- “знак константного” — “константный”
- слово “константный” в *sc-идентификаторах* можно опускать, так как константность подразумевается по умолчанию

]
• [Для каждого *sc-элемента* можно построить *sc-идентификатор*, являющийся *именем собственным*, которое всегда начинается с большой буквы (заглавной) буквы.]
• [Если *sc-элемент* является обозначением некоторого класса *sc-элементов*, то этому *sc-элементу* можно поставить в соответствие не только *имя собственное*, но и *имя нарицательное*, которое начинается маленькой (строчной) буквы. В спецификацию каждого sc-класса (каждого понятия) входит перечень эквивалентных (синонимичных) *sc-идентификатор*, среди которых есть как *имена собственные*, так и *имена нарицательные*.]
}

обозначение sc-множества
:= [SC-элемент, являющийся знаком множества всевозможных *обозначений sc-множеств*]
∈ *имя собственное*
:= [Знак множества всевозможных *обозначений sc-множеств*]
∈ *имя собственное*
:= [Множество всевозможных *обозначений sc-множеств*]
∈ *имя собственное*
:= [Класс *обозначений sc-множеств*]
∈ *имя собственное*
:= [sc-элемент, являющийся обозначением множества *sc-элементов*]
∈ *имя нарицательное*
:= [sc-обозначение множества *sc-элементов*]
∈ *имя нарицательное*
:= [обозначение множества, каждый элемент которого является *sc-элементом*]
:= [обозначение информационной конструкции, принадлежащей *SC-коду*]
:= *часто используемый sc-идентификатор**:
[обозначение *sc-конструкции*]
⇒ *разбиение**:
{• *sc-множество*
:= [знак константного *sc-множества*]
= (*обозначение sc-множества* ⋂ *sc-константа*)
• *переменное sc-множество*
= (*обозначение sc-множества* ⋂ *sc-переменная*)
}

следует отличать*

∋ {• *обозначение sc-множества*
:= [*обозначение sc-множества*, которое может быть как константным sc-множеством, так и переменным sc-множеством]
:= [обозначение внутренней для *sc-памяти* сущности]
:= [обозначение внутренней для *sc-памяти* информационной конструкции (*sc-конструкции*)]
⇒ *разбиение**:
{• *sc-множество*
:= [обозначение конкретного множества]
:= [знак множества]
= (*sc-константа* ⋂ *обозначение sc-множества*)
:= [конкретное *sc-множество*]
:= [знак константного *sc-множества*]
:= [константное *sc-множество*]
• *переменное sc-множество*
:= [произвольное *sc-множества*]
:= [обозначение произвольного *sc-множества*]
= (*sc-переменная* ⋂ *обозначение sc-множества*)
}
• *sc-множество*
• *переменное sc-множество*
• *обозначение внешней сущности*
:= [обозначение сущности, не являющейся множеством sc-элементов (*sc-множеством*)]
⊃ *обозначение файла*
:= [*обозначение файла*, хранимого либо в файловой памяти той же *ostis-системы*, в *sc-памяти* которой хранится знак этого *файла*, либо в файловой памяти другой дополнительно указываемой *компьютерной системы*]
⊃ *обозначение информационной конструкции, не являющейся ни sc-множеством, ни файлом*
⇒ *примечание**:
[Примерами такой информационной конструкции являются напечатанный текст, речевое сообщение, которой следует отличать от его записи в виде аудио-файла.]
⊃ *обозначение внешней сущности, не являющейся информационной конструкцией*
⇒ *примечание**:
[Примером такой внешней сущности является любой материальный объект, не являющийся информационной конструкцией]
}

sc-множество

:= [*sc-конструкция*]
:= [информационная конструкция, принадлежащая *SC-коду*]
:= *часто используемый sc-идентификатор**:
[*SC-код*]
∈ *имя собственное*
:= [Множество всевозможных *sc-конструкций*]

обозначение sc-связки

⇒ *разбиение**:
{• *sc-связка*
• *переменная sc-связка*
}

sc-связка

:= [знак связи (связки) между *sc-элементами*]
⇒ *примечание**:
[Если элементами *sc-связки* являются знаки *внешних сущностей*, то *sc-связка* является отображением (моделью) некоторой связи, которая связывает указанные *внешние сущности*]
⇒ *пояснение**:
[Понятие *sc-связки* — это попытка формализации понятия целостности, понятия перехода некоторой совокупности сущностей в некоторое новое качество, которое не сводится к свойствам каждой сущности, входящей в эту совокупность. Таким образом, связками следует считать:
• множество всех чисел, являющихся слагаемыми для заданного числа;
• множество всех сотрудников заданной организации, в заданный момент времени;

- множество всех сотрудников заданной организации, которые работают или работали в ней.

]

⇒ *примеры**:

[Примерами *sc-связок* являются:

- конкретная окружность, (множество всех точек, равноудаленных от некоторой заданной точки);
- конкретный отрезок (множество всех точек, лежащих между двумя заданными точками с включением этих точек);
- конкретный линейный треугольник (множество всех точек, лежащих между каждыми двумя из трех заданных точек с включением этих точек);
- пары граничных точек различных отрезков;
- тройки вершин различных треугольников.

]

обозначение sc-синглетона

⇒ *разбиение**:

{• *sc-синглетон*
• *переменный sc-синглетон*
}

sc-синглетон

:= [*sc-множество*, являющиеся синглетоном]
:= [одномощное *sc-множество*]
:= [*sc-множество*, имеющее мощность, равную единице]
:= [*sc-элемент*, являющийся знаком унарной *sc-связки*]
:= [знак унарной *sc-связки*]
:= [унарная *sc-связка*]
:= [знак одномощного множества, единственный элемент которого является *sc-элементом*]

обозначение sc-пары

∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:

{• ***sc-пара***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная sc-пара***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

обозначение неориентированной sc-пары

∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:

{• ***неориентированная sc-пара***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная неориентированная sc-пара***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

обозначение ориентированной sc-пары

∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:

{• ***ориентированная sc-пара***

:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
- ***переменная ориентированна sc-пара***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*

}

обозначение sc-пары принадлежности
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-пара принадлежности***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
- ***переменная sc-пара принадлежности***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*

}

обозначение sc-пары нечеткой принадлежности
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-пара нечеткой принадлежности***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
- ***переменная sc-пара нечеткой принадлежности***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*

}

обозначение sc-пары позитивной принадлежности
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-пара позитивной принадлежности***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
- ***переменная sc-пара позитивной принадлежности***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*

}

sc-пара постоянной позитивной принадлежности
:= [константная позитивная постоянная sc-пара принадлежности]
:= [sc-пара константной постоянной позитивной принадлежности]

sc-пара временной позитивной принадлежности
:= [sc-пара константной временной позитивной принадлежности]

обозначение sc-пары негативной принадлежности
∈ *sc-константа*
∈ *sc-класс*

⇒ *разбиение**:
{• ***sc-пара негативной принадлежности***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная sc-пара негативной принадлежности***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

обозначение sc-пары, не являющейся парой принадлежности
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-пара, не являющаяся парой принадлежности***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная sc-пара, не являющаяся парой принадлежности***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

обозначение sc-связки, не являющейся ни синглетоном, ни парой
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-связка, не являющаяся ни синглетоном, ни парой***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная sc-связка, не являющаяся ни синглетоном, ни парой***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

обозначение sc-класса
⇒ *разбиение**:
{• *sc-класс*
• *переменный sc-класс*
}
⇒ *разбиение**:
{• *обозначение sc-класса обозначений sc-связок*
• *обозначение sc-класса обозначений sc-классов*
• *обозначение sc-класса обозначений sc-структур*
• *обозначение sc-классов обозначений внешних сущностей*
}

sc-класс
⇒ *разбиение**:
{• *sc-класс sc-связок*
⊃ *sc-отношение*
⊃ *бинарное sc-отношение*
⇒ *разбиение**:
{• *бинарное неориентированное sc-отношение*
• *бинарное ориентированное sc-отношение*
⊃ *ролевое sc-отношение*
}

- *sc-класс sc-классов*
  ⊃ *sc-параметр*
- *sc-класс sc-структур*
- *sc-класс внешних сущностей*
  ⊃ *sc-класс файлов*
  ≔ [*sc-класс* sc-элементов, являющихся знаками *внешних сущностей*]
- *sc-класс sc-констант разного структурного типа*
  ∋ *пример'*:
  - *sc-константа*
  - *постоянная сущность*

}

⇒ *пояснение**:

[Требованием, предъявляемым к каждому *sc-классу* является наличие общего свойства, присущего всем элементам этого *sc-класса*. Формулировку указанного общего свойства обычно называют *определением* соответствующего *sc-класса* (в частности, *понятия*). Некоторые *sc-классы* могут быть заданы с помощью *отношений эквивалентности*, если эти классы являются *классами эквивалентности* соответствующих *отношений эквивалентности*, то есть являются элементами *фактор-множеств*, соответствующих этим *отношениям*.]

следует отличать*

∋ {• *sc-связка*
- *sc-класс*

}

⇒ *сравнение**:

[В отличие от *sc-связки* принципом формирования *sc-класса* является наличие общего свойства, присущего всем элементам этого *sc-класса* и только им, (или присущего всем сущностям, которые обозначаются указанными *sc-элементами*). Таким общим свойством может быть определение *sc-класса* либо принадлежность одному из значений некоторого параметра, то есть одному из элементов *фактор-множества*, соответствующего некоторому *отношению эквивалентности* или толерантности.]

⇒ *пояснение**:

[Примерами *связок* являются:

- множество людей живущих сейчас (динамическое множество);
- множество сотрудников некоторой конкретной организации (динамическое множество);
- конкретный отрезок, конкретный треугольник.

Здесь речь не идет об эквивалентности свойств самих людей и геометрических точек безотносительно к тому, в состав чего они входят. Поэтому это не является *sc-классом*.]

∋ {• *sc-класс эквивалентности*
  ⇒ *пояснение**:
  [В *sc-класс эквивалентности* входит не просто некоторое количество попарно эквивалентных между собой сущностей, а абсолютно все такие сущности]
- *sc-связка попарно эквивалентных сущностей*

}

∋ {• *множество всех треугольников, подобных одному из них*
  ⊂ *sc-класс*
- *конечное множество подобных треугольников*
  ⊂ *sc-связка попарно эквивалентных треугольников*

}

∋ {• *sc-параметр*
  ≔ *часто используемый sc-идентификатор**:
  [параметр]
  ⊂ *sc-класс sc-классов*
- *признак различия*
  ≔ [признак классификации]

}

⇒ *пояснение**:

{

параметр

⊂ *бесконечное множество*

признак различия

⊂ *конечное множество*

∋ *пример′*:
- *Признак конечности множеств*
  = {• *конечное множество*
  • *бесконечное множество*
  }
- *Признак наличия кратных элементов*
  = {• *мультимножество*
  • *множество без кратных вхождений элементов*
  }

}

sc-класс

⇒ *правила построения внешних идентификаторов sc-элементов заданного класса**:
Правила построения внешних идентификаторов sc-элементов, являющихся знаками sc-классов
= {• [Слово “обозначение” в начале идентификатора используется тогда, когда в идентифицируемый класс sc-элементов включаются знаки как константных, так и переменных сущностей соответствующего вида]
• [Слово “переменный” в начале идентификатора используется, когда элементами идентифицируемого sc-класса являются только sc-переменные]
• [Слово “константный” в начале идентификатора можно опустить, так как константность подразумевается по умолчанию]
}

обозначение sc-структуры

∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***sc-структура***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменная sc-структура***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

следует отличать*

∋ {• *sc-структура*
• *sc-связка*
}
⇒ *сравнение**:
{• [В отличие от *sc-связок* в каждую *sc-структуру* должна входить по крайней мере одна *sc-связка* вместе с компонентами этой *sc-связки*]
}

обозначение внешней сущности

:= [обозначение сущности, не являющейся sc-множеством]

внешняя сущность

:= [синглетон внешней сущности]
:= [сущность, не являющаяся sc-множеством]
:= [обозначение синглетона внешней сущности]
:= [*sc-элемент*, обозначающий синглетон, элементом которого является некоторая внешняя описываемая сущность]
:= [множество, являющееся 1-мощным множеством, единственным элементом которого является сущность, внешняя по отношению к sc-памяти, то есть сущность, не являющаяся *sc-элементом*]
⇒ *примечание**:
• [обозначение внешней сущности, то есть *sc-элемент*, обозначающий соответствующий синглетон, можно также трактовать как *sc-элемент*, обозначающий соответствующую внешнюю описываемую сущность, которую, в свою очередь, можно считать денотатом указанного *sc-элемента*]
•

[очевидно, что пара принадлежности, связывающая *sc-элемент*, обозначающий синглетон внешней сущности, не может быть непосредственно представлена в виде соответствующей *sc-пары принадлежности*, так как второй компонент этой *sc-пары* не находится в *sc-памяти*]

следует отличать*
∋ {• *внешня сущность*
• *sc-синглетон*
:= [синглетон, единственным элементом которого является некоторый *sc-элемент*]
⊂ *sc-множество*
:= [*sc-элемент*, обозначающий множество, элементами которого являются только sc-элементы]
:= [множество *sc-элементов*]
}

обозначение файла
∈ *sc-константа*
∈ *sc-класс*
⇒ *разбиение**:
{• ***файл***
:= [константная sc-пара]
⊂ *sc-константа*
∈ *sc-константа*
∈ *sc-класс*
• ***переменный файл***
⊂ *sc-переменная*
∈ *sc-константа*
∈ *sc-класс*
}

файл
:= [внутренний образ (копия) информационной конструкции, хранимый в *файловой памяти ostis-системы*]
:= [файл *ostis-системы*]
⇒ *примечание**:
• [*файловая память ostis-системы*, хранящая различного рода *информационные конструкции* (образы, модели), не являющиеся *sc-конструкциями*, должна быть тесно связана с *sc-памятью* этой же *ostis-системы*. Как минимум, каждый *файл ostis-системы* должен быть связан с тем *sc-элементом*, которых является знаком этого *файла* (точнее, знаком синглетона, элементом которого является указанный файл)]

Перейдем к пояснению смысла понятий используемых в *Логической классификации sc-элементов*.

sc-константа
:= [sc-элемент, обозначающий константную сущность]
⇒ *сокращение**:
[обозначение константной сущности]
:= [обозначение константной сущности]
:= [знак константной сущности]
⇒ *сокращение**:
[константная сущность]
⇒ *сокращение**:
[сущность]
:= [константная сущность]
:= [конкретная сущность]
:= [сущность]
:= [константный sc-элемент]
:= [sc-элемент, имеющий одно логико-семантическое значение, каковым является он сам]
:= [sc-элемент, являющийся знаком константной (конкретной, фиксированной) сущности]
⇒ *сокращение**:
[знак константной (конкретной, фиксированной) сущности]
⇒ *сокращение**:
[константная (конкретная, фиксированная) сущность]
⇒ *сокращение**:
[константная сущность]

sc-переменная
:= [переменный sc-элемент]
:= [sc-элемент, являющийся обозначением некоторой произвольной (нефиксируемой, переменной) сущности]
⇒ *сокращение**:
[обозначение произвольной (переменной) сущности]
⇒ *сокращение**:
[переменная сущность]
∈ *sc-константа*
∈ *sc-класс*
⇒ *примечание**:
[Сам *sc-элемент*, имеющий внешний идентификатор "*sc-переменная*" является *sc-константой* (константным sc-элементом), которая является знаком соответствующего класса sc-элементов.]

sc-элемент
:= [обозначение константной или переменной сущности]
:= [константная или переменная сущность]
:= [sc-константа или sc-переменная]
:= [обозначение описываемой сущности, которая может быть как константной, так и переменной сущностью, как внутренней, так и внешней sc-конструкцией для заданной ostis-системы, как информационной конструкцией, так и сущностью которая информационной конструкцией не является, как временной сущностью, так и постоянной, как динамической, так и статической сущностью]

обозначение sc-множества
⇒ *разбиение**:
{• *sc-множество*
:= *часто используемый sc-идентификатор**:
[множество sc-элементов]
:= [константное (конкретное) sc-множество]
:= [обозначение (знак) конкретного множества]
⊂ *sc-константа*
∈ *sc-константа*
⇒ *разбиение**:
{• *sc-множество sc-констант*
:= [sc-множество, элементами которого являются только sc-константы]
:= [множество, являющееся подмножеством Множества всевозможных констант]
• *sc-множество sc-переменных*
:= [sc-множество, элементами которого являются только sc-переменные]
• *sc-множество sc-констант и sc-переменных*
:= [множество, элементами которого являются как константы, так и переменные]
∈ *sc-константа*
⇔ *следует отличать**:
sc-множество sc-переменных
}
• *переменное sc-множество*
:= [обозначение переменного (произвольного) sc-множества]
}

Перейдем к пояснению смысла понятий, используемых в *Классификации sc-элементов по темпоральным характеристикам обозначаемых ими сущностей*.

обозначение временной сущности
⇒ *разбиение**:
{• *обозначение временной сущности существующей сейчас*
:= [обозначение временной сущности, существующей в текущий (настоящий) момент]
• *обозначение прошлой временной сущности*
:= [обозначение бывшей временной сущности]
:= [обозначение временной сущности, которая уже перестала существовать, прекратила свое существование]
• *обозначение будущей временной сущности*
:= [обозначение временной сущности, появление которой прогнозируется (планируется, обеспечивается)]
⇒ *примечание**:

[проектирование и производство новых, ранее не существующих полезных сущностей — это основное направление человеческой деятельности]

}

⇒ *примечание**:

[ostis-системы должны постоянно мониторить состояние временных сущностей, так как в процессе их функционирования будущие сущности становятся настоящими, а настоящие — прошлыми]

динамическое sc-множество

:= [sc-процесс]

:= [процесс]

⇒ *определение**:

[*sc-множество*, у которого некоторые позитивные пары принадлежности, связывающие знак этого множества с его элементами, носят временный характер]

⇒ *примечание**:

[Сами элементы *динамического sc-множества*, связанные с ним временными позитивными парами принадлежности, могут обозначать как временные, так и постоянные сущности. Но чаще всего такими временными элементами динамического sc-множества являются знаки временных связок.]

⇒ *разбиение**:

{• *внешний процесс*

• *процесс в sc-памяти*

}

темпоральная декомпозиция динамического sc-множества

:= [покадровая развертка динамического sc-множества]

:= [представление sc-множества в виде кортежа (последовательности) ситуаций]

следует отличать*

∋ {• *временная сущность*

• *обозначение временной сущности*

• *переменная временная сущность*

}

обозначение временной сущности

⇒ *разбиение**:

{• *временная сущность*

:= [знак конкретной (константной) временной сущности]

• *переменная временная сущность*

:= [обозначение произвольной временной сущности]

}

сформированное sc-множество

:= [sc-множество, у которого в текущем состоянии sc-памяти перечислены все его элементы]

∈ *динамическое sc-множество*

⇒ *примечание**:

[Очевидно, что сформированным sc-множеством может стать только конечное sc-множество]

формируемое sc-множество

sc-множество, элементы которого не известны

сформированный файл

формируемый файл

файл, структура которого не известна

Перейдем к рассмотрению семантически выделяемых классов *sc-элементов*, которые необходимо ввести дополнительно к выше рассмотренным классам *sc-элементов*.

sc-элемент, не являющийся ни sc-синглетоном, ни sc-парой

sc-элемент, копируемый в других компьютерных системах

:= [*sc-элемент*, имеющий в других компьютерных системах свои копии и/или копии обозначаемой им информационной конструкции]

отношение, заданное на множестве sc-элементов, копируемых в других компьютерных системах

∋ *ostis-система, в sc-памяти которой хранится копия заданного sc-элемента**

∋ *компьютерная система, в файловой памяти которой хранится заданный файл**

⇒ *примечание**:
[указанная компьютерная система назначается хранителем файла]

∋ *ostis-система, в sc-памяти которой хранится копия знака заданного sc-множества и все известные в текущий момент его элементы**
⇒ *примечание**:
[указанная ostis-система назначается основным хранителем указанного sc-множества]

информационная конструкция
⇒ *разбиение**:
{• *sc-множество*
:= [sc-конструкция]
:= [информационная конструкция *SC-кода*]
:= [внутренняя информационная конструкция *ostis-системы*, хранимая в ее *sc-памяти*]
• *файл*
:= [файл ostis-системы]
:= [информационная конструкция *ostis-системы*, хранимая в ее файловой памяти]
⇒ *примечание**:
[файл, может храниться в памяти другой компьютерной системы и, в частности, в файловой памяти другой *ostis-системы*]
• *внешняя информационная конструкция, не являющаяся ни файлом, ни sc-конструкцией*
}

sc-идентификатор
:= [внешний идентификатор sc-элемента]
⊃ *файл*
⇒ *разбиение**:
{• *основной идентификатор*
• *часто используемый sc-идентификатор*
• *дополнительный sc-идентификатор*
}

sc-идентификатор*
:= [бинарное ориентированное отношение, связывающее *sc-элементы* с их внешними идентификаторами]

Пункт 2.2.1.3. Структура базовой семантической спецификации sc-элемента

базовая семантическая спецификация sc-элемента
:= *пояснение**:
[Класс *sc-структур*, каждая из которых описывает базовые семантические свойства (характеристики) соответствующего (описываемого, специфицируемого) *sc-элемента*]
⊂ *sc-структура*
⊂ *sc-спецификация*
:= [представленная в *SC-коде* семантическая окрестность (спецификация) некоторого (специфицируемого) *sc-элемента*]
⊂ *sc-спецификация*
⇐ *второй домен**:
*базовая семантическая спецификация sc-элемента**
:= *пояснение**:
[бинарное ориентированное отношение, каждая пара которого связывает *sc-элемент* с его базовой семантической спецификацией*]
:= *пояснение**:
[хранимая в *sc-памяти* ostis-системы спецификация каждого *sc-элемента*, необходимая для эффективной обработки этого *sc-элемента*]
⇒ *примечание**:
[базовая спецификация *sc-элементов* осуществляется как явно с помощью соответствующих sc-конструкций, так и неявно с помощью соответствующих семантических меток, приписываемых sc-элементам]
⇒ *пояснение**:
[Базовая семантическая спецификация каждого *sc-элемента* включает в себя:

- перечисление всех тех *базовых классов sc-элементов*, которым принадлежит специфицируемый *sc-элемент*;

- уточнение "привязки" временной сущности, обозначаемой специфицируемым sc-элементом к текущему моменту и другим моментам времени;
- уточнение того, какие важные характеристики специфицируемого *sc-элемента* в текущем состоянии *sc-памяти* и файловой памяти *ostis-системы* не известны.

]

базовая семантическая спецификация sc-элемента, обозначающего временную сущность включает в себя указание дополнительных темпоральных характеристик, позволяющих уточнить темпоральные "координаты" этих временных сущностей (то есть их "координаты" во времени), а также их основные темпоральные связи с другими временными сущностями. К числу понятий, обеспечивающих описание указанных темпоральных характеристик временных сущностей, относятся:

- *момент времени^*
- *Текущий момент времени*
- *прошлая сущность*
- *будущая сущность*
- *момент начала**
- *момент завершения**
- *внешняя ситуация*
- *ситуация в sc-памяти*
- *внешнее событие*
- *событие в sc-памяти*
- *внешний процесс*
- *процесс в sc-памяти*

момент времени^

∈ *параметр*

∈ *параметр, заданный на множестве временных сущностей*

:= [глобальная приблизительно точная ситуация^]

:= [глобальная ситуация пренебрежительно малого отрезка времени^]

:= [множество (класс) всех временных сущностей, существующих одновременно в соответствующий момент времени^]

⇒ *примечание**:

[момент времени, соответствующий глобальной точечной ситуации может быть задан с различной и дополнительно указываемой степенью точности — с точностью до секунды, до минуты, до часа, до даты, до календарного месяца, до календарного года и так далее. В том смысле корректнее говорить не о моменте времени, а об интервале времени, длительность которого считается пренебрежимо малой для рассмотрения описываемых процессов]

Текущий момент времени

:= [Глобальная ситуация текущего (настоящего) момента времени]

:= [Глобальная ситуация, имеющая место сейчас]

:= [Класс всех сущностей, существующих в настоящий момент времени]

∈ *sc-синглетон*

∈ *динамическое sc-множество*

⇐ *включение множества**:

момент времени

⇒ *пояснение**:

[Из знака *Текущего момента времени* (который является также знаком *sc-синглетона*) "выходит" sc-пара временной принадлежности, представляющая собой, образно говоря, "стрелку" внутренних часов *ostis-системы*, которая всегда указывает только на один элемент множества моментов времени, но в разные моменты времени указывает на разные элементы этого множества]

прошлая сущность

:= [временная сущность, уже завершившая свое существование]

будущая сущность

:= [прогнозируемая, планируемая или создаваемая временная сущность]

момент начала*

:= [момент времени, соответствующий началу существования заданной временной сущности]

:= [бинарное ориентированное отношение, каждая пара которого, связывает (1) знак некоторой временной сущности и (2) глобальную точечную ситуацию (значение параметра "*момент времени^*"), элементом которой является условно точечная временная сущность, представляющая собой начальный этап существования временной сущности, указанной в первом компоненте рассматриваемой ориентированной пары]

⇒ *примечание**:
[Начальный этап существования временной сущности (переходный процесс от небытия к реальному существованию) может рассматриваться с любой степенью детализации]

⇒ *первый домен**:
временная сущность

⇒ *второй домен**:
*момент времени^*

момент завершения*

:= [момент времени, соответствующий завершению существования заданной временной сущности]

ситуация

⇒ *разбиение**:
{• *внешняя ситуация*
• *ситуация в sc-памяти*
}

событие

⇒ *разбиение**:
{• *внешнее событие*
• *событие в sc-памяти*
}

динамическое sc-множество

⇒ *разбиение**:
{• *внешний процесс*
:= [процесс, происходящий в окружающей среде ostis-системы]
• *процесс в sc-памяти*
}

внешняя ситуация

:= [ситуация во внешней среде]

:= [ситуация одновременного существования (в соответствующий период времени) указанных временных внешних сущностей]

⊂ *временная сущность*

⊂ *sc-структура*

⊂ *sc-константа*

⊂ *обозначение внешней ситуации*

∈ *sc-класс*

класс внешних ситуаций

⇒ *примечание**:
[В простейшем случае внешние ситуации, входящие в класс внешних ситуаций являются изоморфными]

внешний процесс

:= [темпоральная детализация внешней динамической сущности]

внешнее событие

:= [факт появления (возникновения) некоторой внешней сущности (в том числе некоторой внешней ситуации) или факт завершения существования некоторой внешней сущности (в том числе некоторой внешней ситуации)]

ситуация в sc-памяти
:= [внутренняя ситуация]
:= [sc-ситуация]
:= [хранимый в sc-памяти фрагмент базы знаний, рассматриваемый в контексте его появления в sc-памяти или его исчезновения (из-за удаления некоторые sc-элементов)]

класс ситуаций в sc-памяти
:= [класс внутренних ситуаций]

обобщенное описание класса ситуаций в sc-памяти

процесс в sc-памяти
:= [внутренний процесс]
:= [информационный процесс, происходящий в sc-памяти]
:= [sc-процесс]

событие в sc-памяти

Важной частью ***базовой семантической спецификации sc-элемента*** является фиксация того, что *ostis-система* знает и чего она не знает о специфицируемом *sc-элементе* или об обозначенной им сущности:
- Если в спецификации *sc-элемента* указывается его принадлежность к некоторому классу *sc-элементов*, но не указывается его принадлежность одному из подклассов, на которые *разбивается* указанный выше класс, то это означает, что в текущий момент времени *ostis-система* этого не знает;
- Если специфицируемый *sc-элемент* является обозначением конечного множества sc-элементов (в частности, пары *sc-элементов*), и если в текущий момент времени *ostis-системе* не известны все этого множества (то есть специфицируемый *sc-элемент* не соединен соответствующими парами принадлежности со всеми элементами обозначаемого им множества *sc-элементов*), то этот специфицируемый *sc-элемент* следует отнести к *sc-классу* "***обозначение несформированного sc-множества***";
- Если специфицируемый *sc-элемент* является обозначением ориентированной *sc-пары* и если в текущий момент времени *ostis-системе* не известна направленность этой ориентированной пары *sc-элементов* (то есть не известно, какой элемент этой пары является первым ее компонентом, а какой ее элемент является ее вторым компонентом), то этот специфицируемый *sc-элемент* следует отнести к *sc-классу* "***обозначение ориентированной sc-пары неизвестной направленности***".

К числу понятий, используемых для описания полноты ***базовой семантической спецификации sc-элемента***, относятся:
- *обозначение бесконечного sc-множества*;
- *обозначение конечного sc-множества*;
- *мощность обозначаемого sc-множества**;
- *обозначение sc-множества неизвестной мощности*;
- *обозначение sc-множества, о котором не известно, является ли оно sc-парой*;
- *обозначение sc-пары, о которой не известно, является ли она ориентированной или нет*;
- *обозначение ориентированной sc-пары неизвестной направленности*;
- *обозначение сформированного sc-множества*;
- *обозначение несформированного sc-множества*;
- *обозначение частично сформированного sc-множества*;
- *обозначение полностью несформированного sc-множества*;
- *обозначение сформированного файла*;
- *обозначение несформированного файла*;
- *обозначение частично сформированного файла*;
- *обозначение полностью несформированного файла*;

Подчеркнем то, что базовую семантическую спецификацию должны иметь абсолютно все *sc-элементы*, хранимые в *sc-памяти* в текущий момент времени, в том числе и все *sc-элементы*, являющиеся ключевыми знаками в рамках ***Предметной области Базовой денотационной семантики SC-кода***. Приведем пример базовой семантической спецификации одного из таких sc-элементов:

обозначение sc-множества
:= [Множество всевозможных sc-элементов, обозначающих sc-множества]
∈ *имя собственное*
∈ *обозначение sc-множества*

⇒ *примечание**:
[Одним из элементов данного множества является знак, обозначающий это множество. Это означает, это данное множество является *рефлексивным множеством*]
∈ *обозначение множества sc-элементов разного структурного типа*
⇒ *примечание**:
[Элементами данного множества являются обозначения различных:
- sc-синглетонов;
- sc-пар;
- sc-связок, не являющихся ни sc-синглетонами, ни sc-парами;
- sc-классов;
- sc-структур.

]
∈ *обозначение множества sc-элементов, содержащего как константные, так и переменные sc-элементы*
∈ *sc-константа*
⇒ *примечание**:
[Само данное множество является константным, несмотря на то, что его элементами являются как sc-константы, так и sc-переменные]
∈ *обозначение множества sc-элементов, содержащего sc-элементы, обозначающие как постоянные, так и временные сущности*
∈ *постоянная сущность*
⇒ *примечание**:
[Следует отличать постоянство / временность сущности, обозначаемой sc-элементом и постоянство / временность sc-множества, одним из элементов которого указанный sc-элемент является]
∈ *обозначение множества sc-элементов, содержащего sc-элементы, обозначающие как статические, так и динамические sc-множества*
∈ *статическое sc-множество*
⇒ *примечание**:
- [Следует отличать статичность / динамичность sc-множества, обозначаемого соответствующим sc-элементом и статичность / динамичность sc-множества, одним из элементов которого указанный выше sc-элемент является]
- [Напомним, что статический характер sc-множества означает отсутствие временных sc-пар принадлежности (временных sc-дуг принадлежности), выходящих из знака этого sc-множества]

∈ *sc-класс*
:= [Класс всевозможных sc-элементов, обозначающих sc-множества]
:= [Класс обозначающий sc-множеств]
⇒ *примечание**:
[Следует отличать разные sc-элементы, являющиеся обозначениями соответствующих sc-множеств, и класс, элементами которого являются всевозможные такие sc-элементы]

Пункт 2.2.1.4. Онтологическая формализация Базовой денотационной семантики SC-кода

Суть онтологической формализации различных областей знаний, различных фрагментов *баз знаний* интеллектуальных компьютерных систем заключается в следующем:

- Выделяется достаточно большой семантически целостный фрагмент *баз знаний*, включающий в себя:
  - все элементы некоторого одного ключевого класса рассматриваемых объектов (объектов исследования) или конечного числа таких ключевых классов объектов исследования;
  - все связи между выделенными объектами исследования, соответствующие заданному семейству отношений, параметров и классов структур, которое условно будем называть предметом исследования.
- Указанный семантически целостный фрагмент *базы знаний*, являющийся чаще всего бесконечной структурой, будем называть ***предметной областью***.
- Сама формальная ***онтология*** представляет собой формальную спецификацию выделенной *предметной области* и включает в себя следующие ***частные онтологии***:
  - ***структурную спецификацию предметной области***, в которой указываются роли всех ключевых элементов (ключевых знаков), входящих в состав *предметной области*. К числу таких ролей относятся:
    - *максимальный класс объектов исследования'*
    - *немаксимальный класс объектов исследования'*
    - *ключевой объект исследования'*

- *исследуемый класс связок'*
- *исследуемый класс классов'*
- *исследуемый класс структур'*
- *неисследуемый класс'*
  ≔ [*sc-класс*, исследуемый в другой (смежной) *предметной области*]

- ***теоретико-множественную онтологию***, в которой описываются теоретико-множественные связи между всеми классами (*sc-классами*), исследуемыми в рамках заданной (специфицируемой) *предметной области*
- ***логическую онтологию***, которая включает в себя
  - определения исследуемых классов (исследуемых понятий);
  - логическую иерархию исследуемых понятий, которая связывает каждое понятие со множеством тех понятий, которые явно используются в определении этого понятия;
  - аксиомы и теоремы, описывающие свойства специфицируемой предметной области;
  - тексты доказательств теорем;
  - логическую иерархию теорем, которая связывает каждую теорему со множеством теорем, на основе которых она доказывается.

- ***терминологическую спецификацию предметной области***, в которой указывается *sc-идентификаторы* всех ключевых *sc-элементов* специфицируемой *предметной области*, а также приводятся правила построения *основных sc-идентификаторов* для элементов всех *sc-классов* (понятий), исследуемых в рамках специфицируемой *предметной области*;
- ***дидактическую спецификацию предметной области***, в которой приводится дополнительная информация, предназначенная для того, чтобы пользователи и разработчики (инженеры знаний), которые используют или совершенствуют специфицируемую *предметную область* и ее *онтологию*, могли быстрее усвоить их особенности (см. *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*)
- ***проектную спецификацию предметной области и соответствующей ей онтологии***, в которой приводится информация об истории эволюции этой *предметной области и онтологии*, а также о направлениях и планах организации дальнейшего их развития.

Более подробно о *предметных областях* см. в *§ 2.5.4. Формализация понятия предметной области*, а более детальное рассмотрение формальных *онтологий*, представленных в SC-коде см. в *§ 2.5.5. Формализация понятия онтологии*.

Онтологическая формализация *базовой денотационной семантики SC-кода* трактуется нами как *формальная онтология*, представленная в *SC-коде* и описывающая детонационную семантику *семантически корректных sc-конструкций*. Указанную *формальную онтологию* будем называть ***Базовой денотационной семантикой SC-кода***. Для того, чтобы уточнить *предметную область*, специфицируемую этой *онтологией*, введем следующие понятия:

синонимия sc-элементов
≔ [бинарное ориентированное *отношение эквивалентности*, каждая пара которого связывает два *sc-элемента*, обозначающие одну и ту же сущность*]
⇒ *примечание**:
[Синонимия двух *sc-элементов* возможна только в том случае, если эти *sc-элементы* хранятся в *sc-памяти* (входят в состав *баз знаний*) разных *ostis-систем*. В рамках каждой *ostis-системы* синонимичные *sc-элементы* совпадают (отождествляются, склеиваются, считаются одним и тем же *sc-элементом*).]

отношение эквивалентности
⇐ *ключевое понятие**:
§ 2.4.2. Формальная онтология связок и отношений

sc-память

база знаний ostis-системы

ostis-система
⇒ *разбиение**:
{• *индивидуальная ostis-система*
• *коллективная ostis-система*

}

[*sc-конструкция*]
⇐ *часто используемый sc-идентификатор**:
sc-множество
:= [информационная конструкция, представляющая собой множество *sc-элементов*]
⊃ *sc-текст*
:= [текст SC-кода]
:= [*sc-конструкция*, являющаяся семантически корректной по отношению к *Базовой денотационной семантике SC-кода*]
:= [*sc-конструкция*, удовлетворяющая (соответствующая) правилам *Базовой денотационной семантики SC-кода*]
:= *часто используемый sc-идентификатор**:
[*SC-код*]
∈ *имя собственное*
:= [Класс (Множество всевозможных) sc-текстов]
⊃ *sc-знание*

sc-знание
:= [*sc-текст*, являющийся либо фрагментом (подструктурой) соответствующей *предметной области*, либо *высказыванием*, описывающим некоторое свойство (в частности, некоторую закономерность) этой *предметной области*]
:= [знание, представленное в *SC-коде*]
:= [*sc-текст*, обладающий истинным значением по отношению к соответствующей *предметной области*]
⊂ *связная sc-конструкция*
⇒ *примечание**:
[Разные *sc-знания* могут противоречить друг другу, то есть отражать разные точки зрения на некоторую *предметную область*, но любое *sc-знание* должно быть *sc-текстом*, то есть не должно противоречить правилам *Базовой денотационной семантики SC-кода*]

интеграция sc-конструкций*
:= [объединение sc-конструкций*]
:= [объединение sc-множеств*]
⇒ *примечание**:
[При интеграции sc-конструкций sc-элементы, обозначающие одну и ту же сущность, то есть синонимичные sc-элементы, считаются одинаковыми (совпадающими, тождественными) и, следовательно, должны склеиваться (отождествляться)]

SC-пространство
:= [Результат интеграции всевозможных sc-конструкций, *семантически корректных* по отношению к *Базовой денотационной семантики SC-кода*]
:= [Предметная область, специфицируемая (описываемая) *Базовой денотационной семантикой SC-кода*, которая является формальной онтологией, представленной средствами SC-кода]
:= [Результат интеграции всевозможных sc-текстов (текстов SC-кода)]
:= [Максимальный sc-текст]
:= [Текст SC-кода, включающий в себя всевозможные sc-тексты]
:= [Пространство sc-конструкций, семантически корректных по отношению к *Базовой денотационной семантике SC-кода*]
⇒ *примечание**:
- [Особенностью *SC-пространство* является то, что оно включает в себя и формальную онтологию, описывающую его свойства]
- [очевидно, что *SC-пространство* является бесконечным *sc-текстом*, то есть текстом, содержащим бесконечное количество *sc-элементов*. В частности, в состав *SC-пространства* входят все *sc-элементы*, являющиеся элементами всех *sc-множеств*, знаки которых входят в состав *SC-пространства*]
- [*SC-пространство* является "вместилищем" семантически корректных (по отношению к *Базовой денотационной семантике SC-кода*) частей баз знаний всевозможных ostis-систем и, в том числе, глобальной (объединенной) *Базы знаний Экосистемы OSTIS*. Подчеркнем при этом, что *Экосистема OSTIS* является примером распределенных иерархических *ostis-систем*]
- [Тот факт, что корректная (с точки зрения *Базовой денотационной семантики SC-кода*) часть базы знаний каждой *ostis-системы* входит в состав *SC-пространства*, позволяет трактовать описание соотношения

между текущим состоянием *базы знаний ostis-системы* и *Sc-пространством* как описание того, что указанная *ostis-система* в текущий момент времени не знает. Например, *ostis-система* в некоторый момент времени может не знать (1) всех элементов некоторого конкретного конечного *sc-множества* (конечно sc-конструкции), (2) количества элементов указанного конечного *sc-множества*, (3) какому подклассу заданного *sc-класса* принадлежит указанный элемент этого *sc-класса* и так далее]

- [В *памяти ostis-системы* каждый *sc-элемент* считается в рамках этой памяти временной сущностью (имеется в виду сам *sc-элемент*, а не обозначаемая им сущность), поскольку он появляется в *памяти ostis-системы* и удаляется из нее независимо от того, что он обозначает. В отличие от этого в *SC-пространстве* все sc-элементы считаются постоянными (постоянно присутствующими) в рамках этого пространства]

Базовая денотационная семантика SC-кода

:= [Онтология Базовой денотационной семантики SC-кода]

:= [Формальная *онтология*, представленная в *SC-коде* и являющаяся материнской *онтологией* (онтологией самого высокого уровня) для всех остальных *формальных онтологий*, представленных в *SC-коде*]

:= [Онтология SC-пространства]

:= [Описание (представление) системы *правил построения семантически корректируемых sc-конструкций*, удовлетворяющих требованиям Базовой денотационной семантики SC-кода]

∈ *sc-онтология*

:= [формальная онтология, представленная в SC-коде]

В состав *Базовой денотационной семантики SC-кода* включается:

- Приведенный выше текст *Пункт 2.2.1.1. Семантическая классификация sc-элементов по базовым признакам*
- Приведенный выше текст *Пункт 2.2.1.2. Уточнение смысла выделенных классов sc-элементов и связей между этими классами*
- Средства базовой семантической спецификации sc-элементов, рассмотренные в *Пункт 2.2.1.3. Структура базовой семантической спецификации sc-элемента*

Логическая онтология SC-пространства

⇐ *логическая онтология**:
Базовая денотационная семантика SC-пространства

⇒ *примечание**:
[Приведем некоторые правила, входящие в состав данной логической онтологии:

- Вторыми компонентами *sc-пар* константной парой принадлежности могут быть sc-элементы любого типа(в том числе, и *sc-переменные*), но первыми компонентами таких *sc-пар* могут быть только константные *sc-множества*
- Знак *sc-ситуации* связан с элементами этой ситуации *sc-парами* константной постоянной позитивной принадлежности. То есть позитивная принадлежность считается постоянной в рамках интервала времени существования соответствующей ситуации. В этом смысле ситуацию можно считать квазистатической
- Знак атомарной логической формулы связан со всеми элементами этой формулы *sc-парами* константной постоянной позитивной принадлежности, в том числе, и с теми элементами атомарной формулы, которые являются *sc-переменными*
- Из переменного *sc-множества* могут выходить только переменные *sc-пары принадлежности*
- Не существует sc-пар принадлежности выходящих из обозначений внешних сущностей и *sc-пар*
- и другие

]

Заключение к § 2.2.1.

SC-код может быть использован в качестве метаязыка для описания собственной денотационной семантики и синтаксиса.

Типология *sc-конструкций* с точки зрения *Денотационной семантики и Синтаксиса SC-кода* выглядит следующим образом

sc-множество

:= [*sc-конструкция*]

:= [информационная конструкция, принадлежащая *SC-коду*]

⊃ *sc-структура*

⊃ *sc-текст*
:= *часто используемый sc-идентификатор**:
[*SC-код*]
∈ *имя собственное*
:= [синтаксически целостная и синтаксически корректная (правильно построенная) информационная конструкция SC-кода]
:= [Класс (Множество всевозможных) sc-текстов]
⊃ *sc-знание*
:= [семантически целостный и семантически корректный *sc-текст*, являющийся адекватным фрагментом соответствующей *предметной области* или ее спецификации (онтологии)]

Перечислим аспекты представления знаний в памяти *интеллектуальных компьютерных систем*, которые требуют особой аккуратности:
- константный и переменный характер денотационной семантики знаков, хранимых в памяти *интеллектуальных компьютерных систем*;
- динамический характер знаковых конструкций, хранимых в памяти, обусловленный либо выполняемыми в памяти *информационными процессами*, либо динамическим характером структур внешних объектов, описываемых этими знаковыми конструкциями;
- временный характер существования внешних описываемых объектов и временный характер существования различных конфигураций знаковых конструкций и даже самих знаков, хранимых в памяти;
- наличие информационного мусора (излишеств) в хранимых знаниях;
- неизвестность (отсутствие в памяти) востребованной информации различного вида, неполнота знаний, их недостаточность для решения актуальных задач;
- синтаксическая и семантическая некорректность, неадекватность и, в частности, противоречивость некоторых имеющихся знаний.

Последние из перечисленных аспектов представления знаний называют *не-факторами представления знаний* (см. *Нариньяни А.С.нФактоНиНРиВ-2000ст*).

§ 2.2.2. Синтаксис SC-кода

⇒ *подраздел**:
- *Пункт 2.2.2.1. Синтаксическое Ядро SC-кода*
- *Пункт 2.2.2.2. Уточнение понятия синтаксически корректной sc-конструкции*
- *Пункт 2.2.2.3. Синтаксические расширения Ядра SC-кода*

Введение в § 2.2.2.

SC-коду соответствует несколько синтаксических модификаций, каждая из которых задается:
- своим алфавитом, то есть семейством *синтаксически выделяемых классов sc-элементов*;
- своим способом представления (кодирования) *пар инцидентности sc-элементов*, связывающих *sc-элементы* между собой.

алфавит синтаксической модификации SC-кода
:= *пояснение**:
[семейство синтаксических меток, приписываемых *sc-элементам* в рамках соответствующей синтаксической модификации SC-кода и указывающих факт принадлежности *sc-элемента* соответствующему классу *sc-элементов* (*sc-классу*)]

Минимальный алфавит SC-кода^
⇒ *пояснение**:
[Если известен смысл выделяемых классов sc-элементов (*sc-классов*), каждый из которых в *sc-памяти* представлен константным *sc-элементом*, обозначающим этот *sc-класс*, то для "анализа" и понимания *sc-конструкций*, хранимых в *sc-памяти*, достаточно синтаксически выделить только Класс *константных постоянных позитивных sc-пар принадлежности*, с помощью которых каждый *sc-элемент* будет явно соединяться с *sc-элементами*, обозначающими те *sc-классы*, которым этот *sc-элемент* принадлежит. Очевидно, что таким явным способом выделить указанные *константные постоянные позитивные sc-пар принадлежности* с помощью самих этих sc-пар невозможно.

Таким образом, любой класс *sc-элементов* можно выделить явно путем:

- включения в состав базы знаний *sc-элемента*, являющегося знаком этого класса sc-элементов (*sc-класса*);
- проведения *постоянных позитивных sc-пар принадлежности* во все *sc-элементы*, являющиеся элементами выделяемого *sc-класса* и хранимые (присутствующие) в текущем состоянии *sc-памяти*.

Минимальным алфавитом SC-кода является *Класс константных постоянных позитивных sc-пар принадлежности* и Класс всех остальных *sc-элементов* (задаваемых по умолчанию)]

⇒ *примечание**:
[Тем не менее, если учитывать особенности обработки в *sc-памяти* разных классов *sc-элементов*, целесообразно сделать расширение *Минимального алфавита SC-кода* и, соответственно, ввести понятие ***Синтаксического Ядра SC-кода***, а также целого семейства *синтаксических расширений Ядра SC-кода*]

Пункт 2.2.2.1. Синтаксическое Ядро SC-кода

Синтаксическая структура линейных *информационных конструкций* (строк символов) задается:
- алфавитом используемых символов (элементарных, атомарных фрагментов информационных конструкций, каковыми, в частности, являются буквы), то есть семейством таких попарно непересекающихся классов синтаксически эквивалентных символов, для которых существует простая процедура, позволяющая для любого символа по его синтаксическим особенностям установить факт его принадлежности одному из указанных классов;
- бинарным ориентированным отношением, определяющим непосредственный порядок (последовательность) символов в строках символов.

Аналогично этому, синтаксическая структура *sc-конструкций* задается:
- семейством классов синтаксически эквивалентных *sc-элементов*, в каждый из которых входят *sc-элементы* с одинаковыми синтаксическими характеристиками или, условно говоря, с одинаковыми наборами синтаксических меток;
- двумя бинарными ориентированными *отношениями инцидентности sc-элементов*, заданными на множестве всех *sc-элементов*:
  - *Отношением инцидентности обозначений sc-пар с их компонентами*
  - *Отношением инцидентности обозначений ориентированных sc-пар с их вторыми компонентами*
    ⇒ *примечание**:
    [Данное отношение является подмножеством *Отношения инцидентности обозначений sc-пар с их компонентами*]

Отношение инцидентности обозначений sc-пар с их компонентами*
⇒ *часто используемый sc-идентификатор**:
- [пара инцидентности sc-элементов*]
- [*пара инцидентности обозначения sc-пары с ее компонентом**]

∈ *имя нарицательное*
⇒ *примечание**:
- [Каждая пара, принадлежащая данному отношению семантически трактуется как *обозначение sc-пары принадлежности*, но синтаксически оформляется не в виде самостоятельного *sc-элемента*, а в виде бинарной ориентированной связи между *sc-элементами*, что аналогично бинарным ориентированным связям, описывающим последовательность символов в строке символов. Заметим при этом, что конфигурация *sc-конструкций* в отличие от строк символов не является линейной. Заметим также, что уточнение семантической интерпретации пар инцидентности *sc-элементов* полностью определяется семантической типологией первых компонентов этих пар инцидентности, то есть семантической типологией *обозначений sc-пар*, являющихся первыми компонентами рассматриваемых пар инцидентности:
  - если указанное *обозначение sc-пары* является *sc-константой*, то соответствующая пара инцидентности трактуется как *пара константной принадлежности*;
  - если указанное *обозначение sc-пары* является *sc-переменной*, то соответствующая пара инцидентности трактуется как *пара переменной принадлежности*;
  - если указанное *обозначение sc-пары* является *обозначением постоянной сущности*, то соответствующая пара инцидентности трактуется как *пара постоянной принадлежности*;
  - если указанное *обозначение sc-пары* является *обозначением временной сущности*, то соответствующая пара инцидентности трактуется как *пара временной принадлежности*.

  ]

- [Подчеркнем, что первыми компонентами пар инцидентности *sc-элементов* всегда являются *обозначения sc-пар*, но вторыми компонентами пар инцидентности *sc-элементов* могут быть *sc-элементы* любого типа (в том числе, и *обозначения sc-пар*)]

⇒ *пояснение**:
[Каждая *sc-пара* (константная пара sc-элементов), каждая *переменная sc-пара* и каждое *обозначение sc-пары* связывается со своими элементами не явно вводимыми константными или переменными *sc-парами позитивной принадлежности*, а реализуемыми на "физическом" уровне связями (парами) инцидентности. Таким образом *пары инцидентности sc-элементов* — это специальным образом синтаксически выделенные константные или переменные *sc-пары позитивной принадлежности*, связывающие *обозначения sc-пар* с элементами этих пар. Соответственно этому синтаксические особенности имеют и все *обозначения sc-пар*, поскольку только из них могут выходить ориентированные *пары инцидентности*. Поэтому с синтаксической точки зрения *обозначения sc-пар* будем называть ***sc-коннекторами***, *обозначения неориентированных sc-пар* — ***sc-ребрами***, а *обозначения ориентированных sc-пар* — ***sc-дугами***. При этом из класса *пар инцидентности sc-элементов* выделим подкласс пар, связывающих обозначения sc-дуг с теми sc-элементами, в которые эти дуги входят. Такую пару инцидентности будем называть ***парой инцидентности входящей sc-дуги***.]

Какие *семантически выделяемые классы sc-элементов* следует также рассматривать, как и ***синтаксически выделяемые классы sc-элементов***:

- *обозначение неориентированной sc-пары*
  ⇒ *примечание**:
  [Каждый *sc-элемент*, принадлежащий этому классу, связывается с элементами обозначаемого им множества]
- *обозначение ориентированной sc-пары не являющейся двумя парами инцидентности постоянной позитивной sc-парой принадлежности*
  ⇒ *примечание**:
  [Каждый *sc-элемент*, принадлежащий этому классу, связывается с элементами обозначаемого им множества:
  - одной парой инцидентности, связывающей *обозначение sc-пары* с ее компонентом;
  - одной парой инцидентности, связывающей *обозначение ориентированной sc-пары* с ее вторым компонентом

  ]
- *постоянная позитивная sc-пара принадлежности*
  ⇒ *примечание**:
  [Каждый элемент этого класса, как и любое другое *обозначение ориентированной sc-пары*, является первым компонентом *пары инцидентности обозначения sc-пары с ее компонентом*, а также первым компонентом *пары инцидентности обозначения ориентированной sc-пары с ее вторым компонентом*]
- *файл*
  ≔ [знак файла]
  ⇒ *примечание**:
  [Для *sc-элементов* этого класса необходимо на "синтаксическом" уровне обеспечить возможность связи этого *sc-элемента* с обозначаемым им *файлом*, хранимым в *файловой памяти* этой же *ostis-системы*]
- sc-элемент, не являющийся ни знаком файла, ни обозначением sc-пары
  ⇒ *примечание**:
  [Данный класс включает в себя:
  - *обозначение sc-синглетона*;
  - *обозначение sc-связки, не являющейся ни синглетоном, ни парой*;
  - *обозначение sc-класса*;
  - *обозначение sc-структуры*;
  - *обозначение внешней сущности, не являющейся файлом*.

  ]

sc-ребро
≔ [Класс *sc-элементов*, имеющих в рамках *Ядра SC-кода* синтаксическую метку *обозначений неориентированных sc-пар*]
≔ [Синтаксическая метка *обозначения неориентированной sc-пары*, используемая в рамках *Ядра SC-кода*]
∈ *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*

sc-дуга общего вида
≔

[Класс *sc-элементов*, имеющих в рамках *Ядра SC-кода* синтаксическую метку *обозначений ориентированных sc-пар, не являющихся постоянными позитивными sc-парами принадлежности*]

базовая sc-дуга
:= [Класс *sc-элементов*, имеющих в рамках *Ядра SC-кода* синтаксическую метку *постоянных позитивных sc-пар принадлежности*]

sc-узел, являющийся знаком файла
:= [*sc-элементов*, имеющий в рамках *Ядра SC-кода* синтаксическую метку *sc-элементов*, являющихся знаками *файлов*]

sc-узел, не являющийся знаком файла
:= [*sc-узел*, имеющий в рамках *Ядра SC-кода* синтаксическую метку *sc-элементов, не являющихся ни знаками файлов, ни обозначениями sc-пар*]

Перечисленное семейство синтаксически выделяемых классов *sc-элементов* составляет *Алфавит Ядра SC-кода^*.

Алфавит Ядра SC-кода^
= {• *sc-ребро*
• *sc-дуга общего вида*
• *базовая sc-дуга*
• *sc-узел, являющийся знаком файла*
• *sc-узел, не являющийся знаком файла*
}

Приведем синтаксическую классификацию sc-элементов в рамках Ядра SC-кода

sc-элемент
⇒ *разбиение**:
{• *sc-коннектор*
⇒ *разбиение**:
{• *sc-дуга*
⇒ *разбиение**:
{• *базовая sc-дуга*
• *sc-дуга общего вида*
}
• *sc-ребро*
}
• *sc-узел*
⇒ *разбиение**:
{• *sc-узел, являющийся знаком файла*
• *sc-узел, не являющийся знаком файла*
}
}

sc-узел
= (*sc-узел, являющийся знаком файла* ⋃ *sc-узел, не являющийся знаком файла*)

sc-дуга
= (*базовая sc-дуга* ⋃ *sc-дуга общего вида*)

sc-коннектор
= (*sc-дуга* ⋃ *sc-ребро*)

синтаксически выделяемый sc-класс
:= [класс *sc-элементов*, имеющих общий (одинаковый) синтаксический признак, который задается либо одной синтаксической меткой, каждая из которых семантически эквивалентна (синонимична) *sc-элементу*, обозначающему соответствующий синтаксически выделяемый *класс sc-элементов*, либо набором таких меток]

⇒ *примечание**:
- [Наличие у двух разных *sc-элементов* одной и той же синтаксической метки означает то, что оба эти *sc-элемента* принадлежат *sc-классу*, знаком которого является *sc-элемент*, семантически эквивалентный указанной метке]
- [Если *sc-элементу* приписывается несколько меток, то *синтаксически выделяемым sc-классом* является пересечение *sc-классов*, синтаксически выделяемых по каждой из этих меток]

:= *пояснение**:
[*sc-элемент*, обозначающий *sc-класс*, принадлежность которому может быть представлена либо с помощью *sc-пары постоянной позитивной принадлежности*, либо с помощью соответствующей метки, приписываемой этому *sc-элементу*, или набора таких меток]

⇒ *примечание**:
[Приписывание *sc-элементам* меток ускоряет проверку принадлежности sc-элементов соответствующим классам]

:= [sc-класс, каждому sc-элементу которого приписывается соответствующая этому sc-классу синтаксическая метка, которая является неявной (синтаксической) формой указания факта принадлежности указанного *sc-элемента* указанному *sc-классу*]

⇒ *разбиение**:
{• *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*
• *синтаксически выделяемый sc-класс в рамках расширения Ядра SC-кода*
}

⇒ *примечание**:
[Каждый *синтаксически выделяемый sc-класс* можно считать элементом алфавита соответствующей *Синтаксической модификации элементов SC-кода*. Но, в отличие от других языков, синтаксически выделяемые *sc-классы* большинства синтаксических модификаций SC-кода могут пересекаться. В отличие от этого, особенностью привычных нам языков является то, что каждый элементарный (атомарный) фрагмент информационных конструкций может иметь только одну метку, то есть может быть элементом только одного синтаксически выделяемого класса элементарных фрагментов.]

синтаксически выделяемый sc-класс в рамках Ядра SC-кода

:= [синтаксически выделяемый в рамках *Ядра SC-кода* класс sc-элементов]

:= [синтаксическая метка, приписываемая sc-элементам в рамках *Ядра SC-кода*]

:= [синтаксическая метка sc-элементов, выделяющая в рамках *Ядра SC-кода* соответствующий класс синтаксически эквивалентных sc-элементов]

:= [класс синтаксически эквивалентных sc-элементов в рамках *Ядра SC-кода*]

:= [синтаксический тип sc-элементов, выделяемый в рамках *Ядра SC-кода*]

⇒ *примечание**:
[В различных синтаксических *расширениях Ядра SC-кода* синтаксически выделяемые sc-классы могут пересекаться. То есть sc-элемент может принадлежать сразу несколькими синтаксически выделяемым *sc-классам*.]

Формирование, семейства *синтаксически выделяемых sc-классов* (то есть семейства синтаксических меток, приписываемых sc-элементам) может осуществляться на основе синтаксической классификации *sc-элементов* по различным признакам. Желательно при этом, чтобы такая синтаксическая классификация *sc-элементов* была согласована с семантической классификацией sc-элементов, которая рассмотрена в *Пункт 2.2.1.1. Семантическая классификация sc-элементов по базовым признакам*. Другими словами, каждый *синтаксически выделяемый sc-класс* (каждая синтаксическая метка) должен иметь четкую семантическую интерпретацию, то есть должен быть одновременно и *семантически выделяемым sc-классом*.

следует отличать*

∋ {• *синтаксически выделяемый sc-класс в рамках Ядра SC-кода*
• *синтаксически выделяемый sc-класс в рамках расширения Ядра SC-кода*
• *sc-класс*
:= [Класс sc-элементов, выделяемый (задаваемый) явно с помощью sc-конструкции, состоящей (1) из sc-элемента, являющего знаком этого класса и (2) из константных постоянных позитивных sc-пар принадлежности, соединяющих указанный знак выделяемого класса sc-элементов со всеми sc-элементами, принадлежащими этому классу и хранимыми в текущем состоянии sc-памяти]
}

∋ {• денотационную семантику каждого *sc-элемента,* то есть соотношение между *sc-элементом* и тем, что он обозначает (его денотатом) и соответствующую семантическую классификацию всего множества *sc-элементов*

- синтаксический тип каждого *sc-элемента,* то есть синтаксическую метку (значение синтаксического признака-параметра), приписываемую каждому *sc-элементу* и соответствующую синтаксическую классификацию всего множества sc-элементов (Алфавит SC-кода^))

}

Пункт 2.2.2.2. Уточнение понятия синтаксически корректной sc-конструкции

Синтаксис SC-кода
:= [Онтология синтаксиса SC-кода]
:= [Описание правил построения *синтаксически корректных sc-конструкций*]
:= [Описание требований, предъявляемых к *синтаксически корректным sc-конструкциям*]
∈ *sc-онтология*

sc-множество
:= *часто используемый sc-идентификатор**:
[*sc-конструкция*]
:= [множество *sc-элементов*, которые могут быть (но не обязательно) связаны между собой бинарными ориентированными *парами инцидентности*, каждая из которых связывает некоторый *sc-коннектор* с *sc-элементами*, которые связываются этим *sc-коннектором*]
:= [информационная конструкция, каждый элемент (атомарный фрагмент) которой входит в состав некоторого текста, принадлежащего *SC-коду*, но при этом *конфигурация* всей указанной информационной конструкции не всегда позволяет считать ее *текстом SC-кода*, удовлетворяющим целому ряду синтаксических и семантических требований]
⇒ *разбиение**:
{• *синтаксически корректная sc-конструкция*
• *синтаксически некорректная sc-конструкция*
}

синтаксически корректная sc-конструкция
:= [синтаксически правильно построенная *sc-конструкция*]

правило построения синтаксически корректных sc-конструкций
:= [синтаксическое правило SC-кода]
:= [требование (одно из требований), предъявляемое к *синтаксически корректным sc-конструкциям*]
∋ **{**• Каждая *sc-пара принадлежности*, связывающая *sc-элемент*, обозначающий пару *sc-элементов*, с компонентом этой пары (то есть с *sc-элементом*, связываемым этой *sc-парой* с другими *sc-элементом*) синтаксически "преобразуется" из *sc-элемента*, обозначающего *sc-пару принадлежности* в *пару инцидентности sc-элементов*, которая синтаксически уже не является *sc-элементом*
• Поскольку для каждого о*бозначения sc-пары* осуществляется *синтаксическая* замена *sc-пар принадлежности* их элементов на *пары инцидентности* этих элементов соответствующее синтаксическое преобразование происходит и с самими *обозначениями sc-пар* — они "превращаются" в *sc-коннекторы*. Соответственно этому *обозначения неориентированных sc-пар* "преобразуется" в *sc-ребра*, а *обозначения ориентированных sc-па*р — в *sc-дуги*
}

синтаксически некорректная sc-конструкция
:= [*sc-конструкция*, содержащая одну или несколько синтаксических ошибок]
⊃ *минимальная синтаксически некорректная sc-конструкция*
:= [*sc-конструкция*, не содержащая подструктур, являющихся *синтаксически некорректными sc-конструкциями*]
⇒ *примечание**:
[Каждой *минимальной синтаксически некорректной sc-конструкции* ставится в соответствие одно из синтаксических правил *SC-кода*, которому указанная *sc-конструкция* противоречит]
⇒ *примечание**:
[Строго говоря, *синтаксически некорректные sc-конструкции* не являются *sc-текстами*, то есть информационными конструкциями, принадлежащими *SC-коду*]

⇐ *невключение**:
sc-текст
≔ [*sc-конструкция* принадлежащая SC-коду]

Пункт 2.2.2.3. Синтаксические расширения Ядра SC-кода

Способ кодирования *sc-конструкций* в различных вариантах реализации *sc-памяти* может быть различным. То есть каждому варианту реализации *sc-памяти* может соответствовать своя *синтаксическая модификация SC-кода*. При этом она может касаться не только способа представления меток *sc-элементов*, но и представления (кодирования) *отношений инцидентности sc-элементов*. В любом случае каждая такая модификация должна быть четко описана.

Расширение семейства синтаксически выделяемых классов sc-элементов целесообразно:
- Для того, чтобы ускорить определение семантического типа каждого *sc-элемента* в ходе обработки *sc-конструкций*
- Чтобы быстро уточнить содержание *базовой спецификации sc-элемента* (что необходимо о нем знать, чтобы с ним эффективно работать). В частности, необходимо знать то, что о нем не известно.

Но при этом число меток, приписываемых sc-элементу должно быть минимизировано, то есть эти метки должны быть информативными.

Поскольку *SC-код* является языком внутреннего представления информации в *sc-памяти* ostis-системы, *Синтаксис SC-кода* является уточнением формы такого представления и, как следствие уточнением того, как устроена ***sc-память ostis-систем***. Поскольку хранимая в *sc-памяти* информационная конструкция представляет собой множество *sc-элементов*, можно ввести понятие ***ячейки sc-памяти***, каждая из которых обеспечивает хранение одного из *sc-элементов*.

ячейка sc-памяти
⇒ *пояснение**:
[фрагмент *sc-памяти*, в котором может храниться один *sc-элемент* (точнее, основная информация об этом sc-элементе) и который должен содержать:
- набор синтаксических меток, приписываемых хранимому *sc-элементу*;
- уникальный (взаимнооднозначный) идентификатор хранимого *sc-элемента* (аналог адреса ячейки в адресной памяти);
- связи хранимого *sc-элемента* со смежными *sc-элементами* (пары инцидентности);
- ссылка на хранимый файл, если хранимый *sc-элемент* является *знаком файла*, хранимого в файловой памяти этой же индивидуальной *ostis-системы*.

]

Формы представления меток *sc-элементов* могут быть разными:
- приписывание *sc-идентификатора* того *sc-класса* которому принадлежит данный *sc-элемент*;
- формирование вектора признаков в некотором пространстве признаков (Каждому признаку ставится в соответствии свое поле, в которое записывается соответствующее значение признака — в качестве этого значения тоже можно использовать sc-идентификатор соответствующего значения этого признака).

§ 2.2.3. Смысловое пространство ostis-систем

⇒ *ключевое понятие**:
- *обобщенная sc-связка*
- *обобщенное sc-отношение*
- *бинарное sc-отношение*
- *слотовое sc-отношение*
- *sc-структура**
- *элементарно представленный элемент'*
- *полносвязно представленный элемент'*
- *полностью представленный элемент'*
- *sc-связка'*
- *sc-отношение'*
- *sc-класс'*

- *сущностное замыкание**
- *содержательное замыкание**
- *sc-отношение сходства по слотовым отношениям**
- *sc-отношение семантического сходства по слотовым отношениям**
- *связная sc-структура**
- *семантическое сходство sc-структур**
- *семантическое непрерывное сходство sc-структур**
- *ключевой запрос'*
- *минимальный ключевой запрос'*
- *полная семантическая окрестность элемента**
- *интроспективный ключевой элемент'*
- *топологическое пространство*
- *топологическое пространство замыкания инцидентности коннекторов*
- *топологическое пространство синтаксического замыкания*
- *топологическое пространство сущностного замыкания*
- *топологическое пространство содержательного замыкания*
- *метрика*
- *семантическая метрика*
- *метрическое пространство*
- *метрическое конечное синтаксическое пространство*
- *метрическое конечное семантическое пространство*
- *псевдометрика*
- *псевдометрическое пространство*
- *псевдометрическое конечное семантическое пространство*

Понятие ***SC-пространства*** наряду с понятием ***SC-кода*** является необходимым для уточнения и формализации понятия смысла *информационных конструкций* и в *унификации смыслового представления информации*. В ***SC-пространстве*** можно выделять структуры, связанные как с синтаксическими свойствами текстов ***SC-кода***, так и с его семантикой. В последнем случае речь можно вести о “смысловом пространстве”. Смысл *информационной конструкции*, в конечном счете, определяется (1) внутренними связями всех элементарных фрагментов этой конструкции и (2) ее внешними связями с элементами *Смыслового пространства* (ее положением в контексте). *Смысловое пространство* является результатом естественного становления знаний в процессе их интеграции. Важнейшим достоинством ***SC-пространства*** является возможность уточнения таких понятий, как понятие аналогичности (сходства и отличия) различных описываемых "внешних" сущностей, аналогичности унифицированных *семантических сетей* (текстов ***SC-кода***), понятие семантической близости описываемых сущностей (в том числе, и текстов ***SC-кода***).

Следует отметить, что в силу абстрактности языков модели *унифицированного семантического представления знаний* и условности выбора меток элементов их текстов, нельзя исключить, что объединение двух произвольных текстов таких языков не будет текстом языка модели *унифицированного семантического представления знаний*. Чтобы избежать результатов подобных эклектических объединений с точки зрения синтаксиса или семантики, для абстрактных языков следует рассматривать множество “смысловых пространств”. Однако, для конкретных (реальных) языков может оказаться достаточным рассмотрение одного “смыслового пространства”. Далее рассмотрим:

- возможность перехода от sc-текстов к графовым структурам и от них к топологическому пространству;
- возможность перехода от sc-текстов к графовым структурам и от них к многообразию (топологическому пространству);
- возможность перехода от sc-текстов к графовым структурам и от них к метрическому пространству.

Чтобы исследовать структурные свойства ***SC-пространства***, можно использовать уже разработанные модели пространств и связь их известными топологическими моделями, например, такими как *графы*. При этом изначально не будем принимать в расчет динамические особенности, связанные с обработкой знаний, однако позже будет показано, что учет динамики в процессах обработки и при становлении знаний является необходимым для вычисления семантической метрики, являющейся одним из определяющих признаков знаний. Обратимся к исследованию структурно-топологических свойств пространства.

Структурно-топологические свойства могут свидетельствовать о синтаксических или семантических зависимостях обозначений текстов языка, позволяющих упростить работу со структурами за счет перехода к более простым структурам на уровнях управления данными или знаниями в характерных задачах управления для *библиотеки многократно используемых компонентов ostis-систем* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*). На множестве элементов, образующих ***SC-пространство***, можно изучать топологические свойства и рассматривать ***SC-пространство*** как топологическое пространство. Следует заметить, что, несмотря на то, что ***SC-код*** ориентирован на смысловое представление знаний, в силу наличия *не-факторов*, не все смыслы могут быть представлены в некоторый момент времени

и не будет известна структура соответствующего представления. Поэтому структурно-топологические свойства текстов *языка представления знаний* скорее определяют синтаксическое пространство, нежели семантическое (смысловое). Хотя оба могут приближаться друг к другу по мере устранения неопределенностей, вызванных *нефакторами*.

Рассмотрим следующие виды *топологических пространств*:
- *топологическое пространство замыкания инцидентности коннекторов*;
- *топологическое пространство синтаксического замыкания*;
- *топологическое пространство сущностного замыкания*;
- *топологическое пространство содержательного замыкания*.

топологическое пространство
⇒ *пояснение**:
[*топологическое пространство — множество* с определенным над ним *множеством* (семейством) (открытых) подмножеств, включая само *множество* и *пустое множество*. Для любого *подмножества* семейства результат объединения принадлежит *семейству множеств*, а для любого конечного *подмножества семейства* результат пересечения также принадлежит *семейству множеств*. Дополнения множеств семейства до наибольшего из множеств называются *замкнутыми множествами*.**]**

Чтобы рассмотреть более детально некоторые виды *топологических пространств* введем следующие понятия.

обобщенная sc-связка
:= **[**непустое sc-множество**]**

обобщенное sc-отношение
:= **[**sc-множество непустых sc-множеств**]**
⇒ *пояснение**:
[обобщенное sc-отношение — sc-множество обобщенных sc-связок.**]**

бинарное sc-отношение
⇒ *пояснение**:
[Бинарное sc-отношение — sc-множество sc-пар (или обобщенных sc-связок, которым существуют две различные принадлежности sc-элементов или одного и того же sc-элемента).**]**

узловая sc-пара
⇒ *пояснение**:
[узловая sc-пара — sc-пара, которая не может быть обозначена sc-дугой принадлежности (позитивной, негативной или нечеткой).**]**

явление принадлежности
⇒ *пояснение**:
[явление принадлежности — множество явлений, каждое из которых является слотовым sc-отношением, которому постоянно непринадлежат sc-дуги постоянной непринадлежности.**]**

становление*
⇒ *пояснение**:
[становление* — бинарное sc-отношение между событиями (состояниями) или явлениями.**]**

непосредственно прежде'
⇒ *первый домен**:
*становление**
⇒ *второй домен**:
установленное событие или явление

непосредственно после'
⇒ *первый домен**:
*становление**
⇒ *второй домен**:
устанавливающее событие или явление

продолжительность*

⇒ *пояснение**:
[продолжительность* — транзитивное замыкание sc-отношения становления.]

раньше'

⇒ *первый домен**:
*продолжительность**
⇒ *второй домен**:
раннее событие или явление

позже'

⇒ *первый домен**:
*продолжительность**
⇒ *второй домен**:
позднее событие или явление

слотовое sc-отношение

⇒ *пояснение**:
[Слотовое sc-отношение — бинарное sc-отношение (sc-множество (ориентированных) sc-пар), элементы которого не являются узловыми sc-парами.]

sc-структура*

⇒ *пояснение**:
[sc-структура* — sc-множество, в котором есть непустое sc-подмножество-носитель (множество первичных элементов sc-структуры*).]

sc-структура'

⇒ *первый домен**:
*sc-структура**
⇒ *второй домен**:
непустое sc-множество

носитель sc-структуры'

⇒ *первый домен**:
*sc-структура**
⇒ *второй домен**:
непустое sc-множество

элементарно представленное sc-множество'

:= [элементарно представленный элемент']
⇒ *пояснение**:
[Элементарно представленный элемент'— элемент sc-структуры*, sc-множество, все элементы которого являются элементами sc-структуры*.]

полносвязно представленное sc-множество'

:= [полносвязно представленный элемент']
⇒ *пояснение**:
[полносвязно представленный элемент'— элемент sc-структуры*, sc-множество, все элементы и все принадлежности которому являются элементами sc-структуры*, или sc-дуга, являющаяся элементарно представленным элементом'этой sc-структуры*.]

полностью представленное sc-множество'

:= [полностью представленный элемент']
⇒ *пояснение**:
[Полностью представленный элемент'— полносвязно представленный элемент'sc-структуры*, с любым элементом, не являющимся sc-дугой, выходящей из него, связанный принадлежащей этой sc-структуре* sc-дугой принадлежности или sc-дугой непринадлежности.]

sc-связка'

⇒ *пояснение**:

[sc-связка'— полносвязно представленный элемент'sc-структуры*, принадлежащий sc-отношению'этой sc-структуры*, являющийся sc-связкой.]

sc-отношение'

⇒ *пояснение**:

[sc-отношение'— полносвязно представленный элемент'sc-структуры*, sc-отношение, все элементы которого являются sc-связками'этой sc-структуры*.]

sc-класс'

⇒ *пояснение**:

[sc-класс'— полносвязно представленный элемент'sc-структуры*, все элементы которого являются элементами sc-структуры*, не являющийся ни sc-отношением', ни sc-связкой'этой sc-структуры*.]

сущностное замыкание*

⇒ *пояснение**:

[Сущностное замыкание* — наименьшее надмножество* (структура*), в котором каждый элемент является элементарно представленным'.]

сущностное замыкание'

⇒ *первый домен**:

*сущностное замыкание**

⇒ *второй домен**:

сущностное замыкание

носитель сущностного замыкания'

⇒ *первый домен**:

*сущностное замыкание**

⇒ *второй домен**:

непустое sc-множество

содержательное замыкание*

⇒ *пояснение**:

[содержательное замыкание* — наименьшее надмножество* (структура*), в котором каждый элемент является полносвязно представленным']

содержательное замыкание'

⇒ *первый домен**:

*содержательное замыкание**

⇒ *второй домен**:

содержательное замыкание

носитель содержательного замыкания'

⇒ *первый домен**:

*содержательное замыкание**

⇒ *второй домен**:

непустое sc-множество

sc-отношение сходства по слотовым отношениям*

⇒ *пояснение**:

[sc-отношение сходства по слотовым sc-отношениям* — sc-отношение, являющееся рефлексивным по этим слотовым отношениям, то есть для любого элемента, входящего в связку этого sc-отношения под одним из слотовых sc-отношений, найдется связка этого sc-отношения, в которую он входит под каждым из этих слотовых sc-отношений.]

sc-отношение сходства по слотовым отношениям'
⇒ *первый домен**:
*sc-отношение сходства по слотовым отношениям**
⇒ *второй домен**:
sc-отношение сходства по слотовым отношениям

слотовые отношения сходства sc-отношения'
⇒ *первый домен**:
*sc-отношение сходства по слотовым отношениям**
⇒ *второй домен**:
слотовые отношения сходства sc-отношения

sc-отношение семантического сходства по слотовым отношениям*
⇒ *пояснение**:
[sc-отношение семантического сходства по слотовым отношениям* — sc-отношение сходства по слотовым sc-отношениям* si и sj, в котором каждый элемент под слотовым sc-отношением si, может быть преобразован к элементу синтаксического типа элемента под слотовым sc-отношением sj; два инцидентных sc-элемента под слотовым sc-отношением si, в рамках этого sc-отношения семантического сходства соответствуют инцидентным элементам соответственно под слотовым sc-отношением sj.]

sc-отношение семантического сходства по слотовым отношениям'
⇒ *первый домен**:
*sc-отношение семантического сходства по слотовым отношениям**
⇒ *второй домен**:
sc-отношение семантического сходства по слотовым отношениям

слотовые отношения семантического сходства sc-отношения'
⇒ *первый домен**:
*sc-отношение семантического сходства по слотовым отношениям**
⇒ *второй домен**:
слотовые отношения семантического сходства sc-отношения

связная sc-структура*
⇒ *пояснение**:
[Связная sc-структура* — sc-структура*, являющаяся связной.]

связная sc-структура'
⇒ *первый домен**:
*связная sc-структура**
⇒ *второй домен**:
связное непустое sc-множество

носитель связной sc-структуры'
⇒ *первый домен**:
*связная sc-структура**
⇒ *второй домен**:
непустое sc-множество

семантическое сходство sc-структур*
:= [семантическое подобие sc-структур*]
⇒ *пояснение**:
[Семантическое сходство sc-структур* — связывает sc-множество sc-структур* с sc-структурой* sc-отношением семантического сходства по слотовым sc-отношениям si, sj так, что для каждой sc-структуры* из sc-множества найдется ее элемент и связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением si, а под слотовым sc-отношением sj входит элемент sc-структуры*, также для каждого элемента sc-структуры найдется связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением sj, а под слотовым sc-отношением si входит элемент sc-структуры* из sc-множества.]

sc-отношение семантического сходства sc-структур'
⇒ *первый домен**:
*семантическое сходство sc-структур**
⇒ *второй домен**:
*sc-отношение семантического сходства по слотовым отношениям**

семантическое сходство sc-структур'
⇒ *первый домен**:
*семантическое сходство sc-структур**
⇒ *второй домен**:
*sc-структура семантического сходства sc-структур**

sc-структура семантического сходства sc-структур'
⇒ *первый домен**:
*sc-структура семантического сходства sc-структур**
⇒ *второй домен**:
sc-структура семантического сходства sc-структур

множество семантически сходных sc-структур'
⇒ *первый домен**:
*sc-структура семантического сходства sc-структур**
⇒ *второй домен**:
множество семантически сходных sc-структур

семантическое непрерывное сходство sc-структур*
:= [семантическое непрерывное подобие sc-структур*]
⇒ *пояснение**:
[Семантическое непрерывное сходство sc-структур* — связывает sc-множество sc-структур* со связной sc-структурой* sc-отношением семантического сходства по слотовым sc-отношениям si, sj так, что для каждой sc-структуры* из sc-множества найдется ее элемент и связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением si, а под слотовым sc-отношением sj входит элемент связной sc-структуры*, также для каждого элемента связной sc-структуры найдется связка этого sc-отношения сходства, в которую он входит под слотовым sc-отношением sj, а под слотовым sc-отношением si входит элемент sc-структуры* из sc-множества.]

sc-отношение семантического непрерывного сходства sc-структур'
⇒ *первый домен**:
*семантическое непрерывное сходство sc-структур**
⇒ *второй домен**:
*sc-отношение семантического непрерывного сходства по слотовым отношениям**

семантическое непрерывное сходство sc-структур'
⇒ *первый домен**:
*семантическое непрерывное сходство sc-структур**
⇒ *второй домен**:
*sc-структура семантического непрерывного сходства sc-структур**

sc-структура семантического непрерывного сходства sc-структур'
⇒ *первый домен**:
*sc-структура семантического непрерывного сходства sc-структур**
⇒ *второй домен**:
sc-структура семантического непрерывного сходства sc-структур

множество семантически непрерывно сходных sc-структур'
⇒ *первый домен**:
*sc-структура семантического непрерывного сходства sc-структур**
⇒ *второй домен**:
множество семантически непрерывно сходных sc-структур

ключевой запрос'
⇒ *первый домен**:
*ключевой запрос**
⇒ *второй домен**:
ключевой запрос
⇒ *пояснение**:
[Ключевой запрос'— поисковый-проверочный запрос (от одного известного элемента), который выполняется хотя бы от одного элемента и не выполняется хотя бы от одного элемента.]

элемент ключевого запроса'
⇒ *первый домен**:
*ключевой запрос**
⇒ *второй домен**:
элемент ключевого запроса

минимальный ключевой запрос'
⊂ *ключевой запрос'*
⇒ *пояснение**:
[Минимальный ключевой запрос — ключевой запрос, который находит sc-подмножества множеств элементов, находимых всеми другими ключевыми запросами, которые имеют те же области известных элементов выполнимости и невыполнимости.]

элемент минимального ключевого запроса'
⇒ *первый домен**:
*минимальный ключевой запрос**
⇒ *второй домен**:
элемент минимального ключевого запроса

полная семантическая окрестность элемента*
⇒ *пояснение**:
[Полная семантическая окрестность элемента* — все элементы, находимые выполнимыми минимальными ключевыми запросами от этого элемента (с учетом дизъюнктивного поиска и отрицания поиска).]

полная семантическая окрестность элемента'
⇒ *первый домен**:
*полная семантическая окрестность элемента**
⇒ *второй домен**:
полная семантическая окрестность элемента

элемент полной семантической окрестности'
⇒ *первый домен**:
*полная семантическая окрестность элемента**
⇒ *второй домен**:
элемент полной семантической окрестности

интроспективный ключевой элемент'
⇒ *пояснение**:
[интроспективный (базовый) ключевой элемент — элемент множества (из класса наименьших таких множеств) элементов такого, что любая полная семантическая окрестность любого элемента является sc-подмножеством объединения их полных семантических окрестностей.]

топологическое пространство замыкания инцидентности коннекторов
⇒ *пояснение**:
[Топологическое пространство замыкания инцидентности коннекторов на множестве sc-элементов — топологическое пространство, все замкнутые множества которого содержат все sc-элементы этого множества, до которых есть маршрут по ориентированным связкам отношения инцидентности коннекторов.]
⇒ *примечание**:

[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. Прагматика рассмотрения таких пространств обуславливается операциями удаления sc-элементов и коннекторов, которым они инцидентны. Удаление sc-элемента требует удаления всех коннекторов, замыканию любой открытой окрестности которых он принадлежит.]

топологическое подпространство замыкания инцидентности коннекторов'

⇒ *первый домен**:
*включение топологических пространств замыкания инцидентности коннекторов**

⇒ *второй домен**:
топологическое пространство замыкания инцидентности коннекторов

топологическое надпространство замыкания инцидентности коннекторов'

⇒ *первый домен**:
*включение топологических пространств замыкания инцидентности коннекторов**

⇒ *второй домен**:
топологическое пространство замыкания инцидентности коннекторов

топологическое пространство синтаксического замыкания

⇒ *пояснение**:
[Топологическое пространство синтаксического замыкания на множестве sc-элементов — топологическое пространство, все замкнутые множества которого содержат все sc-элементы этого множества, до которых есть маршрут по ориентированным связкам отношения инцидентности.]

⇒ *примечание**:
[В общем случае не удовлетворяет аксиоме отделимости по Колмогорову. В качестве основы замкнутых множеств топологического пространства можно выделить синтаксическое замыкание, однако в силу возможности проведения дуг из любого sc-узла в любой в итоговом случае (в итоге процесса устранения не-факторов) такое пространство является тривиальным (антидискретным) пространством. Отношение объединения топологических пространств синтаксического замыкания алгебраически не замкнуто на множестве топологических пространств синтаксического замыкания. По той же причине для любого неполного топологического пространства синтаксического замыкания можно рассмотреть топологическое пространство синтаксического замыкания, носитель которого является надмножеством носителя первого и которое не сохраняет замкнутые множества. В этом смысле топология на основе синтаксического замыкания не является устойчивой относительно процессов становления знаний и ее рассмотрение прагматически не оправдывается. Топология полного же топологического пространства синтаксического замыкания антидискретна (тривиальна). Таким образом, у полного топологического пространства синтаксического замыкания все топологические подпространства синтаксического замыкания обладают антидискретной (тривиальной) топологией.]

топологическое пространство сущностного замыкания

⇒ *пояснение**:
[Топологическое пространство сущностного замыкания на множестве sc-элементов — топологическое пространство, все замкнутые множества которого являются сущностными замыканиями.]

⇒ *примечание**:
[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. В качестве носителя топологического (под)пространства можно выделить сущностное замыкание. Топологическое пространство сущностного замыкания сохраняет замкнутые множества любых топологических пространств сущностного замыкания, носитель которых является подмножеством его носителя и сущностным замыканием. Такие пространства образуют структуру топологических подпространств-топологических надпространств сущностного замыкания. Топология пространств в этой структуре разнообразна.]

топологическое подпространство сущностного замыкания'

⇒ *первый домен**:
*включение топологических пространств сущностного замыкания**

⇒ *второй домен**:
топологическое пространство сущностного замыкания

топологическое надпространство сущностного замыкания'

⇒ *первый домен**:
*включение топологических пространств сущностного замыкания**

⇒ *второй домен**:

топологическое пространство сущностного замыкания

топологическое пространство содержательного замыкания
⇒ *пояснение**:
[Топологическое пространство содержательного замыкания на множестве sc-элементов — топологическое пространство, все замкнутые множества которого являются содержательными замыканиями.]
⇒ *примечание**:
[В общем случае не удовлетворяет аксиоме отделимости по Тихонову. В качестве носителя топологического (под)пространства можно выделить содержательное замыкание. Топологическое пространство содержательного замыкания сохраняет замкнутые множества любых топологических пространств содержательного замыкания, носитель которых является подмножеством его носителя и содержательным замыканием. Такие пространства образуют структуру топологических подпространств-топологических надпространств содержательного замыкания. Топология пространств в этой структуре разнообразна.]

топологическое подпространство содержательного замыкания'
⇒ *первый домен**:
*включение топологических пространств содержательного замыкания**
⇒ *второй домен**:
топологическое пространство содержательного замыкания

топологическое надпространство содержательного замыкания'
⇒ *первый домен**:
*включение топологических пространств содержательного замыкания**
⇒ *второй домен**:
топологическое пространство содержательного замыкания

Возможен переход от sc-структур к многообразиям и топологическим пространствам путем сведения sc-структур к графовым структурам, подробно вопросы сведения sc-структур к графовым структурам и далее — к многообразиям и топологическим пространствам рассмотрены в работе *Ivashenko V.P.GenerPSRLaSS-2022art*.

Для более сложных структур таких, как полная семантическая окрестность, топологические свойства подлежат дальнейшему изучению.

Далее можно рассмотреть метрические пространства, в частности — конечные подпространства с полностью представленными sc-элементами.

метрика
⇒ *пояснение**:
[Метрика — функция двух аргументов, принимающая значения на (линейно) упорядоченном носителе группы, неотрицательна, равна нейтральному элменету (нулю) только при равенстве аргументов, симметрична, удовлетворяет неравенству треугольника.]

метрическое пространство
⇒ *пояснение**:
[Метрическое пространство — множество, с определенной на нем функцией двух аргументов, являющейся метрикой, принимающей значения на упорядоченном носителе группы.]

семантическая метрика
:= [семантическая близость]
⇒ *пояснение**:
[Семантическая метрика — метрика, определенная на знаках и выражающая количественно близость их значений.]

метрическое конечное синтаксическое пространство
⇒ *пояснение**:
[Метрическое конечное синтаксическое пространство SC-кода — метрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение метрики может быть определено через отношения инцидентности элементов без учета их семантического типа.]

метрическое конечное синтаксическое подпространство'
⇒ *первый домен**:
*включение метрических конечных синтаксических пространств**
⇒ *второй домен**:
метрическое конечное синтаксическое пространство

метрическое конечное синтаксическое надпространство'
⇒ *первый домен**:
*включение метрических конечных синтаксических пространств**
⇒ *второй домен**:
метрическое конечное синтаксическое пространство

метрическое конечное семантическое пространство
⇒ *пояснение**:
[Метрическое конечное семантическое пространство SC-кода — метрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение метрики не может быть определено через отношения инцидентности элементов без учета их семантического типа.**]**

метрическое конечное семантическое подпространство'
⇒ *первый домен**:
*включение метрических конечных семантических пространств**
⇒ *второй домен**:
метрическое конечное семантическое пространство

метрическое конечное семантическое надпространство'
⇒ *первый домен**:
*включение метрических конечных семантических пространств**
⇒ *второй домен**:
метрическое конечное семантическое пространство

Метрическое конечное синтаксическое пространство может быть построено (см. *Ivashenko V.P.GenerPSRLaSS-2022art*) в соответствии с моделью обработки строк и определениями метрики, которые даны в *Ivashenko V.P.StrinPMfKDS-2020art*.

псевдометрика
⇒ *пояснение**:
[Псевдометрика — функция двух аргументов, принимающая значения на (линейно) упорядоченном носителе группы, неотрицательна, симметрична, удовлетворяет неравенству треугольника.**]**

псевдометрическое пространство
⇒ *пояснение**:
[псевдометрическое пространство — множество, с определенной на нем функцией двух аргументов, являющейся псевдометрикой (см. *Collatz L.FunctAaNM-1966bk*), принимающей значения на упорядоченном носителе группы.**]**

псевдометрическое конечное семантическое пространство
⇒ *пояснение**:
[Псевдометрическое конечное семантическое пространство SC-кода — псевдометрическое пространство с конечным носителем, элементами которого являются обозначения (sc-элементы), а значение псевдометрики не может быть определено через отношения инцидентности элементов без учета их семантического типа.**]**

В силу неполноты выразительных средств для представления изменяющихся со временем знаний, отсутствия определенной пространственно-временной модели, наличия семантически неопределенных или слабоопределенных обозначений в текстах да и наличия недоопределенности самих текстов описанного в предыдущих разделах языка, на данном этапе в этом описании затруднительно предложить какую-либо модель метрического пространства для более сложных структур, учитывающих не-факторы, связанные с пространством-временем.

Некоторые такие модели были предложены в работе *Ivashenko V.P.GenerPSRLaSS-2022art*. Предложенные модели полагались на представление, способное выразить семантику переменных обозначений и операционную семантику

расширенными средствами алфавита. Для построения подобных моделей, кроме расширенных средств алфавита, предлагается полагаться на модели, описывающие процессы интеграции и становления знаний (см. *Ивашенко В.П.ОнтолМПВОСиЯвПОЗ-2017ст*), на средства спецификации знаний (см. *Ивашенко В.П.Модел иАИЗнООСС-2014дс*, *Ivashenko V.P.GenerPSRLaSS-2022art*), ориентированные на рассмотрение финитных структур, что позволяет перейти к рассмотрению сложных метрических соотношений в рамках метамодели смыслового пространства.

В разных науках исследователи затрагивали вопросы касающиеся смыслов и их размещения и взаимосвязи. Можно выделить следующие работы, которые соотносятся с тремя подходами: экстериорный подход (см. *Bohm D.tUndivUaOIoQT-1993bk* , *Bohm D.Whole atIO-2002bk*), интериорные подходы на основании количественных (см. *Налимов В.В..РеальНВМБ-1995кн*, *Налимов В.В.СпонтСВТСиСАЛ-1989кн*, *Налимов В.В.ВероятМЯоСЕиИЯ-1979кн*) и структурно-динамических признаков (см. *Мартынов В.В.Язык вПиВкПГС-2004кн*). В современных работах в технических науках (см. *Manin Y.I.SemanS-2016art*), возможно, наиболее близкими понятиями являются понятия, выражающие смысл термина “семантическое пространство” (интериорный подход). Общим во многих подходах к работе с “семантическим пространством” является рассмотрение словоформ или лексем (множеств словоформ) и их признаков. В литературе (см. *Manin Y.I.SemanS-2016art*) встречаются следующие подходы:

- подход на основе семантических осей и пространства признаков (бинарных $\{0,1\}^n$, монополярных $[0;1]^n$, биполярных $[-1;1]^n$);
- подход на основе семантических осей и нейронного кодирования места в поле смыслов (слова и словосочетания имеют области (подмножества) значений, связываясь другими частями речи как включением и пересечением, тексты соответствуют пути связанных областей, бинарное кодирование групп нейронов, распознающих смыслы);
- подход на основе модели “смысл-текст” (см. *Melcuk I.Langu fMtT-2016bk*) (отражение неполноты семантических шкал и анализ синтагм и поверхностно-синтаксической структуры);
- нейролингвистические данные отражает процессы синтеза и восприятия речи в нейронных сетях (сеть лексического синтеза), близка к модели “смысл-текст”;
- модели, построенные на основе статического анализа (корпусов) текстов (модель векторного пространства).

Статистический подход к обработке естественного языка противопоставляется интуиции и коммуникативному опыту ученых (см. *Manin Y.I.SemanS-2016art*). В основе подхода лежит семантическая статистическая гипотеза, что смысл слов (лексем) определяется контекстом использования (его статистическим образом) в языке (с коммуникативной структурой, см. *Manin Y.I.SemanS-2016art*).

Модель векторного пространства семантики (см. *Manin Y.I.SemanS-2016art*). Модель рассматривается для двух случаев: большого словаря ($N \leq M$) и задачи информационного поиска ($M \leq N$). M – размер словаря, N – количество контекстов.

На основе статистики строится матрица размерности $M \times N$ частот p_{ij} появления лексемы (слова) w_i в документе (контексте, подтексты, которые могут перекрываться) c_j.

$$x_{ij} = \max\left(\{0\} \cup \left\{\log\left(\frac{p_{ij}}{(\Sigma_j p_{ij}) * (\Sigma_i p_{ij})}\right)\right\}\right)$$

В знаменателе – оценки вероятности слова и контекста соответственно.

В случае невырожденной матрицы $r = N$ каждая такая матрица задает точку в грассманиане N-мерных подпространств M-мерного пространства ($N \leq M$).

В случае невырожденной матрицы $r = M$ каждая такая матрица задает точку в грассманиане M-мерных подпространств N-мерного пространства ($M \leq N$).

Каждый текст – точка в грассманиане (см. *Harris J.AlgebGaFC-1992bk*), соответствующем проективному пространству $P^{M-1} = Gr\left(\langle 1, M\rangle\right)$, относительно одного выделенного контекста. Для всех контекстов получая ориентированную N-ку, в соответствии с порядком контекстов в текстах, можно построить маршрут (путь), соединяя геодезическими соседние точки в N-ке. Для двух текстов T и T' это будут две ломанные, между которыми можно вычислить метрику Фреше (см. *Alt H.Compu tFDbTPC-1995art*), используя метрику Фубини-Штуди (см. *Study E.KürzeWiKG-1905art*) в P^{M-1}, для этого следует параметризовать пути $\Gamma\left(T\right)$ и $\Gamma\left(T'\right)$ через t ($\gamma \in \Gamma\left(T\right)^{[0;1]}$,$\gamma' \in \Gamma\left(T'\right)^{[0;1]}$):

$$\delta\left(\langle \Gamma\left(T\right), \Gamma\left(T'\right)\rangle\right) = \inf_{\gamma,\gamma'} \max_{t\in[0;1]} \left(\left\{d_{FS}\left(\langle \gamma\left(t\right), \gamma'\left(t\right)\rangle\right)\right\}\right).$$

Другой способ задать линейный порядок – это рассмотреть фильтрацию (флаг, флаговое многообразие, см. *Kostrikin A..LineaAaG-1997bk*) в $\mathbb{R}^M$, заданную расширяющимися контекстами. В итоге для текста получаем точки (флаги) во флаговом многообразии. Для флаговых многообразий тоже можно вычислить метрику Фубини-Штуди (см. *Study E.KürzeWiKG-1905art*).

Этот порядок соответствует временному измерению (процессу коммуникации во времени), что может быть существенным. Другой порядок может быть не зависимым от этого, например алфавитный или порядок в соответствии с законом Ципфа (см. *Lowe W.Towar aToSS-2001art*, *Manin Y.I.ZipfsLaLLPD-2014art*).

Таблица. Сравнение подходов к построению "семантических пространств"
=

	экстериорный (физический) подход	интериорный (абстрактный, логико-семиотический) подход на основании анализа количественных признаков (вероятностных (аддитивных) мер)	интериорный (абстрактный, логико-семиотический) подход на основании анализа структурно-динамических признаков
анализ когнитивных процессов (интроспекция)	+	-	?
адаптация	+	-	+
унификация	-	+	+

	семантические оси и пространства признаков	семантические оси и нейронное кодирование признаков	модель "смысл-текст"	нейролингвистическое кодирование	статистическая модель (модель векторного пространства семантики)
определенные семантические оси	+	+	-	-	-
динамическая (вычислительная) декомпозиция	-	+	-	+	-
анализ когнитивных процессов (интроспекция)	-	+	-	+	-
учет не-факторов (неполнота)	-	-	+	+	+

Вопросы соотнесения смыслов, их формализации, развития языков в пространстве и времени рассмотрены в работах В.В. Мартынова (см. *Мартынов В.В.Язык вПиВкПГС-2004кн*, *Мартынов В.В.вЦентрСЧ-2009кн*, *Гордей А.Н.ТеориАПАЗ-2014ст*).

Глава 2.3.

Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний

⇒ *автор**:
- *Садовский М. Е.*
- *Голенков В. В.*
- *Жмырко А. В.*
- *Ефимова А. А.*

⇒ *аннотация**:
[В главе рассматриваются понятия внешних и внутренних языков *интеллектуальных компьютерных систем нового поколения*. Описываются внешние языки представления знаний в рамках *Технологии OSTIS*, а именно *SCg-код*, *SCs-код* и *SCn-код*. Для каждого из внешних языков детально рассматривается его *синтаксис* и *денотационная семантика*.]

⇒ *подраздел**:
- *§ 2.3.1. Внешние идентификаторы sc-элементов — знаков, входящих в конструкции SC-кода*
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

⇒ *ключевое понятие**:
- *sc-идентификатор*
- *SCg-код*
- *SCs-код*
- *SCn-код*

⇒ *ключевое знание**:
- *Правила построения sc-идентификаторов*
- *Синтаксис и Денотационная семантика SCg-кода*
- *Синтаксис и Денотационная семантика SCs-кода*
- *Синтаксис и Денотационная семантика SCn-кода*

⇒ *библиография**:
- *Голенков В.В..СтандОТОП-2021кн*
- *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*
- *МетасOSTIS-2022эл*
- *Zhmyrko A.Famil oELoNGCSCttLoISRoK-2022art*
- *SoftwIoSLP-el*
- *OSTISSR-el*
- *SoftwIoStST-el*

Введение в Главу 2.3.

Для эксплуатации *интеллектуальных компьютерных систем нового поколения* кроме способа абстрактного внутреннего представления *баз знаний* требуются способы внешнего изображения абстрактных текстов, удобных для пользователей и используемых при оформлении исходных текстов *баз знаний* указанных *интеллектуальных компьютерных систем* и исходных текстов фрагментов этих *баз знаний*, а также используемых для отображения пользователям различных фрагментов *баз знаний* по пользовательским запросам(см. *Голенков В.В..СтандОТОП-2021кн*).

В рамках данной главы рассматриваются правила идентификации *sc-элементов*, а также *внешние языки ostis-систем* (*SCg-код*, *SCs-код*, *SCn-код*), которые являются различными вариантами внешнего представления текстов *SC-кода*(см. Главу 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код

(Semantic Computer Code)). Такие языки являются универсальными и, следовательно, семантически эквивалентными языками. Для каждого из указанных *внешних языков* определяется его *синтаксис* и *денотационная семантика*.

§ 2.3.1. Внешние идентификаторы sc-элементов — знаков, входящих в конструкции SC-кода

⇒ *подраздел**:
- *Пункт 2.3.1.1. Понятие sc-идентификатора sc-элемента*
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*
- *Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента*
- *Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента*
- *Пункт 2.3.1.5. Понятие сложного sc-идентификатора sc-элемента*

⇒ *ключевое понятие**:
- *sc-идентификатор*
- *основной sc-идентификатор*
- *неосновной sc-идентификатор*
- *часто используемый sc-идентификатор*
- *строковый sc-идентификатор*
- *нестроковый sc-идентификатор*
- *системный sc-идентификатор*
- *нетранслируемый sc-идентификатор*
- *локальный нетранслируемый sc-идентификатор*
- *глобальный нетранслируемый sc-идентификатор*
- *простой sc-идентификатор*
- *sc-выражение*
- *имя собственное*
- *имя нарицательное*

⇒ *ключевое знание**:
- *Правила построения sc-идентификаторов*
- *Правила построения основных sc-идентификаторов*
- *Правила построения системных sc-идентификаторов*
- *Правила построения нетранслируемых sc-идентификаторов*
- *Правила построения простых sc-идентификаторов*
- *Правила построения сложных sc-идентификаторов*

Пункт 2.3.1.1. Понятие sc-идентификатора sc-элемента

sc-идентификатор (или *внешний идентификатор sc-элемента*) необходим *ostis-системе* исключительно для того, чтобы осуществлять обмен информацией с другими *ostis-системами* или со своими пользователями. Для того чтобы представить свою *базу знаний*, решать самые различные *задачи*, связанные с анализом текущего состояния и эволюцией своей *базы знаний*, задачи, связанные с анализом текущего состояния (текущих ситуаций) окружающей среды, принятием соответствующих решений (целей) и организацией соответствующих *действий*, направленных на выполнение принятых решений (на достижение поставленных целей), *ostis-системе* не нужны никакие внешние идентификаторы (в частности, имена) соответствующие *sc-элементам*. Но для понимания сообщений, принимаемых от других субъектов (что для *ostis-системы* означает построение *sc-текста*, *семантически эквивалентного* принятому сообщению) и для анализа сообщений, передаваемых другим субъектам (что для *ostis-системы* означает синтез *внешнего текста*, *семантически эквивалентного* заданному *sc-тексту* и удовлетворяющего некоторым дополнительным требованиям, например, эмоционального характера) *ostis-системе* необходимо знать, как в принимаемом или передаваемом сообщении изображаются (представляются) *знаки*, синонимичные sc-элементам, которые уже хранятся или могут храниться в составе *базы знаний ostis-системы*. В качестве *внешнего идентификатора sc-элемента* чаще всего используются имена (термины) соответствующих (обозначаемых) сущностей, представленные отдельными словами или словосочетаниями на различных естественных языках, но также могут использоваться иероглифы, условные обозначения, пиктограммы.

В общем случае *sc-элементу* может соответствовать несколько синонимичных ему имен на разных *естественных языках*. Более того, *sc-элементу* может соответствовать несколько синонимичных ему имен на одном и том же *естественном языке*. В этом случае одно из этих имен объявляется как *основной внешний идентификатор* для

соответствующего *sc-элемента* и соответствующего *естественного языка*. Основное требование, предъявляемое к таким внешним идентификаторам это отсутствие как синонимов, так и омонимов в рамках множества *основных внешних идентификаторов sc-элементов* для каждого естественного языка.

Каждый *внешний идентификатор sc-элемента*, используемый *ostis-системой*, может быть описан (представлен) в ее памяти в виде *внутреннего файла ostis-системы*, то есть в виде электронного образа всевозможных вхождений данного внешнего идентификатора во всевозможные внешние тексты соответствующего внешнего языка. В некоторых случаях явное представление в памяти не требуется, например, в случае *нетранслируемых sc-идентификаторов*.

Каждому sc-элементу может соответствовать целое семейство внешних идентификаторов этого *sc-элемента*, которые обычно являются терминами, именующими обозначаемую сущность. Среди этих внешних идентификаторов для каждого идентифицируемого *sc-элемента* выделяется один как основной идентификатор. А неосновные термины (имена), соответствующие этим sc-элементам (в том числе и классам), поясняют *денотационную семантику* указанного *sc-элемента*.

sc-идентификатор
⇒ *часто используемый sc-идентификатор**:
[внешний идентификатор sc-элемента]
:= [строка символов или пиктограмма, взаимно однозначно представляющая соответствующий sc-элемент, хранимый в sc-памяти]
⇒ *разбиение**:
{• *простой sc-идентификатор*
:= [простой внешний идентификатор sc-элемента]
• *sc-выражение*
:= [сложный внешний идентификатор sc-элемента, в состав которого входит один или несколько идентификаторов других sc-элементов]
}
⇒ *разбиение**:
{• *основной sc-идентификатор*
:= [основной sc-идентификатор для носителей дополнительно указываемого языка общения (например, соответствующего естественного языка)]
⇒ *примечание**:
[*основной sc-идентификатор* является уникальным (не имеет синонимов и омонимов) в рамках соответствующего естественного языка]
⊃ *основной международный sc-идентификатор*
• *неосновной sc-идентификатор*
⊃ (*неосновной sc-идентификатор* ∩ *пояснение*)
:= [неосновной sc-идентификатор, являющийся одновременно и пояснением обозначаемой сущности(см. *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*)]
⊃ (*неосновной sc-идентификатор* ∩ *определение*)
:= [неосновной sc-идентификатор, являющийся одновременно и определением обозначаемого понятия(см. *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*)]
⊃ *часто используемый sc-идентификатор*
}

Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента

В качестве ***основного sc-идентификатора*** могут использоваться также общепринятые международные условные обозначения некоторых сущностей, например, обозначения часто используемых функций (sin, cos, tg, log, и так далее), единиц измерения, денежных единиц и многое другое. Формально каждый *основной международный sc-идентификатор* считается *основным sc-идентификатором* также и для каждого *естественного языка*, несмотря на то, что символы, используемые в *основных международных sc-идентификаторах*, могут не соответствовать алфавиту некоторых или даже всех *естественных языков*.

С помощью *неосновного sc-идентификатора* указываются возможные *синонимы** соответствующего *основного sc-идентификатора*, которые в частности, могут пояснять или даже определять обозначаемую сущность, указывает на важные свойства этой сущности.

Для некоторых *sc-элементов* могут часто использоваться не только основной, но и *неосновной sc-идентификатор* (особенно в неформальных текстах — в пояснениях, примечаниях и тому подобных). Явное выделение такого класса *sc-идентификаторов* позволяет упростить семантический анализ исходных текстов *баз знаний*.

системный sc-идентификатор — это *sc-идентификатор*, являющийся уникальным в рамках всей базы знаний *Экосистемы OSTIS*. Данный *sc-идентификатор*, как правило, используется в исходных текстах *базы знаний*, при обмене сообщениями между *ostis-системами*, а также для взаимодействия *ostis-системы* с компонентами, реализованными с использованием средств, внешних с точки зрения *Технологии OSTIS*, например, программ, написанных на традиционных языках программирования. Алфавит *системных sc-идентификаторов* максимально упрощен для того, чтобы обеспечить удобство автоматической обработки таких *sc-идентификаторов* с использованием современных технических средств, в частности, запрещены пробелы и различные специальные символы.

основной sc-идентификатор
- ⊂ *файл-образец ostis-системы*
- ⇔ *семантическая эквивалентность**:
  - [Все *основные идентификаторы sc-элементов* в памяти *ostis-системы* оформляются в виде копируемых файлов-образцов ostis-системы. Аналогичное утверждение справедливо и для неосновных *часто используемых sc-идентификаторов*. Все остальные *неосновные sc-идентификаторы* считаются вспомогательными файлами-экземплярами.]
  - ⇒ *пояснение**:
    - [Копии основных sc-идентификаторов входят в состав внешних текстов различных языков (*SCg-кода*, *SCs-кода*, *SCn-кода*), а также в различных падежах, склонениях, спряжениях в состав *файлов ostis-систем*.]

sc-идентификатор
- ⇒ *разбиение**:
  - {• *строковый sc-идентификатор*
    - := [sc-идентификатор, представленный строкой символов, которая является именем обозначаемой сущности]
    - := [имя сущности, обозначаемой идентифицируемым sc-элементом]
    - := [имя (термин, словосочетание), синонимичное соответствующему (идентифицируемому) sc-элементу и представленное в соответствующем алфавите символов]
  - • *нестроковый sc-идентификатор*
    - ⇒ *пояснение**:
      - [В общем случае в качестве *sc-идентификатора* некоторого *sc-элемента* может выступать произвольный *внутренний файл ostis-системы*, например, пиктограмма, условное обозначение или даже аудиофрагмент.]
  - }
- ⇒ *примечание**:
  - [Введенные нами *sc-идентификаторы* используются во всех *внешних языках*, близких *SC-коду* — в *SCg-коде*, *SCs-коде* и *SCn-коде*.]

строковый sc-идентификатор
- := [имя, приписываемое идентифицируемому sc-элементу]
- := [имя сущности, обозначаемой идентифицируемым sc-элементом]
- := [строка символов, синонимичная соответствующему идентифицируемому sc-элементу]
- ⊃ *основной строковый sc-идентификатор*
  - := [уникальное для каждого естественного языка внешнее имя, приписываемое идентифицируемому sc-элементу]
  - ⊃ *основной русскоязычный sc-идентификатор*
  - ⊃ *системный sc-идентификатор*
  - ⊃ *основной англоязычный sc-идентификатор*
  - ⊃ *основной германоязычный sc-идентификатор*
  - ⊃ *основной франкоязычный sc-идентификатор*
  - ⊃ *основной италоязычный sc-идентификатор*
  - ⊃ *основной китайскоязычный sc-идентификатор*
- ⊃ *системный sc-идентификатор*

Символами, использующимися в *системном sc-идентификаторе*, могут быть буквы латинского алфавита, цифры, знак нижнего подчеркивания и знак тире. Для обеспечения интернационализации рекомендуется формировать *системные sc-идентификаторы* на основании основных англоязычных *sc-идентификаторов*. Таким образом, наиболее целесообразно формировать *системный sc-идентификатор sc-элемента* из основного англоязычного

путем замены всех символов, не входящих в описанный выше алфавит на символ нижнее подчеркивание ("_"). Кроме того, заглавные буквы чаще всего заменяются на соответствующие строчные.

Для именования *sc-элементов*, являющихся знаками *ролевых отношений*, вместо знака "/" в *системном sc-идентификаторе* используется приставка "rrel" и далее после нижнего подчеркивания записывается имя *ролевого отношения*. Для именования sc-элементов, являющихся знаками *неролевых отношений*, вместо знака "*" в *системном sc-идентификаторе* используется приставка "nrel" и далее после нижнего подчеркивания записывается имя *неролевого отношения*. Для именования sc-элементов, являющихся знаками классов *понятий* в *системном sc-идентификаторе* используется приставка "concept" и далее после нижнего подчеркивания записывается имя *класса*. Для именования sc-элементов, являющихся знаками *структур* в *системном sc-идентификаторе* используется приставка "struct" и далее после нижнего подчеркивания записывается имя *структуры*.

Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента

нетранслируемый sc-идентификатор
⊂ *системный sc-идентификатор*
:= [sc-идентификатор, не представляемый в базе знаний ostis-системы]
:= [sc-идентификатор, существующий только вне базы знаний ostis-системы]
⇒ *разбиение**:
{• *глобальный нетранслируемый sc-идентификатор*
• *локальный нетранслируемый sc-идентификатор*
}

нетранслируемый sc-идентификатор используется только в рамках исходных текстов *баз знаний* (в том числе, *sc.s-текстов*) и при обмене сообщениями между *ostis-системами* в тех случаях, когда необходимо в нескольких фрагментах исходного текста *базы знаний* или передаваемого сообщения использовать имя одного и того же *sc-элемента*, но при этом указанный *sc-элемент* не имеет *системного sc-идентификатора* и вводить его нецелесообразно. Использование *нетранслируемого sc-идентификатора* позволяет повысить читабельность и структурированность исходных текстов *баз знаний*, а также позволяет обратиться к одному и тому же неименуемому (в рамках базы знаний) *sc-элементу* в разных файлах исходных текстов *баз знаний* или в разных сообщениях, передаваемых между *ostis-системами*. В качестве таких *sc-элементов* часто выступают знаки *структур* и *связок*.

Таким образом, *нетранслируемый sc-идентификатор* существует только вне *базы знаний ostis-системы* и при формировании *базы знаний* из исходных текстов или при погружении в *базу знаний* полученного сообщения соответствующий им *внутренний файл ostis-системы* не создается.

нетранслируемый sc-идентификатор строится по тем же принципам, что и *системный sc-идентификатор*, но в начале *нетранслируемого sc-идентификатора* ставится одна или несколько точек ("."), количество которых определяет область видимости данного *нетранслируемого sc-идентификатора*.

глобальный нетранслируемый sc-идентификатор считается уникальным в рамках всего имеющегося набора исходных текстов *баз знаний* и/или передаваемых между *ostis-системами* сообщений. Таким образом, *sc-элементы*, имеющие на уровне исходных текстов (в том числе, в разных файлах исходных текстов) одинаковый *глобальный нетранслируемый sc-идентификатор*, считаются синонимичными и в *базе знаний ostis-системы* представляются одним и тем же *sc-элементом*.

Можно сказать, что областью видимости *глобального нетранслируемого sc-идентификатора* является весь набор (репозиторий) исходных текстов некоторой *базы знаний*. В начале *глобального нетранслируемого sc-идентификатора* ставится одна точка.

локальный нетранслируемый sc-идентификатор считается уникальным в рамках конкретного файла исходных текстов *баз знаний* и/или передаваемого между *ostis-системами* сообщения. Таким образом, *sc-элементы*, имеющие в рамках одного файла исходных текстов одинаковый *локальный нетранслируемый sc-идентификатор*, считаются синонимичными, в то время как *sc-элементы*, имеющие в рамках разных файлов исходных текстов одинаковый *локальный нетранслируемый sc-идентификатор* будут считаться разными *sc-элементами*.

Можно сказать, что областью видимости *локального нетранслируемого sc-идентификатора* является конкретный файл исходных текстов *базы знаний*. В начале *локального нетранслируемого sc-идентификатора* ставится две точки.

Уникальный *нетранслируемый sc-идентификатор* используется для однократного обозначения конкретного неименуемого *sc-элемента* в рамках исходных текстов *баз знаний* и/или передаваемых между *ostis-системами* сообщений.

Кроме того, уникальный *нетранслируемый sc-идентификатор* может использоваться при визуализации *баз знаний* в виде, например, *sc.s-текстов* или *sc.n-текстов*, в тех случаях, когда необходимо визуализировать *sc-элемент*, не имеющий *sc-идентификатор*.

Каждое вхождение уникального *нетранслируемого sc-идентификатора* в исходный текст *базы знаний* или передаваемое сообщение считается обозначением разных *sc-элементов* (чаще всего, *sc-узлов*).

sc-идентификатор*
- ⊃ *основной sc-идентификатор**
  - ≔ [*бинарное ориентированное отношение*, каждая пара которого связывает *sc-элемент* с внутренним файлом *ostis-системы*, который содержит *основной sc-идентификатор* указанного *sc-элемента*.]
  - ⇒ *примечание**:
    [отношение, связывающее *sc-элементы* с файлами, содержащими их *основные sc-идентификаторы*, само имеет свой *основной sc-идентификатор*, который представляет собой строку символов, имеющую вид “основной sc-идентификатор*”. Заметим, что в состав первого домена отношения *“основной sc-идентификатор*”* входит также и *sc-узел*, обозначающий само это отношение.]
- ⊃ *системный sc-идентификатор**
  - ≔ [*бинарное ориентированное отношение*, каждая пара которого связывает *sc-элемент* с внутренним файлом *ostis-системы*, который содержит *системный sc-идентификатор* указанного *sc-элемента*.]

системные sc-идентификаторы отличаются от *основных*, во-первых, требованием к уникальности в рамках всей *базы знаний Экосистемы OSTIS* (а, значит, независимостью от внешнего языка), а во-вторых, более простым алфавитом, удобным для автоматической обработки.

системные sc-идентификаторы и *нетранслируемые sc-идентификаторы* выполняют схожие функции, связанные с именованием *sc-элементов* на уровне исходных текстов *баз знаний* или передаваемых между *ostis-системами* сообщений.

Каждый *системный sc-идентификатор* представляется в *базе знаний* в виде *внутреннего файла ostis-системы* и связан с соответствующим *sc-элементом* парой отношения *системный sc-идентификатор*. *нетранслируемые sc-идентификаторы* не представляются в рамках *базы знаний*, не имеют соответствующих *внутренних файлов ostis-системы* и на уровне *базы знаний* никак не связаны с идентифицируемыми ими *sc-элементами*.

Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента

простой sc-идентификатор — *sc-идентификатор sc-элемента*, в состав которого идентификаторы других sc-элементов не входят и который не содержит *транслируемую в SC-код* информацию об обозначаемой им сущности.

простой строковый sc-идентификатор — *простой sc-идентификатор*, представляющий собой строку (цепочку) символов, которая является именем (названием) той же сущности, что и идентифицируемый sc-элемент. *простые строковые sc-идентификаторы* являются наиболее распространенным видом идентификаторов, приписываемых sc-элементам.

простой строковый sc-идентификатор
- ⊃ *системный sc-идентификатор*
- ⊃ *простой строковый идентификатор sc-переменной*
  - ∋ *пример′*:
    [_var1]
- ⊃ *простой строковый sc-идентификатор неролевого отношения*
  - ≔ [простой строковый идентификатор sc-узла, являющегося знаком неролевого отношения]
  - ∋ *пример′*:
    [включение множеств*]
- ⊃ *простой строковый sc-идентификатор ролевого отношения*
  - ≔ [простой строковый идентификатор sc-узла, являющегося знаком ролевого отношения]
  - ∋ *пример′*:
    [слагаемое′]
- ⊃ *простой строковый sc-идентификатор класса классов*
  - ≔ [простой строковый идентификатор sc-узла, являющегося знаком класса классов]
  - ∋ *пример′*:
    [длина^]
- ⊃ *sc-идентификатор внешнего файла ostis-системы*
  - ≔ [URL-идентификатор]

- ∋ *пример'*:
  [“file:///home/user/image1.png”]
  ⇒ *примечание**:
  [Данный sc-идентификатор описывает абсолютный путь к файлу под названием “image1.png”]
- ∋ *пример'*:
  [“file://image1.png”]
  ⇒ *примечание**:
  [Данный sc-идентификатор описывает относительный путь к файлу под названием “image1.png”]
- ∋ *пример'*:
  [“https://conf.ostis.net/content/image1.png”]
  ⇒ *примечание**:
  [Данный sc-идентификатор описывает путь к файлу под названием “image1.png”, расположенному на удаленном сервере.]

sc-идентификатор внешнего файла ostis-системы предназначен для описания местоположения *внешнего файла ostis-системы* и представляет собой строку символов, которая строится в соответствии со стандартом URL, а затем берется в двойные кавычки. Кавычки нужны для однозначности определения того, где начинается и заканчивается данный *sc-идентификатор*, поскольку в общем случае в URL разрешены пробелы. Целесообразность этого обусловлена тем, что *sc-идентификаторы* данного типа часто используются в файлах исходных текстов баз знаний ostis-систем.

sc-идентификатор внешнего файла ostis-системы с точки зрения *Технологии OSTIS* является *простым строковым sc-идентификатором*, хотя и может содержать специальные символы, например “%” или “/”. Это связано с тем, что указанный идентификатор не несет в себе семантически значимой информации о свойствах самого *sc-элемента*, обозначаемого таким *sc-идентификатором*, а только информацию о его расположении в текущем состоянии внешнего мира *ostis-системы*.

имя нарицательное — *имя*, которое либо не является обозначением какого-либо класса сущностей, либо является обозначением (именем) некоторого класса сущностей, но построенным без использования нарицательного имени этого класса, либо является именем некоторого класса сущностей, построенным с использованием нарицательного имени этого класса либо путем преобразования имени нарицательного во множественное число, либо путем дополнительного использования в начале формулируемого имени таких слов или словосочетаний, как “Знак класса”, “Класс”, “Знак множества”, “Множество”, “Знак множества всевозможных”, “Множество всевозможных”. ***имя собственное*** всегда начинается с большой буквы.

имя нарицательное — *имя* некоторого класса сущностей (а, точнее, имя класса sc-элементов, обозначающих сущности некоторого класса), которое, с одной стороны, является знаком всего указанного класса, а, с другой стороны, соответствует (может быть приписано) любому экземпляру этого класса. *имя нарицательное* всегда начинается с маленькой буквы.

Правила построения простого строкового sc-идентификатора включают в себя:
- алфавит символов, используемых в *простых строковых sc-идентификаторах*;
- префиксы и суффиксы, используемые в *простых строковых sc-идентификаторах*;
- разделители и ограничители, используемые в *простых строковых sc-идентификаторах*;
- правила построения *имен собственных* и *имен нарицательных*, являющихся простыми строковыми sc-идентификаторами;
- правила построения *простых строковых sc-идентификаторов*, определяемые различными классами идентифицируемых *sc-элементов*.

Общим правилом построения *простого sc-идентификатора* является стремление максимально возможным образом использовать сложившуюся терминологию. Но при этом следует подчеркнуть, что необходимость исключения омонимии в *sc-идентификаторах* требует строгого формального уточнения семантической интерпретации каждого используемого термина. Особо подчеркнем то, что в *ostis-системах* процесс построения новых терминов (*sc-идентификаторов*) и процесс совершенствования существующей терминологии по отношению к процессу эволюции *баз знаний*, представленных в *SC-коде*, с технической точки зрения абсолютно не зависят друг от друга. Кроме того, следует помнить, что далеко не все *sc-элементы*, входящие в состав базы знаний *ostis-системы*, должны иметь соответствующие им *sc-идентификаторы* (быть идентифицированными). Существенно подчеркнуть то, что далеко не все sc-элементы должны иметь *простые sc-идентификаторы* по той простой причине, что для многих sc-элементов в случае необходимости можно достаточно легко построить идентифицирующее их *sc-выражение* (sc-выражение). Тем не менее, для целого ряда сущностей (например, для понятий, исторических событий и тому подобных) обойтись без простых sc-идентификаторов очень сложно. Очевидно, что идентифицированными (именованными) должны быть все используемые понятия, вводимые в соответствующих предметных областях и специфицируемые соответствующими онтологиями. Идентифицированными также должны быть обладающие особыми свойствами ключевые экземпляры (элементы) некоторых понятий, различные социально значимые объек-

ты (персоны, населенные пункты, географические объекты, страны, организации, библиографические источники, исторические события и многое другое).

Первым символом каждого *простого строкового sc-идентификатора*, идентифицирующего *sc-переменную* (переменный *sc-элемент*), является *знак подчеркивания*. При этом, если после указанного знака подчеркивания указывается *имя нарицательное* некоторого *класса sc-элементов*, то рассматриваемый *простой строковый sc-идентификатор* становится уже *sc-выражением*, содержащим информацию о том, что *областью возможных значений** идентифицируемой *sc-переменной* является указанный *класс sc-элементов*.

Последним символом *простого строкового sc-идентификатора*, идентифицирующего *sc-узел*, обозначающий *неролевое отношение*, заданное на множестве *sc-элементов*, является надстрочная *звездочка*.

Последним символом *простого строкового sc-идентификатора*, идентифицирующего *sc-узел*, обозначающий заданное на множестве sc-элементов *ролевое отношение* (то есть *отношение*, являющееся подмножеством *отношения принадлежности*), является *штрих* (прямой апостроф).

Последним символом *простого строкового sc-идентификатора*, идентифицирующего *sc-узел*, обозначающий понятие, являющееся классом классов (таковыми в частности, являются различного рода параметры — длина, площадь, объем, масса) является символ *циркумфлекс* (“крышка”). Однако, если отсутствует омонимичный идентификатор без этого суффикса, то указанный символ можно не приписывать.

простой строковый sc-идентификатор может рассматриваться как последовательное сокращение *sc-идентификаторов* одного и того же *sc-элемента* с преобразованием *имен собственных* в *имена нарицательные* и наоборот.

При наличии синонимичных *имен собственных* и *имен нарицательных* при выборе *основного sc-идентификатора* преимущество имеют *имена нарицательные*. Так, например, основным идентификатором *sc-узла*, обозначающего *SC-код*, является термин “*sc-текст*”, а термин “*SC-код*”, являющийся *именем собственным*, считается *неосновным часто используемым sc-идентификатором*. Подчеркнем при этом, что имя нарицательное — это всегда имя некоторого класса сущностей (в частности, понятия). В конце этого имени может быть указан либо признак класса классов, либо признак неролевого отношения (класса связок), либо признак ролевого отношения (подмножества отношения принадлежности), либо не указано ничего. Последнее означает, что именуется класс, не являющийся ни классом классов, ни классом связок. А это, в свою очередь, означает, что именуемым классом является либо класс первичных (терминальных) сущностей, либо класс структур.

Одно и то же словосочетание, которому приписываются разные дополнительные признаки, может быть основой для *внешних идентификаторов* разных *sc-элементов*.

В рамках *SC-кода* целесообразно вводить правила унифицированного построения *простых sc-идентификаторов* и целого ряда других классов идентифицируемых сущностей — *персон*, *библиографических источников* (публикаций), *разделов баз знаний ostis-систем*, *файлов ostis-систем*, самих *ostis-систем*.

Пункт 2.3.1.5. Понятие сложного sc-идентификатора sc-элемента

sc-выражение — *sc-идентификатор*, который не только обозначает соответствующую сущность, но также содержит информацию, представляющую собой по возможности однозначную спецификацию указанной сущности.

Однозначную спецификацию сущности, которая является понятием, называют ***определением*** этого понятия (см. *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*).

sc-выражение
:= [имя соответствующей (именуемой) сущности построенное по принципу “та (тот), которая (который) указываемым образом связана с другими указываемыми сущностями”]
:= [выражение, идентифицирующее sc-элемент]
:= [идентификатор sc-элемента, в состав которого входят другие идентификаторы и денотационная семантика которого точно определяется конкретным набором правил построения таких сложных (комплексных) идентификаторов, состоящих из определенным образом связанных между собой других идентификаторов]
:= [сложный идентификатор, состоящий из других идентификаторов]
:= [идентификатор, который представляет собой конструкцию, состоящую из нескольких других идентификаторов, а также из некоторых разделителей и ограничителей, и денотационная семантика которого однозначно задается конфигурацией указанной конструкции]
:= [сложный sc-идентификатор]
:= [сложный (составной) внешний идентификатор sc-элемента]
:= [выражение, идентифицирующее sc-элемент]
:= [sc-идентификатор, в состав которого входит один или несколько простых sc-идентификаторов]

sc-выражение разбивается на:

- *sc-выражение неориентированного множества — sc-выражение*, ограниченное фигурными скобками;
- *sc-выражение структуры* — sc-выражение, обозначающее структуру, представленную на любом известном и легко определяемом языке (*Русском*, *Английском*, *SCg-коде*, *SCs-коде*, *SCn-коде*). *sc-выражение структуры* обозначает структуру, содержащую sc-текст, семантически эквивалентный тому тексту (на некотором известном языке), который заключен в фигурные скобки. Чаще всего такой текст записывается на формальном языке, например, *SCs-коде*, и может быть автоматически однозначно интерпретирован. Возможна ситуация, когда указанный текст записан на менее неформальном языке, например, *Русском*, но в этом случае его автоматическая интерпретация значительно усложняется и в общем случае не всегда однозначна. В текущей реализации средств разработки исходных текстов баз знаний в соответствии с более старой версией правил простроения sc-выражений вместо фигурных скобок *sc-выражение структуры* ограничивается квадратными скобками со звездочками (“[*” и “*]”);
- *sc-выражение ориентированного множества* — sc-выражение кортежа, ограничиваемое угловыми скобками и обозначающее упорядоченное (ориентированное) множество sc-элементов, порядок которых задается последовательностью перечисляемых их sc-идентификаторов;
- *sc-выражение внутреннего файла ostis-системы* — sc-выражение, обозначающее *внутренний файл ostis-системы*, визуальное представление (изображение) которого ограничивается квадратными скобками. sc-выражение внутреннего файла ostis-системы обозначает *внутренний файл ostis-системы*, содержимое которого заключено в квадратные скобки, ограничивающие данное sc-выражение. Дополнительная спецификация *внутреннего файла ostis-системы* легко осуществляется с помощью *SC-кода*. Сюда входит *язык*, на котором представлена *информационная конструкция*, являющаяся содержимым *файла*, формат кодирования, *автор** и многое другое;
- *sc-выражение, обозначающее файл-образец ostis-системы* — sc-выражение, ограниченное ограниченное квадратными скобками с восклицательными знаками;
- *sc-выражение, построенное на основе бинарного отношения* — sc-выражение, в состав которого входят либо (1) *sc-идентификатор*, обозначающий бинарное ориентированное отношение, и (2) в круглых скобках sc-идентификатор одного из элементов первого домена указанного бинарного ориентированного отношения, либо (1) sc-идентификатор, обозначающий бинарное неориентированное отношение и (2) в круглых скобках sc-идентификатор одного из элементов области определения указанного бинарного неориентированного отношения. sc-выражение, построенное путем указания некоторого бинарного отношения (обычно функционального) и одного из его аргументов (в круглых скобках);
- *sc-выражение, построенное на основе алгебраической операции* — sc-выражение, ограниченное круглыми скобками и построенное путем указания *sc-идентификаторов*, разделенных знаком алгебраической операции;
- *sc-выражение, идентифицирующее sc-коннектор* — sc-выражение, ограниченное круглыми скобками и идентифицирующее *sc-коннектор*, инцидентный двум указанным sc-элементам и имеющий тип, задаваемый путем изображения соответствующего sc.s-коннектора. Для упрощения восприятия и обработки *sc-выражений, идентифицирующих sc-коннектор* вводится следующее ограничение: первым и третьим компонентом такого sc-выражения может быть только *простой sc-идентификатор*. В рамках sc.s-текстов внутри *sc-выражений, идентифицирующих sc-коннектор* допускается также использование sc.s-модификаторов.

Использование *sc-выражений* позволяет существенно сократить число “придумываемых” *sc-идентификаторов*, каковыми в конечном счете становятся только *простые sc-идентификаторы*, поскольку, зная то, как связан идентифицируемый *sc-элемент* с теми *sc-элементами*, которые уже имеют *sc-идентификаторы*, во многих случаях можно построить sc-выражение, идентифицирующее указанный *sc-элемент*. Кроме того, каждое *sc-выражение*, являясь внешним идентификатором, является также и транслируемым формальным текстом, содержащим некоторую информацию об обозначаемой ею сущности.

Очевидно, что некоторые *sc-выражения* могут интерпретироваться неоднозначно. Например, два одинаковых *sc-выражения, идентифицирующих sc-коннектор*, могут изображать как один и тот же *sc-коннектор*, так и кратные sc-коннекторы одного и того же вида, связывающие одни и те же два *sc-элемента*. Аналогичная неоднозначность может возникать при использовании *sc-выражений, построенных на основе бинарного отношения* (*подмножество*(S)*) и ряда других *sc-выражений*.

В то же время, некоторые *sc-выражения* являются однозначными, то есть всегда идентифицируют одну и ту же сущность в любом тексте, в состав которого входят. Например выражение “*sin(x)*” является однозначным (при условии, что “x” в разных текстах обозначает одно и то же). Однако, однозначность или неоднозначность sc-выражений зависит от их семантики, таким образом установить факт однозначности на уровне внешнего текста достаточно сложно.

Для решения проблемы неоднозначности интерпретации sc-выражений такого рода будем считать, что каждое вхождение какого-либо sc-выражения в различные тексты (например, *sc.s-тексты*) обозначает разные *sc-элементы*. После трансляции таких текстов в sc-текст может оказаться, что некоторые из указанных sc-выражений на самом деле обозначали одну и ту же сущность, в этом случае соответствующие sc-элементы будут "склеены", но уже на уровне *sc-текста*, а не на уровне внешнего текста(см. *Голенков В.В..СтандОТОП-2021кн*).

Следует отметить, что факт совпадения *sc-выражений* в рамках некоторого внешнего текста может являться поводом для анализа идентифицируемых этими *sc-выражениями sc-элементов* на возможную синонимичность и явно фиксироваться, например, на этапе погружения внешнего текста в *sc-память*.

ограничитель sc-выражений
:= [ограничитель, используемый в sc-выражениях]
:= [ограничитель, ограничивающий sc-выражения или их компоненты]
⇒ *разбиение**:
{• *левый ограничитель sc-выражений*
:= [начальный ограничитель sc-выражений]
:= [открывающий ограничитель sc-выражений]
• *правый ограничитель sc-выражений*
:= [конечный ограничитель sc-выражений]
:= [закрывающий ограничитель sc-выражений]
}

§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)

⇒ *подраздел**:
- *Пункт 2.3.2.1. Синтаксис SCg-кода*
- *Пункт 2.3.2.2. Денотационная семантика SCg-кода*
- *Пункт 2.3.2.3. Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду*

⇒ *ключевой знак**:
- *SCg-код*
- *Алфавит SCg-кода^*
- *Синтаксис SCg-кода*
- *Денотационная семантика SCg-кода*
- *Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду*

⇒ *ключевое понятие**:
- *sc.g-элемент*
- *sc.g-коннектор*
- *sc.g-ребро*
- *sc.g-дуга*
- *sc.g-рамка*
- *sc.g-шина*

Введение в § 2.3.2.

SCg-код
:= [Semantic Code graphical]
:= [Язык визуального (графического) представления баз знаний ostis-систем]
:= [Язык внешнего графического представления конструкций внутреннего языка ostis-систем]
∈ *графовый язык*

SCg-код представляет собой способ визуализации *sc-текстов* (*информационных конструкций SC-кода*) в виде рисунков этих абстрактных конструкций. Подчеркнем, что абстрактная *графовая структура* и ее рисунок (графическое изображение) — это не одно и то же даже если они *изоморфны* друг другу. *SCg-код* рассматривается нами как объединение *Ядра SCg-кода*, обеспечивающего изоморфное графическое изображение любого *sc-текста*, а также нескольких направлений расширения этого ядра, обеспечивающих повышение компактности и "читабельности" текстов *SCg-кода* (*sc.g-текстов*). *SC-код* — это рассмотрение множества всевозможных графически представленных (визуализированных) графовых структур как универсального языка представления знаний с соответствующим синтаксисом и семантикой(см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*).

Основная цель *SCg-кода* — иметь четкие синтаксические графические признаки изображения *sc.g-элементов*, позволяющие легко выделить и различать такие классы *sc.g-элементов*, как:

- *sc.g-константы* (знаки константных сущностей) и *sc.g-переменные* (изображения переменных, значениями которых являются соответствующие sc-элементы);
- *sc.g-переменные*, значениями которых являются *sc-константы*, и *sc.g-переменные*, значениями которых являются *sc-переменные*;
- знаки постоянных (стабильных) сущностей и знаки временных (нестабильных, временно существующих, ситуативных) сущностей;
- *sc.g-коннекторы* (знаки бинарных связей) и *sc.g-элементы*, не являющиеся *sc.g-коннекторами*;
- неориентированные sc.g-коннекторы (*sc.g-ребра*) и ориентированные (*sc.g-дуги*);
- *sc.g-дуги принадлежности* и *sc.g-дуги*, не являющиеся таковыми;
- *sc.g-дуги позитивной принадлежности*, негативной принадлежности и нечеткой принадлежности.

Пункт 2.3.2.1. Синтаксис SCg-кода

Алфавит Ядра SCg-кода^
:= [Алфавит sc.g-элементов, графически изображающих sc-элементы]
⇐ *ключевой знак**:
*Алфавит Ядра SCg-кода^*
⇒ *пояснение**:
[*Алфавит Ядра SCg-кода^* взаимно однозначно соответствует *Алфавиту SC-кода^*. Указанное соответствие представлено на *Рисунок. SCg-текст. Алфавит SCg-кода^*.]

На *Рисунок. SCg-текст. Алфавит SCg-кода^* приведен перечень элементов *Алфавита SCg-кода^*. Этот перечень оформлен в виде *sc.g-текста* и представляет собой изображение примеров всех введенных видов *sc.g-элементов* (по одному примеру каждого вида). При этом, указанные примеры *sc.g-элементов* разбиты на пять групп (*Рисунок. SCg-текст. Алфавит SCg-кода^*). Первая группа (верхняя строка) включает в себя *sc.g-элементы*, для которых константность и постоянство обозначаемых ими сущностей требует дополнительного уточнения. Остальные четыре группы *sc.g-элементов* аналогичны друг другу и включают в себя соответственно:
- знаки *константных постоянных сущностей*;
- знаки *константных временных сущностей*;
- изображения *sc-переменных*, значениями которых или значениями значений которых (в случае, если значениями переменных являются переменные) являются знаки *константных постоянных сущностей*;
- изображения *sc-переменных*, значениями которых или значениями значений которых (в случае, если значениями переменных являются переменные) являются знаки *константных временных сущностей*.

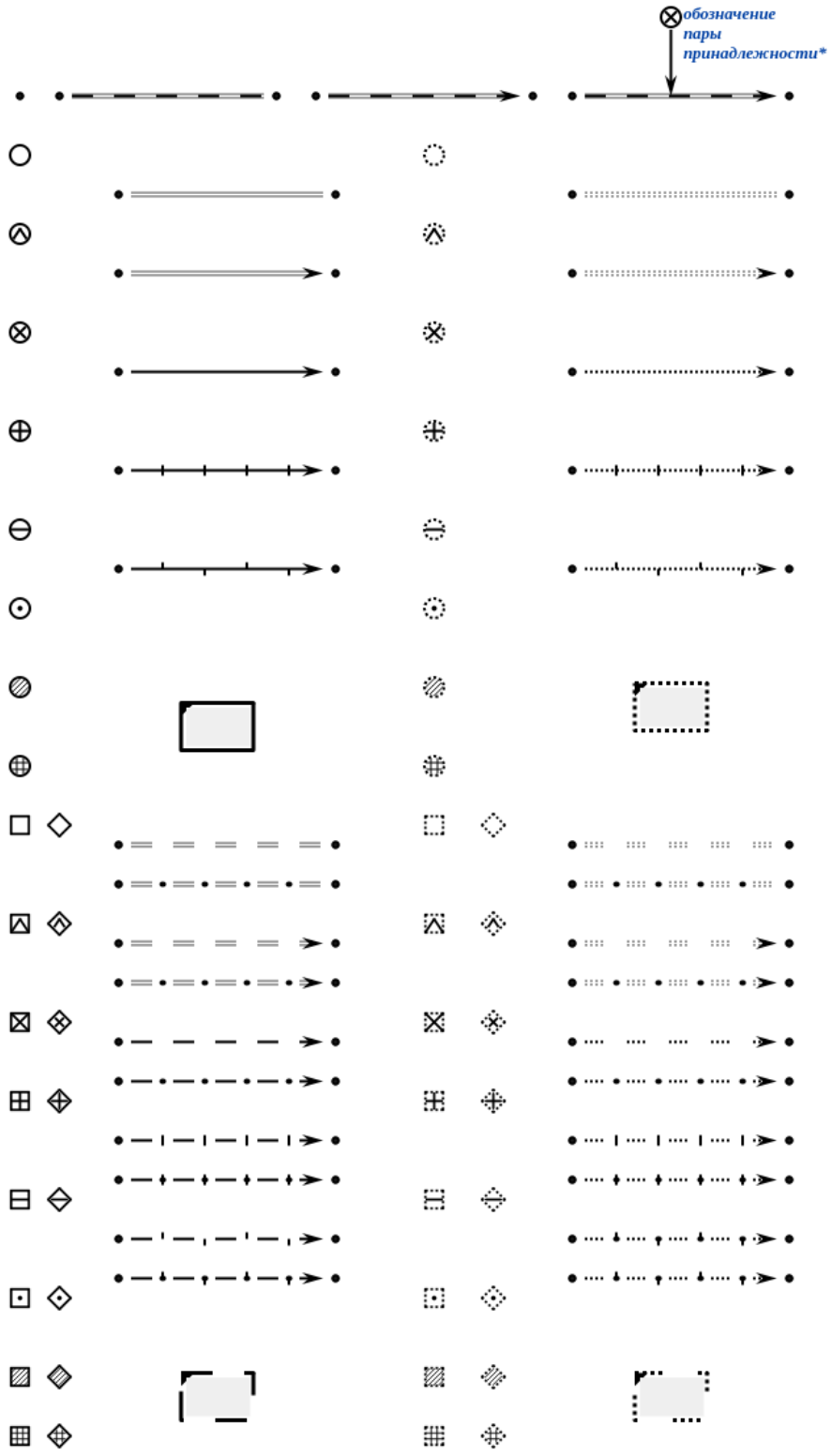
Особое место в *SCg-коде* занимает изображение sc-элементов, являющихся *обозначениями пар принадлежности**, путем явного использования этого *семантически выделяемого класса sc-элементов*. Данный *sc.g-элемент* используется тогда, когда нам необходимо изобразить *sc-дугу*, о которой известно, что она является *обозначением пары принадлежности**, но неизвестно о какой принадлежности идет речь — о константной или переменной, о постоянной или временной, о позитивной, негативной или нечеткой.

Кроме *sc.g-элементов*, перечисленных на *Рисунок. SCg-текст. Алфавит SCg-кода^*, в состав *Алфавита SCg-кода* входят также следующие *sc.g-элементы*:
- внешние идентификаторы *sc-элементов*, идентичные (приписываемые) соответствующим *sc.g-элементам*.
- sc.g-контура, каждый из которых является знаком некоторого sc-текста (структуры, состоящей из sc-элементов). Каждый такой sc-текст может быть:
  - либо константной постоянной структурой;
  - либо константной временной структурой (ситуацией);
  - либо переменной структурой, значениями которой являются постоянные структуры изоморфной конфигурации;
  - либо переменной структурой, значениями которой являются временные структуры (ситуации) изоморфной конфигурации;
- *sc.g-рамки* увеличенного размера являются ограничителями изображения различных файлов, хранимых в памяти ostis-системы;
- *sc.g-шины*, являющиеся обозначениями тех же сущностей, что и инцидентные им sc.g-элементы.

Заметим также, что, кроме всех перечисленных элементов *Алфавита SCg-кода^*, каждый из которых имеет вполне определенную денотационную семантику, для формализации *Синтаксиса SCg-кода* необходимо ввести целый ряд более “мелких” синтаксических объектов, например:
- точек инцидентности *sc.g-коннекторов* с *sc.g-узлами*, с другими *sc.g-коннекторами*, с *sc.g-контурами*, с *sc.g-рамками*;

- точек инцидентности *sc.g-шин*;
- точек излома линейных *sc.g-элементов* (sc.g-коннекторов, sc.g-контуров, sc.g-рамок, sc.g-шин).

Рисунок. SCg-текст. Алфавит SCg-кода^
=



Трудно сразу поверить, что на основе такого простого алфавита можно построить удобный и универсальный *графовый язык*. В рамках *Документации Технологии OSTIS* мы постараемся Вас в этом убедить. Кроме того, нас не должна настораживать простота алфавита, поскольку человечество имеет большой опыт кодирования, хранения

в памяти и передачи по каналам связи самых различных информационных ресурсов, используя алфавит, состоящий только из двух классов элементов — единиц и нулей.

Мы ведем речь о принципиально ином (графовом) способе кодирования информации в *компьютерных системах*, но стараемся при этом свести это кодирование к достаточно простому алфавиту хотя бы для того, чтобы искусственно не усложнять проблему создания нового поколения компьютеров, основанных на указанном способе кодирования информации.

Расширения *Ядра SCg-кода* рассмотрим как направления перехода от текстов *Ядра SCg-кода* к более компактным текстам. Но, поскольку это приводит к усложнению *Синтаксиса SCg-кода* и, в первую очередь, к расширению *Алфавита SCg-кода^*, делать такие расширения необходимо обоснованно с учетом частоты встречаемости в рамках *баз знаний ostis-систем* соответствующих фрагментов.

Пункт 2.3.2.2. Денотационная семантика SCg-кода

В *SCg-коде* выделяются Ядро SCg-кода и его расширения. ***Алфавит Ядра SCg-кода^*** — алфавит *sc.g-элементов*, графически изображаемых *sc-элементов*. *Алфавит Ядра SCg-кода^* взаимно однозначно соответствует *Алфавиту SC-кода^*.

Денотационная семантика Ядра SCg-кода соответствует *Денотационной семантике SC-кода*. Это продемонстрировано в *Таблица. Алфавит Ядра SCg-кода^*.

Таблица. Алфавит Ядра SCg-кода^
=

Алфавит SC-кода	***Алфавит Ядра SCg-кода***	***Изображение sc.g-элемента***
sc-узел общего вида	*sc.g-узел общего вида*	
sc-ребро общего вида	*sc.g-ребро общего вида*	
sc-дуга общего вида	*sc.g-дуга общего вида*	
базовая sc-дуга	*базовая sc.g-дуга*	
внутренний файл ostis-системы	*sc.g-узел с содержимым*	

*Алфавит Ядра SCg-кода^* представлен следующими элементами:

- ***sc.g-узел общего вида*** — *sc.g-элемент*, являющийся графическим изображением *sc-узла общего вида*. Все *sc-узлы*, не являющиеся знаками файлов, в тексте (конструкции) *Ядра SCg-кода*, изображаются в виде небольших черных кругов одинакового диаметра, который обозначим через d, и точная величина которого зависит от масштаба отображения *sc.g-текста*;
- ***sc.g-ребро общего вида*** — *sc.g-элемент*, являющийся графическим изображением *sc-ребра общего вида*. Каждое *sc-ребро* в *Ядре SCg-кода* изображается в виде широкой линии, в которой чередуются фрагменты со сплошной заливкой и без заливки, не имеющей самопересечений и имеющей общую толщину, равную примерно $0.7d$;
- ***sc.g-дуга общего вида*** — *sc.g-элемент*, являющийся графическим изображением *sc-дуги общего вида*. Каждая *sc-дуга* в *Ядре SCg-кода* изображается в виде широкой линии, в которой чередуются фрагменты со сплошной заливкой и без заливки, не имеющей самопересечений, имеющей общую толщину, равную примерно $0.7d$ и имеющей изображение стрелочки на одном из концов этой линии;
- ***базовая sc.g-дуга*** — *sc.g-элемент*, являющийся графическим изображением *базовой sc-дуги*. Каждая входящая в состав sc-текста *базовая sc-дуга* в *Ядре SCg–кода* изображается в виде линии произвольной формы, не имеющий самопересечений, имеющий толщину $0.4d$, и имеющей изображение стрелочки на одном из ее концов;
- ***внутренний файл ostis-системы*** — *sc-узел*, являющийся знаком *внутреннего файла ostis-системы*, sc-знак *внутреннего файла ostis-системы*;
- ***sc.g-узел с содержимым*** — *sc.g-узел*, имеющий содержимое, sc.g-узел, являющийся знаком внутреннего файла ostis-системы, sc.g-рамка. *sc.g-рамка* — это всегда прямоугольник, максимальный размер которого не ограничивается, но минимальный фиксируется и соответствует *sc.g-рамке*, внутри которой обозначаемый ею *файл* не отображается. Каждый входящий в sc-текст *sc-узел, имеющий содержимое*, в *Ядре SCg-кода* изображается в виде прямоугольника произвольного размера с толщиной линии $0.6d$. Внутри этого прямоугольника отображается *файл*, обозначаемый изображаемым *sc-узлом*. Если нет необходимости изображать в тексте сам

файл, то *sc-узел*, обозначающий такой *файл*, в *sc.g-тексте* изображается в виде прямоугольника со сторонами $\mathbf{2}\boldsymbol{d}$ по вертикали и $\mathbf{4}\boldsymbol{d}$ по горизонтали.

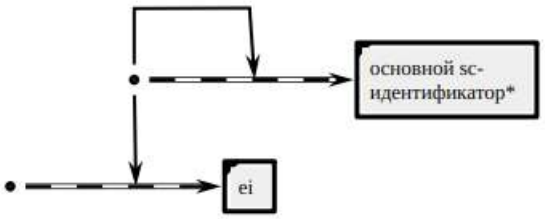
Пункт 2.3.2.3. Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду

⇒ *ключевой знак**:
- *Первое направление расширения Ядра SCg-кода*
- *Второе направление расширения Ядра SCg-кода*
- *Третье направление расширения Ядра SCg-кода*
- *Четвертое направление расширения Ядра SCg-кода*
- *Пятое направление расширения Ядра SCg-кода*
- *Шестое направление расширения Ядра SCg-кода*
- *Седьмое направление расширения Ядра SCg-кода*

Первое направление расширения Ядра SCg-кода

Первое направление синтаксического расширения Ядра SCg-кода — это приписывание некоторым sc.g-элементам *основных sc-идентификаторов** (чаще всего — строковых идентификаторов, то есть имен) *sc-элементов* , изображаемых этими *sc.g-элементами*. Указываемые идентификаторы являются уникальным для каждого идентифицируемого (именуемого) *sc.g-элемента*. Приписывание *sc.g-элементам* уникальных идентификаторов дает возможность в рамках одного *sc.g-текста* дублировать (копировать) некоторые *sc.g-узлы* при условии, если всем таким копиям будут приписаны соответствующие идентификаторы. Такое дублирование *sc.g-узлов* является дополнительным средствам наглядного размещения *sc.g-текстов*. Кроме того, приписывание *sc.g-элементу* соответствующего ему основного (уникального) *sc-идентификатора** представляет собой более компактный вариант изображения *sc.g-текстов*.

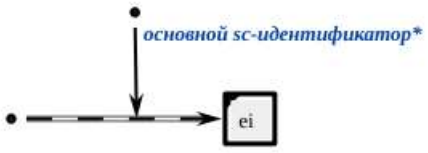
Рисунок. Пример sc.g-текста, трансформируемого по Первому направлению расширения Ядра SCg-кода
=



Здесь (в левом нижнем углу приведенного sc.g-текста) представлен *sc.g-узел общего вида*, изображающий *sc-узел общего вида*, которому соответствует *основной sc-идентификатор** в виде строки "***ei***".

sc.g-узлу общего вида изображающему *sc-узел*, внешним идентификатором которого является строка "*основной sc-идентификатор**" и который, соответственно является знаком *бинарного ориентированного отношения*, каждая *пара* которого связывает идентифицируемый *sc-элемент* с его основным внешним sc-идентификатором, приписывается указанный внешний идентификатор изображаемого им *sc-элемента*. Это продемонстрировано на *Рисунок. Трансформация sc.g-текста по Первому направлению расширения Ядра SCg-кода*.

Рисунок. Трансформация sc.g-текста по Первому направлению расширения Ядра SCg-кода
=



В результате данной трансформации исходный *sc.g-текст* трансформируется в один *sc.g-общего вида*, которому приписывается *основной sc-идентификатор* "***ei***". Это продемонстрировано на *Рисунок. Результат трансформации sc.g-текста по Первому направлению расширения Ядра SCg-кода*.

Рисунок. Результат трансформации sc.g-текста по Первому направлению расширения Ядра SCg-кода
=

Подчеркнем, что рассматриваемая трансформация преобразует исходный текст Ядра *SCg-кода* в текст, семантически эквивалентный, но принадлежащий не Ядру *SCg-кода*, а его расширению.

Второе направление расширения Ядра SCg-кода

Второе направление расширения Ядра SCg-кода — это уточнение типологии *константных постоянных сущностей* и расширение *Алфавита Ядра SCg-кода^*, позволяющее типологию *константных постоянных сущностей* привести в соответствие с синтаксической типологией новых вводимых элементов *Алфавита SCg-кода^*. Рассмотрим подробнее sc.g-элементы, знаки *константных постоянных сущностей* различного вида. Графическим признаком *константных постоянных sc-узлов* в конструкциях SCg-кода является их изображение в виде окружностей диаметра $3d$, где d — диаметр sc.g-узла общего вида. Такое изображение является более компактной записью факта принадлежности заданного sc-узла (назовем его $\boldsymbol{vi}$) классу sc-констант и классу обозначений постоянных сущностей. Запись этого факта в *Ядре SCg-кода* потребует (1) явного изображения sc-узла, обозначающего класс всевозможных константных sc-элементов (класс *sc-констант*), (2) явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс sc-констант, с изображением заданного константного sc-узла, (3) явного изображение sc-узла, обозначающего класс всевозможных sc-элементов, обозначающих *постоянные сущности*, (4) явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс обозначений *постоянных сущностей* с изображением рассматриваемого sc-узла $\boldsymbol{vi}$ (см. *Файл. Изображение спецификации sc.g-элемента средствами Ядра SCg-кода и Первого расширения Ядра SCg-кода*).

Второе направление расширения Ядра SCg-кода — это уточнение типологии *константных постоянных сущностей* и расширение *Алфавита Ядра SCg-кода^*, позволяющее типологию *константных постоянных сущностей* привести в соответствие с синтаксической типологией новых вводимых элементов *Алфавита SCg-кода^*. Рассмотрим подробнее sc.g-элементы, знаки *константных постоянных сущностей* различного вида. Графическим признаком *константных постоянных sc-узлов* в конструкциях SCg-кода является их изображение в виде окружностей диаметра $3d$, где d — диаметр sc.g-узла общего вида. Такое изображение является более компактной записью факта принадлежности заданного sc-узла (назовем его $\boldsymbol{vi}$) классу sc-констант и классу обозначений постоянных сущностей. Запись этого факта в *Ядре SCg-кода* потребует:

- явного изображения sc-узла, обозначающего класс всевозможных константных sc-элементов (класс *sc-констант*);
- явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс sc-констант, с изображением заданного константного sc-узла;
- явного изображение sc-узла, обозначающего класс всевозможных sc-элементов, обозначающих *постоянные сущности*;
- явного изображения базовой sc-дуги, соединяющего изображение sc-узла, обозначающего класс обозначений *постоянных сущностей* с изображением рассматриваемого sc-узла $\boldsymbol{vi}$ (см. *Файл. Изображение спецификации sc.g-элемента средствами Ядра SCg-кода и Первого расширения Ядра SCg-кода*).

Общепринятая запись данного факта выглядит следующим образом:

"*sc-константа* $\ni \boldsymbol{vi}$; *постоянная сущность* $\ni \boldsymbol{v_i}$;"

- *Константные постоянные sc-ребра* в конструкциях SCg-кода изображаются в виде двойной линии, каждая из которых имеет толщину примерно $d/7$, а расстояние между ними равно примерно $3d/7$.
- *Константные постоянные sc-дуги* изображаются в виде такой же двойной линии, но со стрелочкой. Все *базовые sc-дуги*, а также все sc-узлы, имеющие содержимое, по определению являются *константными постоянными sc-элементами*.
- *Константные sc.g-узлы*, изображаемые окружностями диаметра $3d$ и толщиной границы $d/5$, обозначают *константные постоянные сущности*, о которых мало что известно, но известно то, что они не являются парами (то есть множествами, *мощность** которых равна 2) и, следовательно, не могут быть изображены в виде sc.g-дуг или sc.g-ребер. Но, если при этом об обозначаемой *константной постоянной сущности* ($\boldsymbol{vi}$) известно, что она является классом сущностей, то явное указание принадлежности sc-элемента *vi* всевозможных классов можно заменить на специальное графическое изображение sc-элемента *vi*, предполагаемое

указанную принадлежность. Это приводит к расширению *Алфавита SCg-кода^* (см. *Примеры sc.g-текстов, трансформируемых по Второму направлению расширения Ядра SCg-код*).

Аналогичным образом вводятся:
- *sc.g-узел*, являющийся изображением *класса*;
- *sc.g-узел*, являющийся изображением *класса классов*;
- *sc.g-узел*, являющийся изображением *отношения*;
- *sc.g-узел*, являющийся изображением *ролевого отношения*;
- *sc.g-узел*, являющийся изображением *структуры*;
- *sc.g-узел*, являющийся изображением *небинарной связки*;
- *sc.g-узел*, являющийся изображением *первичной сущности* (терминальной сущности, которая не является множеством, а также файлом, хранимым в памяти ostis-системы).

Важное место среди константных постоянных сущностей занимают *константные постоянные пары принадлежности*, обозначаемое соответствующими *sc.g-дугами*. Такие пары принадлежности и обозначающие их sc.g-дуги бывают позитивными, негативными и нечеткими. Константная постоянная позитивная sc.g-дуга принадлежности есть ничто иное, как *базовая sc.g-дуга*. Константная постоянная негативная sc.g-дуга принадлежности изображается в виде *базовой sc.g-дуги*, перечеркнутой штриховыми черточками. Константная постоянная нечеткая sc.g-дуга принадлежности изображается в виде "недочеркнутой" *базовой sc.g-дуги*, с каждой стороны которой отображаются штрихи, по длине равные половине от длины штрихов, которыми перечеркнута *константная постоянная негативная sc.g-дуга*.

Третье направление расширения Ядра SCg-кода

Третье направление расширения Ядра SCg-кода — это расширение его алфавита путем введения дополнительных sc.g-элементов, обозначающих *константные временные сущности* различного вида. Признаком sc.g-элементов, обозначающих *константные временные сущности* являются точечные линии (линии, состоящие из точек, размер которых равен размеру изображаемой линии и которые близко расположены друг к другу на расстоянии, равном половине их размера), с помощью которых рисуются окружности при изображении sc-узлов, а также линии при изображении sc-коннекторов.

Результатом *Третьего направления расширения Ядра SCg-кода* является введение следующих видов *sc.g-элементов* (см. *Примеры sc.g-текстов, трансформируемых по Третьему направлению расширения Ядра SCg-кода*).

Четвертое направление расширения Ядра SCg-кода

Четвертое направление расширения Ядра SCg-кода — это расширение его алфавита путем введения дополнительных элементов, обозначающих *переменные постоянные сущности* различного вида. Признаком sc.g-элементов, обозначающих сущности указанного класса, являются квадратики для изображения обозначений *переменных постоянных сущностей*, не являющихся бинарными связями, а также пунктирные и штрих-пунктирные линии для изображения *переменных постоянных бинарных связей*.

Подчеркнем, что *переменные постоянные сущности* могут отличаться друг от друга по характеру их *области значений**. Этими значениями в общем случае могут быть как *константные постоянные сущности*, так и *переменные постоянные сущности*. В любом случае, значение *переменной сущности* является либо *константной сущностью*, либо *переменной сущностью*. Если каждое значение переменной является константой, то такую переменную будем называть *переменной первого уровня*. Если каждое значение переменной является *переменной первого уровня*, то такую переменную будем называть *переменной второго уровня*.

переменная постоянная сущность первого уровня (первичная sc-переменная), не являющаяся бинарной связью — это переменная, каждым значением которой является *константная постоянная сущность*, не являющаяся бинарной связью. Такая переменная изображается квадратиком, который ориентирован по вертикали и горизонтали.

переменная постоянная сущность второго уровня (вторичная sc-переменная), не являющаяся бинарной связью, изображается квадратиком, повернутым на 45°.

Указанная выше семантика таких изображений приписывается по умолчанию. Это означает, что, если обозначаемая sc-переменная имеет более сложную структуру области ее значений (является sc-переменной третьего и выше уровня или sc-переменной, значения которой имеют различный логический уровень), то эта область должна быть специфицирована явно, при этом такая sc-переменная в SCg-коде изображается так же, как первичная sc-переменная.

Пятое направление расширения Ядра SCg-кода

Пятое направление расширения Ядра SCg-кода — это расширение его алфавита путем введения дополнительных *sc.g-элементов*, обозначающих *переменные временные сущности* различного вида. Указанные дополнительные *sc.g-элементы* аналогичны тем, которые введены в рамках *Четвертого направления расширения Ядра SCg-кода*, и

отличаются только тем, что в *Пятом направлении расширении Ядра SCg-кода* речь идет о переменных временных сущностях, а в *Четвертом направлении расширения Ядра SCg-кода* — о переменных постоянных сущностях.

Шестое направление расширения Ядра SCg-кода

Шестое направление расширения Ядра SCg-кода — это введение в SCg-код *sc.g-контуров* и *sc.g-шин* как средств структуризации sc.g-текстов и повышения наглядности при их размещении. Подчеркнем, что и sc.g-контуры, и sc.g-шины, и sc.g-рамки являются специальными видами sc.g-элементов. При этом sc.g-контуры и sc.g-рамки являются sc.g-ограничителями (ограничителями SCg-кода).

Каждый ***sc.g-контур*** изображается (в 2D-модификации) в виде замкнутой ломаной линии со скругленными изломами, ограничивающей некоторый фрагмент sc.g-текста и обозначает множество всех sc-элементов, sc.g-изображения которых оказались внутри этого контура. Толщина указанной линии составляет примерно $\mathbf{0.4}\boldsymbol{d}$, где $\boldsymbol{d}$ — диаметр *sc.g-узла общего вида*.

Обозначение *множества sc-элементов*, изображаемое *sc.g-контуром*, может быть как константным, так и переменным. Соответственно этому линия, изображающая *sc.g-контур* может быть:
- сплошной непунктирной линией,
- точечной непунктирной линией,
- сплошной пунктирной линией,
- точечной пунктирной линией.

Семантическим эквивалентом *sc.g-контуру* является *sc.g-узел, обозначающий структуру*. Использование *sc.g-контура* вместо указанного *sc.g-узла* исключает необходимость явно изображать *sc-дуги принадлежности*, выходящие из этого *sc.g-узла*. Это существенно повышает уровень наглядности *sc.g-текста*.

Если представленный внутри *sc.g-контура* текст не является *sc.g-текстом*, то считается, что на самом деле внутренностью *sc.g-контура* является *sc.g-текст*, являющийся результатом перевода предоставленного текста в *SCg-код*.

Каждая ***sc.g-шина*** представляет собой замкнутую или незамкнутую линию толщиной примерно равной диаметру *sc.g-узла общего вида*, которая инцидентна только одному *sc.g-элементу* и семантически ему эквивалентна. Идея введения *sc.g-шин* заключается в увеличении «размеров» *sc.g-элементов* для расширения их области инцидентности. Особенно актуально это для *sc.g-узлов*, имеющих большое число инцидентных им *sc.g-коннекторов*.

Седьмое направление расширения Ядра SCg-кода

Седьмое направление синтаксического расширения Ядра SCg-кода — это переход от 2D-изображений *sc.g-текстов* к 3D-изображениям.

Одним из вариантов трехмерного изображения *sc.g-текстов* является следующий:
- все sc.g-узлы изображаются, как и ранее, плоскими графическими примитивами. При изменении точки просмотра они всегда “поворачиваются” параллельно плоскости экрана, но их масштаб (размер на экране) при удалении от точки просмотра уменьшается;
- аналогичным “плоским” образом изображаются *sc.g-рамки* с их “внутренним” содержанием, а также внешние идентификаторы, приписываемые *sc.g-элементам*;
- *sc.g-коннекторы* изображаются непересекающимися линиями в трехмерном пространстве (заметим, что при изображении *sc.g-текстов* на плоскости пересечение *sc.g-коннекторов* часто снижает наглядность, “читабельность” *sc.g-текстов*). Т.е. *sc.g-коннекторы*, которые на плоскости изображаются двойными линиями, в пространстве цилиндрическими, “трубчатыми линиями” с находящейся внутри тонкой, но просвечивающейся осевой линией;
- *sc.g-контур* в пространстве визуализируется несколькими (!) специального вида точками — например там, где есть точки инцидентности *sc.g-контура* с внешними *sc.g-коннекторами*. При этом *sc.g-контур* становится виден только по команде просмотра указываемого контура (указание контура — это указание одной из его точек инцидентности). По этой команде цветом выделяются все граничные точки контура (точки инцидентности) и все внутренние *sc.g-элементы* контура. Если просматривается несколько контуров, то используется несколько цветов.

Вторым вариантом 3D-визуализации *sc.g-текстов* является размещение *sc.g-текстов* на параллельных плоскостях (слоях) с “прошивками” между этими слоями, соединяющими синонимичные *sc.g-узлы*, то есть *sc.g-узлы*, имеющие одинаковые приписываемые им внешние идентификаторы. Такой вариант плоской, но многослойной визуализации *sc.g-текстов* дает возможность широко использовать те средства просмотра и редактирования *sc.g-текстов*, которые разработаны для плоской их визуализации(см. *Голенков В.В..СтандОТОП-2021кн*).

§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)

⇒ *подраздел**:
- *Пункт 2.3.3.1. Синтаксис SCs-кода*
- *Пункт 2.3.3.2. Денотационная семантика SCs-кода*
- *Пункт 2.3.3.3. Иерархическое семейство подъязыков, семантически эквивалентных SCs-коду*

⇒ *ключевой знак**:
- *SCs-код*
- *Алфавит SCs-кода^*
- *Синтаксис SCs-кода*
- *Денотационная семантика SCs-кода*
- *Иерархическое семейство подъязыков, семантически эквивалентных SCs-коду*

⇒ *ключевое понятие**:
- *sc.s-предложение*
- *sc.s-ограничитель*
- *sc.s-разделитель*
- *sc.s-модификатор*
- *sc.s-элемент*
- *sc.s-коннектор*
- *sc.s-ребро*
- *sc.s-дуга*

Введение в § 2.3.3.

SCs-код
:= [Semantic Code string]
:= [Язык линейного представления знаний ostis-систем]
:= [Множество всевозможных текстов *SCs-кода*]
:= [Язык внешнего линейного представления конструкций внутреннего языка ostis-систем]

SCs-код представляет собой множество линейных текстов (*sc.s-текстов*), каждый из которых состоит из предложений (*sc.s-предложений*), разделенных друг от друга двойной *точкой с запятой* (разделителем *sc.s-предложений*). При этом *sc.s-предложение* представляет собой последовательность *sc-идентификаторов*, являющихся именами описываемых *сущностей* и разделяемых между собой различными *sc.s-разделителями* и *sc.s-ограничителями*.

Пункт 2.3.3.1. Синтаксис SCs-кода

Алфавит SCs-кода^
:= [Алфавит символов SCs-кода^]
:= [множество символов SCs-кода]
:= [символ, используемый в текстах SCs-кода]
:= [Язык внешнего линейного представления информационных конструкций внутреннего языка ostis-систем]
⇐ *объединение**:
- {• *Алфавит символов, используемых в sc.s-разделителях^*
- *Алфавит символов, используемых в sc.s-ограничителях^*
- *Алфавит символов, используемых в sc-идентификаторах^*
  - ⇐ *объединение**:
    - {• *Алфавит символов, используемых в простых строковых sc-идентификаторах^*
    - *Алфавит символов, используемых в sc-выражениях^*
    - }
- *Алфавит символов, используемых в неоднозначных sc.s-изображениях sc-узлов^*
- }

Алфавит SCs-кода^ строится на основе современных общепринятых наборов символов, что позволяет упростить разработку средств для работы с *sc.s-текстами* с использованием современных технологий.

В состав *sc.s-текстов*, как и в состав текстов любых других языков, являющихся вариантами внешнего отображения текстов *SC-кода*, могут входить различные файлы, в том числе естественно-языковые или даже файлы, содержащие другие *sc.s-тексты*. В общем случае в таких файлах могут использоваться самые разные символы, в связи с чем будем считать, что в *Алфавит SCs-кода^* эти символы не включаются.

Алфавит символов, используемых в *sc.s-разделителях^* состоит из: пробел, точка с запятой, двоеточие, круглый маркер и знак равенства.

Алфавит символов, используемых в *sc.s-разделителях^*, изображающих связь инцидентности sc-элементов, состоит из: “ < ”, “ > ”, “ | ”, “ - ”.

Базовый алфавит символов, используемых в *sc.s-коннекторах^* состоит из: “ ∼ ”, знак подчеркивания, знак равенства, двоеточие, “ < ”, “ > ”, “ — ”, “ | ”, “ / ”.

Расширенный алфавит символов, используемых в *sc.s-коннекторах^* состоит из: “∈”, “∋”, “⊆”, “⊇”, “⊂”, “⊃”, “≤”, “≥”, “⇐”, “⇒”, “←”, “→”, “⇔”.

При необходимости комбинации указанных признаков перечисленные символы комбинируются так, как показано в параграфе “*Описание sc.s-разделителей и sc.s-ограничителей*”.

Как в *Базовом*, так и в *Расширенном Алфавитах sc.s-коннекторов* используются следующие общие признаки, характеризующие тип изображаемого *sc-коннектора*:

- знак подчеркивания как признак изображений переменных *sc-коннекторов* (один знак подчеркивания для *sc-коннекторов*, являющихся первичными *sc-переменными*, два знака подчеркивания для *sc-коннекторов*, являющихся вторичными *sc-переменными (sc-метапеременными)*);
- вертикальная черта “ | ” как признак изображений *негативных sc-дуг принадлежности*;
- косая черта “ / ” как признак изображений *нечетких sc-дуг принадлежности*;
- тильда “ ∼ ” как признак изображений *временных sc-дуг принадлежности*.

Для упрощения процесса разработки исходных текстов *баз знаний* с использованием *SCs-кода* и создания соответствующих средств вводятся два алфавита символов(см. Таблица. Описание изображения sc.s-коннекторов в Базовом и Расширенном алфавите SCg-кода^). *Базовый алфавит символов, используемых в sc.s-коннекторах^* включает только символы, входящие в переносимый набор символов и имеющиеся на стандартной современной клавиатуре. Таким образом, для разработки исходных текстов баз знаний, использующих только *Базовый алфавит символов, используемых в sc.s-коннекторах^* достаточно обычного текстового редактора. *Расширенный алфавит символов, используемых в sc.s-коннекторах^* включает также дополнительные символы, которые позволяют сделать sc.s-тексты (и sc.n-тексты) более читабельными и наглядными. Для визуализации и разработки sc.s-текстов с использованием расширенного алфавита требуется наличие специализированных средств.

Таблица. Описание изображения sc.s-коннекторов в Базовом и Расширенном алфавите SCg-кода^
=

Класс *sc-элементов*	Изображение *sc.sконнектора* в Расширенном алфавите		Изображение *sc.sконнектора* в Базовом алфавите	
константная постоянная позитивная sc-дуга принадлежности	∋	∈	->	<-
константная постоянная негативная sc-дуга принадлежности	∌	∉	-|>	<|-
константная постоянная нечеткая sc-дуга принадлежности	/∋	∈/	-/>	</-
константная временная позитивная sc-дуга принадлежности	~∋	∈~	~>	<~
константная временная негативная sc-дуга принадлежности	~∌	∉~	~|>	<|~
константная временная нечеткая sc-дуга принадлежности	~/∋	∈/~	~/>	</~
переменная постоянная позитивная sc-дуга принадлежности	_∋	_∈	_->	<-_
переменная постоянная негативная sc-дуга принадлежности	_∌	∉_	_-|>	<|-_
переменная постоянная нечеткая sc-дуга принадлежности	_/∋	∈/_	_-/>	</-_

Таблица. Описание изображения sc.s-коннекторов в Базовом и Расширенном алфавите SCg-кода^ (продолжение)

=

Класс *sc-элементов*	Изображение *sc.sконнектора* в Расширенном алфавите		Изображение *sc.sконнектора* в Базовом алфавите	
переменная временная позитивная sc-дуга принадлежности	_~Э	∈~_	_~>	<~_
переменная временная негативная sc-дуга принадлежности	_~∌	∉~_	_~|>	<|~_
переменная временная нечеткая sc-дуга принадлежности	_~/Э	∈/~_	_~/>	</~_
метапеременная постоянная позитивная sc-дуга принадлежности	__Э	∈__	__->	<-__
метапеременная постоянная негативная sc-дуга принадлежности	__∌	∉__	__-|>	<|-__
метапеременная постоянная нечеткая sc-дуга принадлежности	__/Э	∈/__	__-/>	</-__
метапеременная временная позитивная sc-дуга принадлежности	__~Э	∈~__	__~>	<~__
метапеременная временная негативная sc-дуга принадлежности	__~∌	∉~__	__~|>	<|~__
метапеременная временная нечеткая sc-дуга принадлежности	__~/Э	∈/~__	__~/>	</~__

Алфавит символов, используемых в *sc.s-ограничителях^* состоит из: “(”, “)”, “*”.

Алфавит символов, используемых в неоднозначных *sc.s-изображениях sc-узлов^* состоит из: “{”, “}”, “-”, “!”, “ [”, “] .

Описание sc.s-разделителей и sc.s-ограничителей

sc.s-разделитель и ***sc.s-ограничитель*** являются важными элементами *SCs-кода*.

Существует *sc.s-разделитель*, используемый для структуризации *sc.s-предложений* и *sc.s-разделитель sc.s-предложений*.

sc.s-разделить — разделитель, используемый в *sc.s-текстах*. *sc.s-разделитель* разбивается на:

- *sc.s-разделитель*, используемый для структуризации *sc.s-предложений*.
  - Разделяет *sc-идентификатор бинарного отношения* и второй компонент одной из связок этого отношения в случае, если указанное бинарное отношение и его связка связаны *константной sc-дугой принадлежности*. Представляется в виде двоеточия.
  - Разделяет *sc-идентификатор бинарного отношения* и второй компонент одной из связок этого отношения в случае, если указанное бинарное отношение и его связка связаны *переменной sc-дугой принадлежности*. Представляется в виде двойного двоеточия.
- *sc.s-разделитель sc.s-предложений*, представляется в виде двойной точки с запятой.

sc.s-ограничитель представляется в виде: (![(∗]!∪![∗)]!)

Круглые скобки со звездочкой ограничивают присоединенные *sc.s-предложения*, которые, в свою очередь, могут иметь в своем составе другие присоединенные *sc.s-предложения*.

Также существует ***sc.s-коннектор***. Типология *sc.s-коннекторов* полностью соответствует типологии *sc.g-коннекторов*, и, тем более, *sc-коннекторов*, так как она учитывает устоявшиеся традиции изображения связок целого ряда конкретных отношений.

sc.s-коннектор
:= [изображение *sc-коннектора* во внешнем тексте SCs-кода или SCn-кода]
⊂ *sc.s-разделитель*

Выделяют следующие *sc.s-коннекторы*:
- *ориентированный sc.s-коннектор*,
- *неориентированный sc.s-коннектор*;
- *sc.s-коннектор*, соответствующий *sc.g-дуге принадлежности*,
- *sc.s-коннектор*, соответствующий *sc.g-коннектору*, который не является *sc.g-дугой*.

Типология *sc.s-коннекторов* полностью соответствует типологии *sc.g-коннекторов*, и, тем более, *sc-коннекторов*, так как она учитывает устоявшиеся традиции изображения связок целого ряда конкретных отношений.

На множестве *sc-элементов* задано бинарное ориентированное отношение инцидентности sc-элементов, а так же подмножество этого отношения — отношение инцидентности входящих *sc-дуг*, каждая пара которого связывает sc-дугу с тем sc-элементом, в который она входит. В *SC-коде sc-коннекторы* могут соединять между собой не только *sc-узел* с *sc-узлами*, но и *sc-узел* с *sc-коннектором* и даже *sc-коннектор* с *sc-коннектором*. В последнем случае, указывая инцидентность sc-коннекторов, необходимо уточнить, какой из них является соединяемым (связываемым), а какой-соединяющим (связующим). Поэтому отношение инцидентности, заданное на множестве sc-элементов является ориентированным. Первый компонент пары этого отношения — связующий *sc-коннектор*, а второй — связуемый *sc-элемент*. Очевидно, что связующий *sc-элемент* всегда является *sc-коннектором*, а *sc-узел* может быть только связуемым.

sc.s-разделитель, изображающий связь инцидентности *sc-элементов* разбивается на:
- знак инцидентности “правого” *sc-коннектора* — знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится справа, изображается в виде “ ⊢ ”;
- знак инцидентности “левого” *sc-коннектора* — знак инцидентности *sc-коннектора*, *sc-идентификатор* которого находится слева, изображается в виде “ ⊣ ”;
- знак инцидентности входящей *sc-дуги* справа — знак инцидентности *sc-дуги*, *sc-идентификатор* который находится справа, изображается в виде “ | < ”;
- знак инцидентности входящей *sc-дуги* слева — знак инцидентности *sc-дуги*, *sc-идентификатор* который находится слева, изображается в виде “ > | ”.

sc.s-разделитель, изображающий связь инцидентности sc-элементов
⇒ *разбиение**:
  {• *знак инцидентности “правого” sc-коннектора*
    := [знак инцидентности sc-коннектора, sc-идентификатор которого находится справа]
    = **![|-]!**
  • *знак инцидентности “левого” sc-коннектора*
    := [знак инцидентности sc-коннектора, sc-идентификатор которого находится слева]
    = **![-|]!**
  • *знак инцидентности входящей sc-дуги справа*
    := [знак инцидентности sc-дуги, sc-идентификатор который находится справа]
    = **![|<]!**
  • *знак инцидентности входящей sc-дуги слева*
    := [знак инцидентности sc-дуги, sc-идентификатор который находится слева]

= **![>|]!**
}

Указанные *sc.s-разделители* с точки зрения синтаксической структуры *sc.s-предложений* аналогичны *sc.s-коннекторам*, но с точки зрения их денотационной семантики в отличие от *sc.s-коннекторов* они не являются изображениями соответствующих sc-коннекторов.

Знак равенства является *sc.s-разделителем* двух *sc-идентификаторов*, которые идентифицируют (именуют) одну и ту же сущность и, соответственно, являются *sc-идентификаторами** (внешними уникальными изображениями) одного и того же *sc-элемента*. При этом из указанных двух *sc-идентификаторов* чаще всего один является простым *sc-идентификатором*, а второй — *sc-выражением*. Реже оба эти *sc-идентификатора* являются *sc-выражениями*. И совсем редко оба они являются *простыми sc-идентификаторами*. Последнее обозначает то, что оба эти *sc-идентификатора* являются основными *sc-идентификаторами** одного и того же *sc-элемента*.

Пример: *SC-код* = sc.s-текст;;

Здесь первый *sc-идентификатор* является *именем собственным*, а второй — *именем нарицательным*.

При трансляции *sc.s-текста* в *SC-код* знаку равенства на некотором этапе может быть поставлено в соответствие *sc-ребро*, принадлежащее отношению *синонимии** *sc-элементов*, идентифицируемых *sc-идентификаторами*, связанными знаком равенства. Но на последующем этапе указанное *sc-ребро* удаляется, а связанные им *sc-элементы* склеиваются. Таким образом *sc-ребро*, принадлежащее отношению *синонимии** *sc-элементов*, имеет не только *денотационную*, но и *операционную семантику*.

Знак равенства с включением — изображение *sc-дуги*, принадлежащей отношению погружения*, связывающей два *sc-узла*, обозначающих *sc-тексты*, первый из которых является погружающим, а второй (в который указанная *sc-дуга* входит) является погружаемым, вводимым в состав первого *sc-текста*.

sc-дуга, принадлежащая отношению погружения*, интерпретируется как команда погружения одного sc-текста в состав другого. При выполнении этой команды (1) все sc-элементы погружаемого *sc-текста* становятся элементами, принадлежащими погружающему sc-тексту, (2) все синонимичные *sc-элементы*, оказавшиеся в составе погружающего *sc-текста*, склеиваются, (3) *sc-узел*, обозначающий погружаемый *sc-текст*, а так же спецификация этого *sc-текста* (включая перечень всех его *sc-элементов*) погружается в историю эволюции базы знаний вместе со спецификацией события погружения рассматриваемого *sc-текста* в состав базы знаний.

Указанные *sc.s-коннекторы* отличаются от остальных *sc.s-коннекторов* тем, что они и соответствующие им *sc-коннекторы* (*sc-ребра*, принадлежащих отношению синонимии *sc-элементов* и *sc-дуги*, принадлежащие отношению погружения одного *sc-текста* в состав другого) имеют не только *денотационную*, но и *операционную семантику*, так как являются командами склеивания и командами погружения.

Описание sc.s-предложений

sc.s-предложение
:= [минимальный семантически целостный фрагмент sc.s-текста]
:= [минимальный sc.s-текст]

sc.s-предложение, (1) состоящее или из двух *sc-идентификаторов*, соединенных между собой *sc.s-коннектором*, или из трех *sc-идентификаторов*, разделенных *sc.s-разделителями, изображающими связь инцидентности sc-элементов*, и (2) завершающееся *двойной точкой с запятой*.

Нетрудно заметить, что *простые sc.s-предложения* по сути аналогичны триплетам языка RDF (RDF-триплетам), за тем исключением, что *простое sc.s-предложение* можно “развернуть” при помощи *Операции конверсии sc.s-предложений** не меняя при этом его смысл, а RDF-триплет нельзя. Это является одной из причин, по которой, в отличие от RDF-триплетов, в простых sc.s-предложениях *sc.s-коннекторы* и *sc.s-разделители, изображающие связь инцидентности sc-элементов* не могут быть опущены, поскольку они в том числе показывают направление изображаемой ими связи между sc-элементами.

Признаком завершения любого *sc.s-предложения*, то есть последними его символами является *двойная точка с запятой*, которую, следовательно, можно считать разделителем *sc.s-предложений*.

Выделяют следующие операции над sc.s-предложениями:

- ***Операция конверсии sc.s-предложения****

  Каждое *sc.s-предложение* (в том числе, и *простое sc.s-предложение*) можно преобразовать в семантически эквивалентное ему *sc.s-предложение* путем конверсии (“разворота”) цепочки компонентов *sc.s-предложения*. Так, например, при конверсии (“развороте”) простого *sc.s-предложения* (1) первый его *sc-идентификатор* (первый компонент этого *sc.s-предложения*) становится третьим компонентом конвертированного *sc.s-предложения*, (2) второй его *sc-идентификатор* (третий компонент исходного *sc.s-предложения*)

становится первым компонентом “конвертированного” *sc.s-предложения* и (3) второй компонент исходного *sc.s-предложения* (*sc.s-коннектор* или *sc.s-разделитель, изображающий связь инцидентности sc-элементов*, соединяющий указанные выше компоненты) остается вторым компонентом конвертированного *sc.s-предложения*, но меняет направленность (“$\ni$” заменяется на “$\in$” и наоборот, “$\supset$” на “$\subset$” и наоборот, “$\Rightarrow$” на “$\Leftarrow$” и наоборот и так далее).

- ***Операция присоединения sc.s-предложения****

  Операция соединения двух *sc.s-предложений* при совпадении последнего компонента первого предложения с первым компонентом второго. В результате выполнения данной операции:
  - первый компонент второго *sc.s-предложения* удаляется;
  - оставшаяся часть второго предложения окружается *sc.s-ограничителем* присоединенных предложений ("(*" и "*)"). Разделитель *sc.s-предложений* (";;") также попадает внутрь указанного ограничителя;
  - полученная конструкция помещается между последним компонентом первого предложения и разделителем *sc.s-предложений*, которым заканчивалось первое предложение;
  - второе предложение, таким образом, становится *присоединенным sc.s-предложением*.

  Аналогичным образом к любому присоединенному *sc.s-предложению* могут "пристыковываться" другие присоединенные *sc.s-предложения*, в общем случае уровень такой вложенности не ограничен.

  Присоединенные *sc.s-предложения* используются для того, чтобы продолжить спецификацию какого-либо sc-элемента, *sc-идентификатор* которого является последним компонентом в рамках какого-либо sc.s-предложения, не начиная при этом нового *sc.s-предложения* и, таким образом, не дублируя указанный sc-идентификатор. Внутрь присоединенных sc.s-предложений также могут встраиваться другие присоединенные *sc.s-предложения*, в общем случае уровень вложенности таких предложений не ограничен. Таким образом присоединенные *sc.s-предложения* описывают "ветвление" *sc.s-предложений*, при этом точками такого "ветвления" выступают *sc-идентификаторы*, входящие в состав этих *sc.s-предложений*.

  Благодаря введению *присоединенных sc.s-предложений* появляется возможность любой *sc-текст* изобразить в виде одного *sc.s-предложения*, содержащего необходимое количество *присоединенных sc.s-предложений*. Таким образом, *SCs-код* по выразительной мощности становится эквивалентным *SCn-коду*.
- ***Операция слияния sc.s-предложений****

  Операция присоединения *простого sc.s-предложения* к *sc.s-предложению*, у которого последний sc.s-коннектор совпадает с *sc.s-коннектором простого sc.s-предложения*, а предшествующий указанному *sc.s-коннектору* sc-идентификатор совпадает с первым sc-идентификатором простого sc.s-предложения.

  В результате выполнения этой операции совпадающие *sc-идентификаторы* и *sc.s-коннекторы* соединяемых *sc.s-предложений* "склеиваются" , а последние *sc-идентификаторы* соединяемых *sc.s-предложений* становятся последними компонентами объединенного *sc.s-предложения*, разделенными *точкой с запятой*. Аналогичным образом можно присоединять сколько угодно простых *sc.s-предложений*.
- ***Операция разложения sc.s-предложений на простые sc.s-предложения****

  Каждое *sc.s-предложение* можно разложить на множество *простых sc.s-предложений*, то есть представить в виде последовательности *простых sc.s-предложений*.
- ***Операция разложения sc.s-предложений на простые sc.s-предложения с sc.s-разделителем, изображающим связь инцидентности sc-элементов****

  Каждое *sc.s-предложение* (в том числе и *простое sc.s-предложение* с *sc.s-коннектором*) можно представить в виде семантически эквивалентной последовательности *простых sc.s-предложений* с *sc.s-разделителем, изображающим связь инцидентности sc-элементов*.

  Данная операция осуществляет однозначное (!) формирование множества *простых sc.s-предложений* указанного вида.

Операции, заданные на множестве *sc.s-предложений* можно разделить на три группы:
- группа операций конверсии *sc.s-предложений*, состоящая из одной операции;
- группа операций соединения *sc.s-предложений*;
- группа операций декомпозиции *sc.s-предложений* и, в частности, операций разложения *sc.s-предложений*.

компонент sc.s-предложения*

Каждое *sc.s-предложение* представляет собой последовательность (1) *sc-идентификаторов*, (2) *sc.s-коннекторов* или *sc.s-разделителей*, изображающих связь инцидентности *sc-элементов*, (3) *точек с запятыми*, (4) *ограничителей присоединенных sc.s-предложений*, завершаемая *двойной точкой с запятой*. При этом непосредственно соседствовать друг с другом не могут ни *sc-идентификаторы*, ни *sc.s-коннекторы*, ни, очевидно, *точки с запятыми* и *ограничители присоединенных sc.s-предложений*.
Между *sc-идентификаторами* в рамках *sc.s-предложения* может находиться либо *точка с запятой*, либо *sc.s-*

коннектор, либо *sc.s-разделитель*, изображающий связь инцидентности *sc-элементов*. Слева и справа от *sc.s-коннектора* и от *sc.s-разделителя*, изображающего связь инцидентности *sc-элементов*, должны находиться *sc-идентификаторы*.

Указанные *sc-идентификаторы*, *sc.s-коннекторы* и *sc.s-разделители*, изображающие связь инцидентности *sc-элементов*, считаются компонентами соответствующего *sc.s-предложения*. Понятие "быть компонентом sc.s-предложения" является относительным понятием (отношением), так как в состав некоторых компонентов *sc.s-предложения* (в состав *sc-идентификаторов*, являющихся *sc.s-выражениями*, ограничиваемыми фигурными или квадратными скобками) могут входить других *sc.s-предложения*, состоящие из своих компонентов(см. *Голенков В.В..СтандОТОП-2021кн*).

sc.s-модификатор*

Это дополнительный вид компонентов *sc.s-предложений*. Каждый *sc.s-модификатор*, являющийся компонентом некоторого *sc.s-предложения*, представляет собой *sc-идентификатор*, обозначающий множество (чаще всего, отношение), которому принадлежит sc-коннектор, изображенный *sc.s-коннектором*, который предшествует указанному *sc-идентификатору*. Признаком *sc.s-модификатора* является *двоеточие* (или *двойное двоеточие*), которое ставится после *sc.s-модификатора* и отделяет его либо от следующего за ним другого *sc.s-модификатора* для этого же *sc.s-коннектора*, либо от следующего за ним *sc-идентификатора*, соответствующего sc-элементу, который инцидентен sc-коннектору, изображенному *sc.s-коннектором*, находящимся левее рассматриваемого *sc-идентификатора* после одного или нескольких *sc.s-модификаторов*. Обычное ("одинарное") *двоеточие* обозначает, что sc-элемент, изображенный соответствующим sc.s-модификатором, связан с sc-коннектором, изображенным левее этого sc.s-модификатора, *базовой sc-дугой* (*константной постоянной позитивной sc-дугой принадлежности*), *двойное двоеточие* обозначает, что указанные элементы связаны *переменной постоянной позитивной sc-дугой принадлежности*.

sc.s-текст
:= [конкатенация *sc.s-предложений*]
:= [последовательность *sc.s-предложений*, разделяемых *двойными точками с запятой*]

sc.s-предложение является минимальным *sc.s-текстом*. Смысл *sc.s-текста* (а также *sc.s-текста, включенного в структуру* не зависит от порядка *sc.s-предложений* в этих *sc-текстах*. Т.е. перестановка *sc.s-предложений* в рамках таких sc.s-текстов смысла этих sc.s-текстов не меняет (то есть приводит к семантически эквивалентным *sc.s-текстам*), но сильно влияет на трудоемкость человеческого восприятия (на "читабельность") этих текстов.

Пункт 2.3.3.2. Денотационная семантика SCs-кода

Ниже приведены таблицы, которые описывают соотношение между *sc.s-коннекторами*, Алфавит которых описан выше, и соответствующими им изображениями *sc.g-коннекторов*(см. Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-дугам принадлежности^ и Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-коннекторам, которые не являются sc.g-дугами принадлежности^).

Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-дугам принадлежности^
=

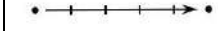
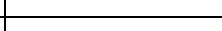
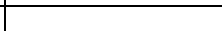
Класс *sc-элементов*	Изображение *sc.g-коннектора*	Изображение *sc.sконнектора* в Расширенном алфавите		Изображение *sc.sконнектора* в Базовом алфавите	
константная постоянная позитивная sc-дуга принадлежности		∋	∈	–>	<–
константная постоянная негативная sc-дуга принадлежности		∌	∉	–|>	<|–
константная постоянная нечеткая sc-дуга принадлежности		/∋	∈/	–/>	</–
константная временная позитивная sc-дуга принадлежности		~∋	∈~	~>	<~
константная временная негативная sc-дуга принадлежности		~∌	∉~	~|>	<|~
константная временная нечеткая sc-дуга принадлежности		~/∋	∈/~	~/>	</~
переменная постоянная позитивная sc-дуга принадлежности		_∋	_∈	_–>	<–_
переменная постоянная негативная sc-дуга принадлежности		_∌	∉_	_–|>	<|–_
переменная постоянная нечеткая sc-дуга принадлежности		_/∋	∈/_	_–/>	</–_
переменная временная позитивная sc-дуга принадлежности		_~∋	∈~_	_~>	<~_
переменная временная негативная sc-дуга принадлежности		_~∌	∉~_	_~|>	<|~_
переменная временная нечеткая sc-дуга принадлежности		_~/∋	∈/~_	_~/>	</~_
метапеременная постоянная позитивная sc-дуга принадлежности		__∋	∈__	__–>	<–__
метапеременная постоянная негативная sc-дуга принадлежности		__∌	∉__	__–|>	<|–__
метапеременная постоянная нечеткая sc-дуга принадлежности		__/∋	∈/__	__–/>	</–__
метапеременная временная позитивная sc-дуга принадлежности		__~∋	∈~__	__~>	<~__
метапеременная временная негативная sc-дуга принадлежности		__~∌	∉~__	__~|>	<|~__
метапеременная временная нечеткая sc-дуга принадлежности		__~/∋	∈/~__	__~/>	</~__

Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-коннекторам, которые не являются sc.g-дугами принадлежности^

=

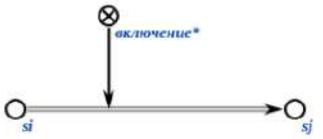
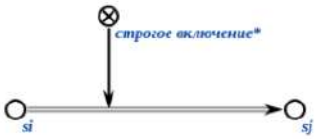
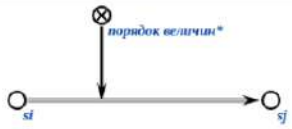
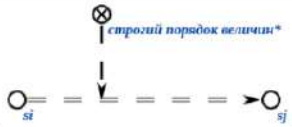
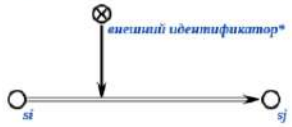
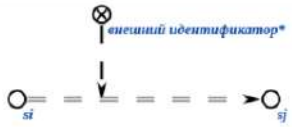
Изображение *sc-коннектора* в SCg	Изображение *sc.s-коннектора* в Расширенном алфавите		Изображение *sc.s-коннектора* в Базовом алфавите	
	↔		<>	
	→	←	»	«
	⇔		<=>	
	⇒	⇐	=>	<=
	~⇔		~<=>	
	~⇒	⇐~	~=>	<=~
	_⇔		_<=>	
	_⇒	⇐_	_=>	<=_
	_~⇔		_~<=>	
	_~⇒	⇐~_	_~=>	<=~_
	__⇔		__<=>	
	__⇒	⇐__	__=>	<=__
	__~⇔		__~<=>	
	__~⇒	⇐~__	__~⇒	⇐~__
	⊇	⊆		
	_⊇	⊆_		
	⊃	⊂		
	_⊃	⊂_		

Таблица. Алфавит sc.s-коннекторов, соответствующих sc.g-коннекторам, которые не являются sc.g-дугами принадлежности^(продолжение)
=

Изображение *sc-коннектора* в SCg	Изображение *sc.s-коннектора* в Расширенном алфавите		Изображение *sc.s-коннектора* в Базовом алфавите
	≥	≤	
	_≥	_≤	
	>	<	
	_>	<_	
	:=		
	_:=		
	=		
	⊃=	=⊂	

Ниже приведены *примеры синтаксической трансформации sc.s-предложений с использованием Расширенного алфавита sc.s-коннекторов и соответствующие семантически эквивалентные конструкции в SCg-коде*.

[*si* ⇒ *включение: ***sj***]
⇒ *синтаксическая трансформация**:
[*si* ⊃ ***sj*]**
⇔ *семантическая эквивалентность**:

[*si* _ ⇒ *включение:: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* _ ⊇ *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* __ ⇒ *включение::: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* __ ⊇ *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* ⇒ *строгое включение: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* ⊃ *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* _ ⇒ *строгое включение:: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* _ ⊃ *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* __ ⇒ *строгое включение::: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* __ ⊃ *sj*]
⇔ *семантическая эквивалентность**:
[

]
[*si* ⇒ *порядок величин: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* ≥ *sj*]
⇔ *семантическая эквивалентность**:
[

]
[*si* _ ⇒ *порядок величин:: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* _ ≥ *sj*]
⇔ *семантическая эквивалентность**:
[

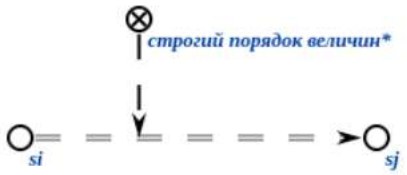
]
[*si* __ ⇒ *порядок величин::: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* __ ≥ *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* ⇒ *строгий порядок величин: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* > *sj*]
⇔ *семантическая эквивалентность**:
[

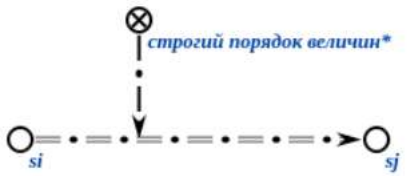
]

[*si* _ ⇒ *строгий порядок величин:: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* _ > *sj*]
⇔ *семантическая эквивалентность**:
[



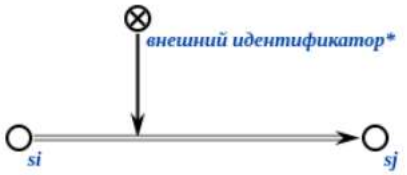
]

[*si* __ ⇒ *строгий порядок величин::: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* __ > *sj*]
⇔ *семантическая эквивалентность**:
[



]

[*si* ⇒ *внешний идентификатор: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* := *sj*]
⇔ *семантическая эквивалентность**:
[



]

[*si* _ ⇒ *внешний идентификатор:: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* _ := *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* __ ⇒ *внешний идентификатор::: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* __ := *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* ⇔ *синонимия: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* = *sj*]
⇔ *семантическая эквивалентность**:
[

]

[*si* ⇒ *погружение: *sj*]**
⇒ *синтаксическая трансформация**:
[*si* ⊃= *sj*]
⇔ *семантическая эквивалентность**:
[

]

Аналогичным образом может быть описана трансформация предложений, содержащих любые классы *sc.s-коннекторов*, за исключением тех классов *sc.s-коннекторов*, которые соответствуют классам *sc-коннекторов*, входящим в *Ядро SC-кода*.

В общем случае *sc-элементы*, инцидентные *sc-коннекторам*, классы которых описаны в данном примере, могут быть как *sc-константами*, так и *sc-переменными* (в том числе *sc-метапеременными*). При этом как *переменному sc-коннектору* может соответствовать *константный sc-узел*, так и *константному sc-коннектору* может соответствовать *переменный sc-узел* (например, если возникает необходимость переменному sc-узлу приписать

*внешний идентификатор**). Последняя ситуация встречается не очень часто и возникает в случае, когда область определения соответствующего *отношения* имеет непустое пересечение с классом *sc-переменных*.

Описание примеров выполнения операций, заданных на множестве sc.s-предложений

С семантической точки зрения *sc.s-предложение* представляет собой описание некоторого маршрута в соответствующем sc-тексте, который является графовой структурой специального вида и структура которого описывается (изображается) с помощью *sc.s-предложений*. Указанный маршрут "проводится" по sc-коннекторам и по связям инцидентности sc-элементов, если маршрут проходит через инцидентные sc-коннекторы. В описании указанного маршрута могут дополнительно указываться множества (чаще всего отношения), которым принадлежат sc-коннекторы, входящие в описываемый маршрут. Кроме того, указанный маршрут в начале и/или в конце может иметь разветвления, когда какой-либо *sc-элемент* одинаково инцидентен нескольким однотипным sc-коннекторам, соединяющим указанный *sc-элемент* с некоторыми другими sc-элементами.

Таким образом каждое указанное разветвление состоит из неограниченного числа ветвей, каждая из которых состоит из одного *sc-коннектора* и одного связываемого им *sc-элемента*.

Пункт 2.3.3.3. Иерархическое семейство подъязыков, семантически эквивалентных SCs-коду

В рамках *SCs-кода* выделяют *Ядро SCs-кода* и *Направления расширения Ядра SCs-кода*.

Ядро SCs-кода
:= **[**Подъязык *SCs-кода*, который использует минимальный набор синтаксических средств, но при этом имеет семантическую мощность, эквивалентную мощности *SCs-кода* в целом**]**

В *Ядре SCs-кода*:
- используются только *простые sc-идентификаторы*, в том числе *sc-идентификаторы внешних файлов ostis-систем* (sc-выражения не используются);
- используются только *sc.s-разделители, изображающие связь инцидентности sc-элементов*, а также sc.s-коннектор, изображающий константную постоянную позитивную пару принадлежности ("∈" и "∋" в Расширенном алфавите и "->" и "<-" в Базовом алфавите). Другие *sc.s-коннекторы* не используются;
- не используются *sc.s-модификаторы* и, соответственно, двоеточия, являющиеся признаком завершения *sc.s-модификаторов*;
- используются только *простые sc.s-предложения*, которые, как следует из вышеуказанных свойств Ядра SCs-кода, либо состоят из двух *простых sc-идентификаторов*, соединяемых sc.s-коннектором, изображающим константную постоянную позитивную пару принадлежности, либо трех *простых sc-идентификаторов*, разделенных *sc.s-разделителями, изображающими связь инцидентности sc-элементов*.

Из перечисленных свойств *Ядра SCs-кода* следует, что для представления (изображения) любого sc-текста средствами *Ядра SCs-кода* необходимо для всех (!) sc-элементов этого sc-текста (кроме константных постоянных позитивных пар принадлежности) построить соответствующие им простые *sc-идентификаторы*, то есть необходимо проименовать все указанные sc-элементы. В свою очередь, тип каждого используемого sc-элемента (кроме константных постоянных позитивных пар принадлежности) задается явно путем указания принадлежности этих элементов соответствующим классам sc-элементов, в том числе классам, входящим в *Ядро SC-кода*.

Как видно из приведенного описания, *Ядро SCs-кода* соответствует *Ядру SCg-кода*, за исключением того, что в *Ядре SCg-кода* нет необходимости именовать все изображаемые *sc-элементы*, а также в *Ядре SCg-кода* присутствуют графические изображения для sc-элементов, принадлежащих соответствующим классам *Ядра SC-кода* и эту принадлежность нет необходимости указывать явно.

Очевидно, что широко практически применять *Ядро SCs-кода* для записи больших фрагментов баз знаний неудобно и неэффективно. Тем не менее, с практической точки зрения *Ядро SCs-кода* может использоваться, например, для обмена информацией со сторонними средствами представления графовых конструкций, рассчитанными на представление информации в виде триплетов (например, RDF-хранилищ). Для обеспечения возможности более широкого практического использования необходимы синтаксические расширения *Ядра SCs-кода* в целях:
- минимизации числа идентифицируемых (именуемых) sc-элементов путем использования *sc-выражений* и ликвидации необходимости идентифицировать (именовать) все (!) *sc-элементы*;
- сокращения текста путем минимизации числа повторений одного и того же *sc-идентификатора* путем соединения *sc.s-предложений*;
- повышение уровня наглядности, "читабельности" *sc.s-текстов*.

Первое направление расширения Ядра SCs-кода
:= **[**Первое направление расширения Ядра SCs-кода и всех иных его расширений**]**

По сравнению с *Ядром SCs-кода* в *Первом направлении расширения Ядра SCs-кода* вместо *sc-идентификато|-ров*, являющихся идентификаторами (именами), которые взаимно однозначно соответствуют синонимичным им (представляемым ими) sc-коннекторам, вводятся *sc.s-коннекторы*, каждый из которых соответствует не одному конкретному *sc-коннектору*, а некоторому классу однотипных sc-коннекторов. Очевидно, что это ликвидирует необходимость каждому sc-коннектору приписывать уникальный *sc-идентификатор*. Кроме того, *Алфавит sc.s-коннекторов^* включает в себя элементы этого Алфавита (классы синтаксически эквивалентных *sc.s-коннекторов*), которые соответствуют всем (!) элементам *Алфавита sc-коннекторов^*, но при этом дополнительно включают в себя и другие элементы *Алфавита sc.s-коннекторов^*, которые соответствуют часто используемым семантически явно выделяемым классам sc-коннекторов. К таким дополнительно вводимым классам *sc.s-коннекторов* относятся *константные sc.s-коннекторы* включения множеств ("⊃" или "⊂"), *переменные sc.s-коннекторы* включения множеств ("_ ⊃" или "⊂ _"), *sc.s-коннектор* синонимии ("="), *sc.s-коннектор* погружения ("=⊂" или "⊃=") и другие.

Заметим, что указанное расширение *Алфавита SCs-кода^ sc.s-коннекторов* аналогично расширенному *Алфавиту sc.g-коннекторов* в *SCg-коде* и ликвидирует необходимость (как и в *SCs-коде*) явно специфицировать (средствами *SCs-кода*) синтаксически выделяемые классы *sc.s-коннекторов*.

Второе направление расширения Ядра SCs-кода

Во *Втором направлении расширения Ядра SCs-кода* вводятся модификаторы *sc.s-коннекторов* (*sc.s-модификаторы*), которые позволяют достаточно компактно дополнительно специфицировать sc-коннекторы, изображаемые (представляемые) соответствующими *sc.s-коннекторами*. Речь идет о такой часто востребованной форме спецификации sc-коннекторов, как указание множества (возможно, нескольких множеств), которому принадлежит специфицируемый sc-коннектор (чаще всего, таким множеством является *бинарное отношение* (в частности, *ролевое отношение*) или *квазибинарное отношение*).

sc.s-модификатор*
∈ *отношение*
:= [относительное понятие]
:= [модификатор sc.s-коннектора*]
sc-идентификатор, который (1) находится либо между *sc.s-коннектором* и *двоеточием*, либо между *двоеточиями* и (2) обозначает множество (чаще всего, отношение), которому принадлежит sc-коннектор, изображаемый ближайшим предшествующим *sc.s-коннектором*. Два подряд идущих двоеточия ("::") обозначают, что указанное множество связано с указанным sc-коннектором *переменной позитивной постоянной sc-дугой принадлежности*.

Очевидно, что, если не использовать *sc.s-модификаторы*, указанного вида спецификация *sc-коннекторов* средствами *SCs-кода* будет выглядеть значительно более громоздкой.

Третье направление расширения Ядра SCs-кода

В *Третьем направлении расширения Ядра SCs-кода* осуществляется переход от использования только *простых sc-идентификаторов* к использованию как *простых sc-идентификаторов*, так и *sc-выражений*, а также к использованию *sc.s-представлений некоторых неидентифицируемых sc-узлов*. Это существенно сокращает число придумываемых *простых sc-идентификаторов*, так как каждое *sc-выражение* в конечном счете — это комбинация *простых sc-идентификаторов*, построенная по правилам, которые достаточно легко семантически интерпретируются. Если проводить аналогию с SCg-кодом, то очевидно, что *sc-выражение*, ограничиваемое фигурными скобками, есть не что иное, как информационная конструкция, ограничиваемая *sc.g-контуром*, а *sc-выражение*, ограничиваемое квадратными скобками есть не что иное, как информационная конструкция, ограничиваемая *sc.g-рамкой*. Отличие здесь заключается в том, что круглыми и квадратными скобками можно ограничивать только линейные информационные конструкции (цепочки символов).

sc.s-представление неидентифицируемого sc-узла
:= [изображение (представление) неидентифицируемого (неименуемого) sc-узла в sc.s-тексте]
:= [sc.s-обозначение неименуемой сущности, не являющейся парой, обозначаемой sc-коннектором]
:= [sc.s-представление sc-узла, не являющееся sc-идентификатором (именем этого sc-узла)]

Если одно и то же обозначение неименуемой сущности встречается в разных *sc.s-предложениях*, то считается, что это обозначения разных сущностей, то есть изображения разных sc-узлов.

Четвертое направление расширения Ядра SCs-кода

В *Четвертом направлении расширения Ядра SCs-кода* осуществляется переход от использования только *простых sc.s-предложений* к использованию также *sc.s-предложений*, построенных с помощью *Операции присоединения sc.s-предложения**. В результате этого, благодаря "склеиванию" одинаковых *sc-идентификаторов*, а также "склеиванию" синтаксически эквивалентных *sc.s-коннекторов* с одинаковыми *sc.s-модификаторами* (несмотря на то,

что эти "склеиваемые" *sc.s-коннекторы* соответствуют разным sc-коннекторам), существенно сокращается число копий используемых *sc-идентификаторов* и *sc.s-коннекторов* с их *sc.s-модификаторами*.

Пятое направление расширения Ядра SCs-кода

В *Пятом направлении расширения Ядра SCs-кода* разрешается использование *присоединенных sc.s-предложений*. В результате этого *sc.s-тексты* становятся более компактными и удобными для восприятия за счет снижения числа дублируемых *sc-идентификаторов* и более широких возможностей их структуризации.

§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)

⇒ *подраздел**:
- *Пункт 2.3.4.1. Синтаксис SCn-кода*
- *Пункт 2.3.4.2. Денотационная семантика SCn-кода*
- *Пункт 2.3.4.3. Язык представления исходных текстов баз знаний на основе языка LaTeX*

⇒ *ключевой знак**:
- *SCn-код*
- *Алфавит SCn-кода^*
- *Синтаксис SCn-кода*
- *Денотационная семантика SCn-кода*

⇒ *ключевое понятие**:
- *страница sc.n-текста*
- *строка sc.n-текста*
- *линия разметки sc.n-текста*
- *sc.n-предложение*
- *sc.n-элемент*
- *sc.n-коннектор*
- *sc.n-ребро*
- *sc.n-дуга*
- *sc.n-контур*
- *sc.n-рамка*

Введение в § 2.3.4.

SCn-код
:= [Язык структурированного представления знаний *ostis-систем*]
:= [Язык внешнего форматированного представления конструкций внутреннего языка *ostis-систем*]

SCn-код является языком структурированного внешнего представления текстов *SC-кода* и представляет собой синтаксическое расширение *SCs-кода*, направленное на повышение наглядности и компактности текстов *SCs-кода*.

SCn-код позволяет перейти от линейных текстов SCs-кода к форматированным и фактически двухмерным текстам, в которых появляется декомпозиция исходного линейного текста SCs-кода на строчки, размещенные “по вертикали”. При этом начало всех строчек текста фиксировано и определяется известным и ограниченным набором правил, что дает возможность использовать это при форматировании sc.n-текста (текста, принадлежащего *SCn-коду*, см. *Голенков В.В..СтандОТОП-2021кн*).

Пункт 2.3.4.1. Синтаксис SCn-кода

*Алфавит SCs-кода^* является также алфавитом символов и *SCn-кода*, то есть *алфавиты** этих языков совпадают.

SCn-код — язык, каждый *текст* которого задается:
- множеством входящих в него *символов*;
- отношением порядка (последовательности) *символов* по "горизонтали";
- отношением порядка(последовательности) *символов* по "вертикали".

sc.n-текст
:= [текст *SCn-кода*]
:= [последовательность предложений *SCn-кода*]
:= [последовательность предложений *SCn-кода*, каждое из которых не является частью какого-либо другого предложения из этой последовательности]

Важной особенностью *SCn-кода* является “двухмерный” характер его текстов. Это проявляется в том, что для каждого фрагмента текста *SCn-кода* важное значение имеет величина отступа от левого края *строчки*.

Символ, входящий в состав *двухмерного текста*, в общем случае может иметь четыре "соседних" *символа*:
- *символ*, находящийся от него слева в рамках той же *строчки*;
- *символ*, находящийся от него справа в рамках этой же *строчки*;
- *символ*, находящийся строго над ним в предыдущей *строчке*;
- *символ*, находящийся строго под ним в следующей *строчке* текста.

Благодаря тому, что в состав *sc.n-текстов* могут входить и *sc.s-тексты*, и *sc.g-тексты* (ограниченные *sc.n-контуром*), *SCn-код* можно считать интегратором различных внешних языков представления знаний. Это дает возможность при визуализации и разработке базы знаний ostis-системы недостатки одного из предлагаемых вариантов внешнего представления sc-текстов (*SCg-кода*, *SCs-кода*, *SCn-кода*) компенсировать достоинствами других вариантов.

страница sc.n-текста
:= [страница, на которой размещается sc.n-текст]

Если *sc.n-текст* является частью какого-либо другого файла, разделяемого на страницы, например, публикации какой-либо части базы знаний, то *sc.n-страницей* считается только часть страницы, на которой изображен *sc.n-текст*, в то время как страница указанного файла может быть больше за счет, например, белых полей по краям страницы, необходимых для последующей распечатки.

строчка sc.n-текста

Максимальное количество символов в строчках *sc.n-текста* для каждого *sc.n-текста* фиксировано и определяется конкретным вариантом размещения *sc.n-текста*. При этом, в зависимости от отступов в рамках конкретного *sc.n-предложения*, строчка *sc.n-текста* может начинаться не с левого края *sc.n-текста* (но всегда с какой-то из вертикальных линий разметки) и иметь произвольную длину, ограничиваемую правой границей *sc.n-страницы*.

линия разметки sc.n-текста
:= [табуляционная линия sc.n-текста]
:= [вертикальная линия разметки sc.n-текста]
:= [вертикальная табуляционная линия]
:= [вертикальная линия, используемая для упрощения восприятия sc.n-текстов и показывающая уровень отступа для компонентов sc.n-предложений]

1-я линия разметки ограничивает левый край *sc.n-страницы*, 2-я линия разметки располагается примерно между 5 и 6 символами строчки и так далее. Расстояние между линиями разметки может меняться в зависимости от размера шрифта, однако в рамках одного *sc.n-текста* всегда остается одинаковым. Общее количество линий разметки ограничивается максимально возможной шириной *sc.n-страницы* в конкретном файле *ostis-системы*, содержащем данный *sc.n-текст*.

следует отличать*
∋ {• *страница sc.n-текста*
• *строчка sc.n-текста*
• *строка*
}

Все компоненты *sc.s-текстов* используются также и в *sc.n-текстах*:
- *sc-идентификаторы*;
- *sc.s-коннекторы*;
- модификаторы *sc.s-коннекторов* с соответствующими разделителями (двоеточиями)
- разделители, используемые в sc-выражениях, обозначающих sc-множества, заданные перечислением элементов с соответствующими разделителями (*точкой с запятой* или *круглым маркером*);
- *круглые маркеры* в перечислениях идентификаторов sc-элементов, связанных однотипными sc-коннекторами с однотипными модификаторами с заданным sc-элементом;

- разделители предложений (двойные точки с запятой) (опускаются при преобразовании sc.s-предложений в sc.n-предложения);
- ограничители присоединенных *sc.s-предложений* (опускаются при преобразовании sc.s-предложений в sc.n-предложения).

В отличие от *sc.s-текстов* в *sc.n-текстах*:
- добавляются новые виды *sc-выражений* (а именно — sc-выражений, имеющих двухмерный характер);
- добавляется новый вид разделителей предложений — пустая строчка;
- меняется размещение предложений с учетом двухмерного характера такого размещения.

В *SCn-коде* по сравнению с *SCs-кодом* добавляются новые виды *sc-выражений*:
- *sc-выражение*, представляющее собой двухмерный *sc.n-текст*, ограниченный *sc.n-контуром* или *sc.n-рамкой*. Каждый *sc.n-контур* изображается условно в виде *открывающей фигурной скобки* и расположенной строго под ней через несколько строчек *закрывающей фигурной скобки*. Внутри указанных скобок (начиная от линии вертикальной разметки, на которой расположены сами скобки, и до правого края *страницы*) размещается sc.n-текст. Полученный sc.n-контур является изображением структуры, являющейся результатом трансляции указанного sc.n-текста в SC-код. Каждая *sc.n-рамка* изображается аналогичным образом, только вместо *фигурных скобок* в ней используются *квадратные скобки*, либо *квадратные скобки* с *восклицательным знаком* (в случае файла-образца);
- *sc-выражение*, представляющее собой двухмерный *sc.g-текст*, ограниченный *sc.n-контуром* или *sc.n-рамкой*.
- *sc-выражение*, представляющее собой ограниченное *sc.n-рамкой* двухмерное графическое изображение *информационной конструкции*, закодированной в некотором *файле ostis-системы*. Такой *информационной конструкцией* может быть таблица, рисунок, фотография, диаграмма, график и многое другое.

Нетрудно заметить, что *sc.n-контур* является, по сути, двухмерным эквивалентом *sc-выражения структуры*, а *sc.n-рамка* — двухмерным эквивалентом *sc-выражения внутреннего файла ostis-системы* или *sc-выражения, обозначающего файл-образец ostis-системы*.

sc.n-рамка
:= [ограничитель изображения файла ostis-системы, используемый в sc.n-предложениях]

Обозначается с помощью квадратных скобок: “ [”, “] ”.

С формальной точки зрения *sc.n-рамка* всегда представляет собой одну *строчку sc.n-текста*. Это означает, что *sc.n-рамка* не может быть синтаксически разделена на части в рамках того *sc.n-текста*, в котором она используется, и внутрь нее не могут вставляться, например, *присоединенные sc.n-предложения* или какой-либо другой текст (за исключением случаев, когда *sc.n-рамка* содержит *sc.n-текст*, но в этом случае указанный *sc.n-текст* все равно будет рассматриваться как целостный внешний файл, а не как фрагмент окружающего его *sc.n-текста*).

sc.n-контур
:= [используемый в sc.n-предложениях ограничитель, являющийся изображением структуры]

Обозначается с помощью фигурных скобок: “ {”, “ }” .

Понятие ***sc.n-предложения*** является естественным обобщением понятия *sc.s-предложения*. Более того, аналогичным для *sc.s-предложений* образом вводятся понятия:
- *простого sc.n-предложения*
- *сложного sc.n-предложения*
- *sc.n-предложения, содержащего присоединенные sc.n-предложения*
- *sc.n-предложения, не содержащего присоединенные sc.n-предложения*
- *присоединенного sc.n-предложения*
- *неприсоединенного sc.n-предложения*

Если каждое *неприсоединенное sc.s-предложение* либо является первым предложением *sc.s-текста*, либо начинается после *разделителя sc.s-предложений* (*двойной точки с запятой*), то каждое *неприсоединенное sc.n-предложение* начинается с начала новой строчки.

Если каждое *присоединенное sc.s-предложение* начинается либо после открывающего ограничителя присоединенных sc.s-предложений (открывающей круглой скобки со звездочкой), либо после *разделителя sc.s-предложений*, то каждое *присоединенное sc.n-предложение* начинается с новой строчки под sc-идентификатором, которым завершается то sc.n-предложение (и соответственно, sc.s-предложение), в которое встраивается данное *присоединенное sc.n-предложение*.

Первый *sc-идентификатор*, входящий в состав *sc.n-предложения* до *sc.s-коннектора* выделяется жирным курсивом; В *sc.n-предложениях двойная точка с запятой* не используется в качестве признака завершения этих

предложений и, соответственно, не используется в качестве разделителя *sc.n-предложений*. Таким разделителем является *пустая строчка*.

Благодаря двухмерности *SCn-кода* появляются более широкие возможности (степени свободы) для наглядного и компактного размещения *sc.n-предложений*.

При оформлении *sc.n-предложения* осуществляется четкая табуляция всех присоединенных к нему *sc.n-предложений*, присоединяемых к исходному "по вертикали". Вертикальная линия табуляции задает левую границу исходного (максимального) sc.n-предложения или левую границу присоединенного *sc.n-предложения*, присоединяемого "по вертикали". Левая граница sc.n-предложения задает начало первого sc-идентификатора, входящего в состав этого sc.n-предложения, а также начало *sc.s-коннектора*, инцидентного указанному sc-идентификатору и размещаемого строго под этим *sc-идентификатором*. Расстояние между вертикальными табуляционными линиями фиксировано и примерно равно максимальной длине *sc.s-коннектора*.

Пункт 2.3.4.2. Денотационная семантика SCn-кода

SCn-код предназначен для представления *sc-графов* в виде отформатированных по заданным правилам последовательностей символов, в которых также могут быть использованы базовые средства гипермедиа, такие как графические изображения, а также средства навигации между частями sc.n-текстов. *SCn-код* имеет много общего с *SCs-кодом* и, за исключением некоторых особенностей, является его двумерным форматированным вариантом.

Поскольку по отношению к *SCn-коду SCs-код* является *синтаксическим ядром языка**, *SCn-код* можно рассматривать как результат интеграции нескольких направлений расширения *SCs-кода*, в основе которых лежат правила синтаксической трансформации *sc.s-текстов* и *sc.n-текстов*, ориентированные на повышение эффективности использования тех возможностей обеспечения наглядности и компактности *sc.n-текстов*, которые открываются при переходе от линейности *sc.s-текстов* к двухмерности *sc.g-текстов*.

Каждый *sc.n-текст* может быть представлен в нескольких вариантах идентификации *sc-элементов* представляемого *sc-графа*:
- с использованием *системных идентификаторов*, которые носят интернациональный характер;
- с использованием *основных идентификаторов** для русскоязычных пользователей;
- с использованием *основных идентификаторов** для англоязычных пользователей.

Каждый *sc.n-текст* представляет собой последовательность *sc.n-статей* (аналог форматированного естественно-языкового текста). Каждая *sc.n-статья*, в свою очередь, представляет собой последовательность *sc.n-предложений*, в начале которой помещается заголовок *sc.n-статьи*, который представляет собой идентификатор ключевого *sc-элемента* того *sc-графа*, который представляется данной *sc.n-статьей*. С семантической точки зрения указанный *sc-граф* является семантической окрестностью, центром которой является указанный ключевой sc-элемент. При этом ключевой sc-элемент некоторой статьи обязательно входит в состав каждого из *sc.n-предложений*, но необязательно является компонентом ключевого (самого первого, считая от начала предложения) *sc-коннектора* данного *sc.n-предложения*. Входящие в *sc.n-статью sc.n-предложения* являются *sc.s-предложениями* либо некоторыми их модификациями(см. *MetacOSTIS-2022эл*)

Разделителями *sc.n-предложений* (как и *sc.s-предложений*) являются двойные точки с запятой. Этот же разделитель отделяет заголовок *sc.n-статьи* от первого предложения этой *sc.n-статьи*. При этом заголовок *sc.n-статьи* можно трактовать как вырожденное *sc.n-предложение*, состоящее только из одного идентификатора.

В отличие от *sc.s-текстов*: в *sc.n-текстах sc.s-коннектор* может быть инцидентен предшествующему *sc-идентификатору* (как простому, так и *sc-выражению*) не только “по горизонтали”, но и “по вертикали”. Для этого *sc.s-коннектор* размещается строго под предшествующим ему sc-идентификатором.

Кроме того “по вертикали” *sc-идентификатор* может быть инцидентен не одному, а нескольким *sc.s-коннекторам*, которые последовательно “по вертикали” размещаются под указанным *sc-идентификатором*. Это позволяет в рамках одного *sc.n-предложения* представлять произвольное число “ответвлений” от каждого sc-идентификатора, то есть произвольное число *sc.s-коннекторов*, инцидентных этому *sc-идентификатору*. Каждый *sc-идентификатор*, включая *sc-выражение*, ограничиваемого фигурными или квадратными скобками, должен размещаться сразу правее вертикальной разметочной линии, если под ним размещается *sc.s-коннектор*.

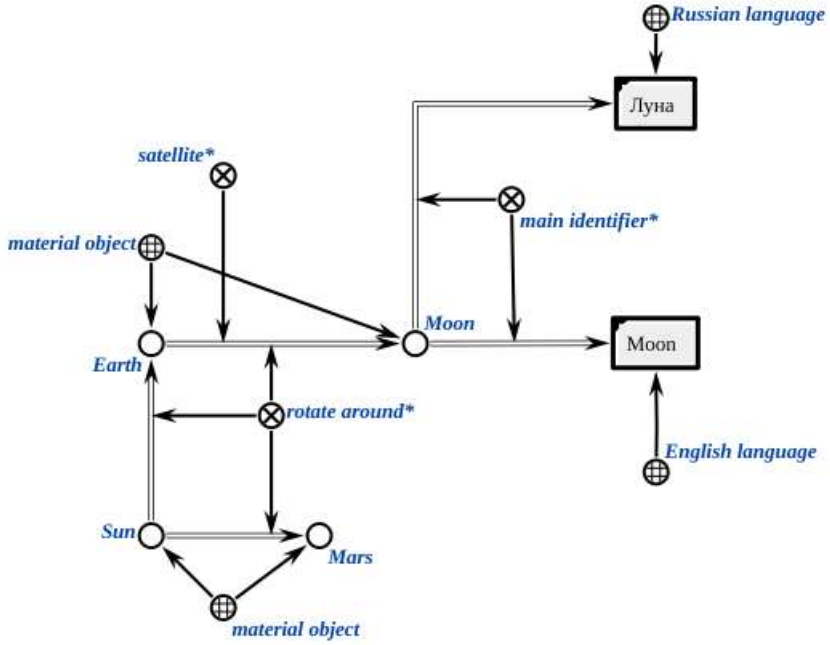
Каждый *sc.s-коннектор* выделяется жирным некурсивным шрифтом и, если он находится под инцидентным ему *sc-идентификатором*, размещается строго между двумя вертикальными разметочными линиями, прижимаясь при этом к левой из этих двух разметочных линий.

Поскольку по отношению к *SCn-коду SCs-код* является *синтаксическим ядром языка**, *SCn-код* можно рассматривать как результат интеграции нескольких направлений расширения *SCs-кода*, в основе которых лежат правила синтаксической трансформации *sc.s-текстов* и *sc.n-текстов*, ориентированные на повышение эффективности

использования тех возможностей обеспечения наглядности и компактности *sc.n-текстов*, которые открываются при переходе от линейности *sc.s-текстов* к двухмерности *sc.g-текстов*.

Рассмотрим фрагмент *sc.g-текста*, изображенного на *Рисунок. Фрагмент sc.g-текста*. На данном фрагменте представлен класс материальных объектов, включающий: Землю, Луну, Солнце, Марс. Материальный объект "Луна"имеет два основных идентификатора на русском и английском языках. "Земля"и "Марс"связаны с "Солнце"с помощью отношения "вращаться вокруг*". "Луна"связана с "Земля"с помощью отношения "спутник*"(см. *Zhmyrko A.Famil oELoNGCSCttLoISRoK-2022art*).

Рисунок. Фрагмент sc.g-текста
=



Любой *sc.g-текст* можно легко представить с помощью *sc.s-текста*. Соответственно, описанный выше фрагмент *sc.g-текста* представлен в *sc.s-тексте* на *Рисунок. Фрагмент sc.s-текста, соответствующий sc.g-тексту*.

Рисунок. Фрагмент sc.s-текста, соответствующий sc.g-тексту
=

```
material object -> Moon; Sun; Mars; Earth;;
Sun => rotate around*: Earth; Mars;;
Earth => rotate around*: Moon;;
Earth => satellite*: Moon;;
Moon
=> main identifier:
    [Луна]
    (* <- Russian language;; *);
    [Moon]
    (* <- English language;; *);;
```

На *Рисунок. Фрагмент sc.n-текста, соответствующий sc.s/sc.g-тексту* продемонстрирован фрагмент вышеуказанного текста в *SCn-коде*.

Рисунок. Фрагмент sc.n-текста, соответствующий sc.s/sc.g-тексту
=

Пункт 2.3.4.3. Язык представления исходных текстов баз знаний на основе языка LaTeX

Одним из удобных вариантов представления исходных текстов баз знаний для различных областей научно-технической деятельности является Язык представления исходных текстов баз знаний на основе языка LaTeX. В его основу положен язык LaTeX, так как:

- LaTeX это популярный общепринятый язык для записи научно-технических текстов;
- LaTeX представляет собой достаточно мощный и легко расширяемый язык;
- LaTeX это достаточно строгий и формальный язык, что позволяет реализовать для него транслятор исходных текстов в базу знаний интеллектуальной системы.

Язык представления исходных текстов баз знаний на основе языка LaTeX был разработан для:

- формирования читабельного текста для публикации всего текста *Стандарта OSTIS* или фрагментов в виде печатных изданий;
- возможности иметь формальный исходный текст *Стандарта OSTIS*, который может быть протранслирован в базу знаний.

Предлагаемый набор команд условно называется scn-tex (см. *SoftwIoSLP-el*), поскольку в его основу положена идея того, чтобы разработчик писал исходный текст максимально приближенно к тому, как он будет видеть результат компиляции этого исходного текста в SCn-коде и при этом максимально был избавлен от необходимости учитывать особенности языка LaTeX в работе.

Основные принципы разработки исходных текстов баз знаний с использованием scn-tex:

- весь исходный текст стандарта формируется исключительно с использованием набора команд scn-tex;
- запрещается использовать любые другие команды для форматирования текста, изменения шрифта, вставки внешних файлов и т.д;
- в рамках естественно-языковых фрагментов, входящих в состав стандарта, допускается использование команд LaTeX для вставки специальных символов и математических формул
- для добавления файлов изображений в текст стандарта используются только команды scn-tex;
- используются для формирования нумерованных и маркированных списков, добавления закрывающих и открывающих скобок различного вида (кроме круглых) используются только команды scn-tex;
- для выделения курсивом идентификаторов в рамках естественно-языковых фрагментов, входящих в состав стандарта, используется только команда \textit{}
- для выделения полужирным курсивом используется комбинация команд \textbf{\textit{}}

Каждая команда из набора scn-tex начинается с префикса \scn, после которого идет имя команды, примерно описывающее то, как связан текущий отображаемый фрагмент текста с описываемой сущностью.

Например:

- \scnrelfrom — дуга ориентированного отношения, которая выходит из описываемой сущности в другую сущность
- \scnrelto — дуга ориентированного отношения, которая входит в описываемую сущность
- \scnnote — естественно-языковое примечание к описываемой сущности

Полный перечень команд можно увидеть в файле scn.sty (см. *SoftwIoSLP-el*), а примеры использования команд каждого типа — в исходных текстах стандарта (см. *OSTISSR-el*).

Для формирования отступов для корректного отображения sc.n-текстов используется окружение \begin{scnindent} — \end{scnindent}. После смещения на определенное число уровней вправо следует смещение на то же число уровней влево.

Пример исходного текста:

```
\scnheader{кибернетическая система}
\scnsuperset{компьютерная система}
\begin{scnindent}
    \scnidtf{искусственная кибернетическая система}
    \scnsuperset{ostis-система}
    \begin{scnindent}
        \scnidtf{компьютерная система, построенная по технологии OSTIS на основе
 интерпретации спроектированной логико-семантическая sc-модель этой системы}
    \end{scnindent}
\end{scnindent}
```

Результат компиляции:

кибернетическая система
⊃ *компьютерная система*
  ≔ [искусственная кибернетическая система]
  ⊃ *ostis-система*
    ≔ [компьютерная система, построенная по технологии OSTIS на основе интерпретации спроектированной логико-семантическая sc-модель этой системы]

Команды из набора scn-tex делятся на классы. К чаще всего используемым классам относятся окружения (environments) и списки (lists), которые являются частным случаем окружения. Окружения могут быть вложенными.

Пример исходного текста:

```
\scnheader{Логико-семантическая модель Метасистемы IMS.ostis}
\scnrelfrom{примечание}{
\begin{scnset}
\scnfilelong{IMS.ostis}
\scnrelto{сокращение}{\scnfilelong{Метасистема IMS.ostis}}
\begin{scnindent}
\scnrelto{сокращение}{\scnfilelong{Intelligent MetaSystem of Open Semantic Technology
 for Intelligent Systems}}
\end{scnindent}
\end{scnset}
}
\scnidtf{Логико-семантическая модель интеллектуального ostis-портала научно-технических
 знаний по Технологии OSTIS}
```

Результат компиляции:

Логико-семантическая модель Метасистемы IMS.ostis
⇒ *примечание**:
  {
  [IMS.ostis]
  ⇐ *сокращение**:
    [Метасистема IMS.ostis]
    ⇐ *сокращение**:
      [Intelligent MetaSystem of Open Semantic Technology for Intelligent Systems]
  }
≔ [Логико-семантическая модель интеллектуального ostis-портала научно-технических знаний по Технологии OSTIS]

Таким образом scn-tex позволяет записать практически любую синтаксически корректную конструкцию SCn-кода.

Для трансляции исходного текста в базу знаний был разработан tex2scs-translator (см. *SoftwIoStST-el*), который переводит фрагменты scn-tex в SCs-код. Каждой scn-tex команде соответствует определенная синтаксическая конструкция SCs-кода. Таким образом, весь исходный текст Стандарта OSTIS, записанный в scn-tex, может быть

протранслирован в базу знаний любой ostis-системы, которая поддерживает сборку базы знаний из sc.s-файлов. Далее рассмотрим пример работы транслятора.

Текст для трансляции в формате scn-tex:

```
\scnheader{множество}
\begin{scnrelfromset}{разбиение}
\scnitem{конечное множество}
\scnitem{бесконечное множество}
\end{scnrelfromset}
```

Результат трансляции в SCs-код:

```
.system_element_0
=> nrel_subdividing: {
.system_element_1;
.system_element_2
};;

.system_element_1 => nrel_main_idtf: [конечное множество] (* <- lang_ru;;
 => nrel_format: format_html;; *);;
.system_element_1 <- sc_node;;

nrel_subdividing => nrel_main_idtf: [разбиение*] (* <- lang_ru;; =>
nrel_format: format_html;; *);;
nrel_subdividing <- sc_node_norole_relation;;

.system_element_2 => nrel_main_idtf: [бесконечное множество] (* <- lang_ru;;
 => nrel_format: format_html;; *);;
.system_element_2 <- sc_node;;

.system_element_0 => nrel_main_idtf: [множество] (* <- lang_ru;; => nrel_format:
 format_html;; *);;
.system_element_0 <- sc_node;;
```

Заключение к Главе 2.3.

В главе были рассмотрены правила идентификации *sc-элементов*, а также *SCg-код*, *SCs-код*, *SCn-код* — универсальные *внешние языки ostis-систем*, близкие *языку внутреннего смыслового представления знаний*. ***SCg-код*** представляет собой способ визуализации *sc-текстов* (*информационных конструкций SC-кода*) в виде рисунков этих абстрактных конструкций. ***SCs-код*** представляет собой множество линейных текстов (*sc.s-текстов*), каждый из которых состоит из предложений (*sc.s-предложений*), разделенных друг от друга двойной *точкой с запятой* (разделителем *sc.s-предложений*). ***SCn-код*** является языком структурированного внешнего представления текстов *SC-кода* и представляет собой синтаксическое расширение *SCs-кода*, направленное на повышение наглядности и компактности текстов *SCs-кода*. Для каждого из указанных *внешних языков* были определены *синтаксис* и *денотационная семантика*, приведены примеры описанных с помощью этих *языков информационных конструкций*.

Глава 2.4.

Представление формальных онтологий базовых классов сущностей в ostis-системах

⇒ *автор**:
- *Бутрин С.В.*
- *Шункевич Д.В.*

⇒ *аннотация**:
[В данной главе рассматриваются актуальные проблемы текущего состояния технологий разработки формальных онтологий сущностей. Предложен список онтологий базовых классов сущностей в рамках *Технологии OSTIS*.]

⇒ *подраздел**:
- *§ 2.4.1. Формальная онтология множеств*
- *§ 2.4.2. Формальная онтология связок и отношений*
- *§ 2.4.3. Формальная онтология параметров, величин и шкал*
- *§ 2.4.4. Формальная онтология чисел и числовых структур*
- *§ 2.4.5. Формальная онтология темпоральных сущностей*
- *§ 2.4.6. Формальная онтология ситуаций и событий, описывающих динамику баз знаний ostis-систем*

⇒ *ключевой знак**:
- *Ядро базы знаний ostis-системы*

⇒ *ключевое понятие**:
- *онтология*
- *предметная область*
- *знание*
- *база знаний*
- *базовый класс описываемых сущностей*
- *онтология верхнего уровня*

⇒ *библиографическая ссылка**:
- *Davydenko I.T.OntolBDoIS-2017art*
- *Гаврилова Т.А..БазыЗИСУ-2000кн*
- *Добров Б.В..ОнтолТМИП-2009кн*
- *Iqbal R..Analy oEM-2013art*
- *Горшков С.В.Введе вОМ-2016кн*
- *Hepp M.OntolSotABPaGC-2008art*
- *Golenkov V.V.OntolBDoIS-2017art*

Введение в Главу 2.4.

Для обеспечения совместного использования различных видов *знаний*, входящих в состав *базы знаний*, необходимо обеспечить их совместимость с указанной *базой знаний*, которая включает *семантическую совместимость*, что подразумевает однозначную и единую для всех *фрагментов базы знаний* трактовку используемых *понятий*.

Среди многообразия средств представления *знаний* к наиболее эффективным относятся *онтологии* (см. *Davydenko I.T.OntolBDoIS-2017art*, *Гаврилова Т.А..БазыЗИСУ-2000кн*, *Добров Б.В..ОнтолТМИП-2009кн*, *Iqbal R..Analy oEM-2013art*). Суть такого подхода при проектировании *базы знаний* состоит в рассмотрении *базы знаний* как иерархической системы выделенных *предметных областей* и соответствующих им *онтологий*(см. *Горшков С.В.Введе вОМ-2016кн* , *Hepp M.OntolSotABPaGC-2008art*). Однако онтологически можно по разному специфицировать *знания*. Чтобы решить эту проблему проектируются *онтологии верхнего уровня*.

Применение современных *онтологий верхнего уровня* при разработке *баз знаний интеллектуальных компьютерных систем* сопряжено с проблемами обеспечения их совместимости (см. *Golenkov V.V.OntolBDoIS-2017art*). Поскольку изначальной целью создания *онтологий верхнего уровня* являлось обеспечение совместимости *онтологий предметных областей* и *прикладных онтологий*, а не самих *интеллектуальных систем*.

Такими проблемами являются:

- свобода трактовки *понятий*, вызванная отсутствием их четкого *определения*;
- отсутствие единой *технологии проектирования баз знаний* на основе *онтологий верхнего уровня*;
- отсутствие принадлежности *онтологий верхнего уровня* к какой-либо *технологии*, что не позволяет использовать их в качестве *многократно используемых компонентов*;

Поэтому возникает необходимость в разработке такой *системы онтологии верхнего уровня*, которая смогла бы обеспечить *семантическую совместимость* между большим количеством *онтологий* различных предметных областей.

Предлагаемый подход подразумевает разработку *семейств Предметных областей и онтологий*, которые бы содержали описания всех необходимых *базовых классов сущностей* для построения *базы знаний интеллектуальной компьютерной системы*.

К таким *Предметным областям и онтологиям* относятся:
- *Предметная область и онтология множеств*;
- *Предметная область и онтология связок и отношений*;
- *Предметная область и онтология параметров, величин и шкал*;
- *Предметная область и онтология чисел и числовых структур*;
- *Предметная область и онтология структур*;
- *Предметная область и онтология темпоральных сущностей*;
- *Предметная область и онтология темпоральных сущностей баз знаний ostis-систем*;
- *Предметная область и онтология семантических окрестностей*;
- *Предметная область и онтология предметных областей*;
- *Предметная область и онтология онтологий*;
- *Предметная область и онтология логических формул, высказываний и формальных теорий*;
- *Предметная область и онтология внешних информационных конструкций и файлов ostis-систем*;
- *Глобальная предметная область действий и задач и соответствующая ей онтология методов и технологий*.

Данные *предметные области* являются частью *Ядра базы знаний*, которое должно быть в каждой *интеллектуальной системе*. Это ядро гарантирует *совместимость интеллектуальных компьютерных* систем за счет общего понятийного аппарата. В зависимости от специфики конкретных систем могут выделяться различные *Ядра базы знаний*, но неизменным должна оставаться наличие базовая части, включающей в себя *предметные области* и *онтологии* указанные выше.

§ 2.4.1. Формальная онтология множеств

⇒ *ключевое понятие**:
- *множество*
- *неориентированное множество*
- *ориентированное множество*
- *мультимножество*
- *класс*
- *семейство множеств*
- *мощность множества*

⇒ *ключевое отношение**:
- *включение**
- *принадлежность**
- *пересечение**
- *объединение**
- *декартово произведение**
- *разбиение**

Под ***множеством*** понимается соединение в некое целое M определенных хорошо различимых предметов m нашего созерцания или нашего мышления (которые будут называться «элементами» множества M).

множество — мысленная сущность, которая связывает одну или несколько сущностей в целое.

Более формально *множество* — это абстрактный математический объект, состоящий из элементов. Связь множеств с их элементами задается бинарным ориентированным отношением ***принадлежность****.

множество может быть полностью задано следующими тремя способами:
- путем перечисления (явного указания) всех его элементов (очевидно, что таким способом можно задать только конечное множество);

- с помощью определяющего высказывания, содержащего описание общего характеристического свойства, которым обладают все те и только те объекты, которые являются элементами (то есть принадлежат) задаваемого множества;
- с помощью теоретико-множественных операций, позволяющих однозначно задавать новые множества на основе уже заданных (это операции объединения, пересечения, разности множеств и другие).

Для любого семантически ненормализованного *множества* существует единственное семантически нормализованное *множество*, в котором все элементы, не являющиеся знаками множеств, заменены на знаки множеств.

множество
⇒ *разбиение**:
{• *конечное множество*
• *бесконечное множество*
}
⇒ *разбиение**:
{• *множество без кратных элементов*
• *мультимножество*
}
⇒ *разбиение**:
{• *связка*
• *класс*
:= [sc-элемент, обозначающий класс sc-элементов]
:= [sc-знак множества sc-элементов, эквивалентных в том или ином смысле]
• *структура*
:= [sc-знак множества sc-элементов, в состав которого входят sc-связки или структуры, связывающие эти sc-элементы]
}
⇒ *разбиение**:
{• *четкое множество*
• *нечеткое множество*
}
⇒ *разбиение**:
{• *множество первичных сущностей*
• *множество множеств*
• *множество первичных сущностей и множеств*
}
⇒ *разбиение**:
{• *рефлексивное множество*
• *нерефлексивное множество*
}
⇒ *разбиение**:
{• *сформированное множество*
• *несформированное множество*
}
⇒ *разбиение**:
{• *кортеж*
• *неориентированное множество*
}

множество без кратных элементов
:= [классическое множество]
:= [канторовское множество]
:= [множество, состоящее из разных элементов]
:= [множество без кратного вхождения элементов]
:= [множество, все элементы которого входят в него однократно]
:= [множество, не имеющее кратного вхождения элементов]
⇒ *пояснение**:
[***множество без кратных элементов*** - это *множество*, для каждого элемента которого существует только одна пара принадлежности, выходящая из знака этого множества в указанный элемент.]

нечеткое множество
⇒ *пояснение**:

[***нечеткое множество*** — это *множество*, которое представляет собой совокупность элементов произвольной природы, относительно которых нельзя точно утверждать — обладают ли эти элементы некоторым характеристическим свойством, которое используется для задания этого нечеткого множества. Принадлежность элементов такому множеству указывается при помощи *нечетких позитивных sc-дуг принадлежности*.]

четкое множество
⇒ *пояснение**:
[***четкое множество*** — это *множество*, принадлежность элементов которому достоверна и указывается при помощи *четких позитивных sc-дуг принадлежности*.]

мультимножество
:= [множество, имеющее кратные вхождения хотя бы одного элемента]
:= [множество, по крайней мере один элемент которого входит в его состав многократно]
⇒ *пояснение**:
[***мультимножество*** - это *множество*, для которого существует хотя бы одна кратная пара принадлежности, выходящая из знака этого множества.]

множество первичных сущностей
⊃ *класс первичных сущностей*
⊂ *множество*
⇒ *пояснение**:
[***множество первичных сущностей*** — это *множество*, элементы которого не являются знаками множеств.]

семейство множеств
:= [множество множеств]
⊃ *класс классов*
⇒ *пояснение**:
[***семейство множеств*** — это *множество*, элементами которого являются знаки множеств.]

нерефлексивное множество
⇒ *пояснение**:
[***нерефлексивное множеств*** — это *множество*, знак которого не является элементом этого множества]

рефлексивное множество
⇒ *пояснение**:
[***рефлексивное множеств*** — это *множество*, знак которого является элементом этого множества]

принадлежность*
:= [принадлежность элемента множеству*]
:= [отношение принадлежности элемента множеству*]
∈ *бинарное отношение*
∈ *ориентированное отношение*

принадлежность* — это *бинарное ориентированное отношение*, каждая связка которого связывает множество с одним из его элементов. Элементами отношения *принадлежность** по умолчанию являются *позитивные sc-дуги принадлежности*.

класс
:= [класс sc-элементов]
⇒ *разбиение**:
{• *класс первичных sc-элементов*
• *класс множеств*
}

класс — множество элементов, обладающих какими-либо явно указываемыми общими свойствами.

включение*
:= *часто используемый sc-идентификатор**:
[включает*]
:= *часто используемый sc-идентификатор**:
[включает в себя*]

:= [включение множеств*]
:= [быть подмножеством*]
∈ *бинарное отношение*
∈ *ориентированное отношение*
∈ *транзитивное отношение*
⇒ *область определения**:
множество
⊃ *строгое включение**

включение* — это бинарное ориентированное отношение, каждая связка которого связывает два множества. Будем говорить, что *Множество Si включает** в себя *Множество Sj* в том и только том случае, если каждый элемент *Множества Sj* является также и элементом *Множества Si*

объединение*
:= [объединение множеств*]
∈ *квазибинарное отношение*
∈ *ориентированное отношение*

объединение* — это *квазибинарное ориентированное отношение*, областью определения которого является семейство всевозможных множеств. Будем говорить, что *Множество Si* является объединением *Множество Sj* и *Множество Sk* тогда и только тогда, когда любой элемент *Множество Si* является элементом или *Множество Sj* или *Множество Sk*.

разбиение*
:= [разбиение множества*]
:= [объединение попарно непересекающихся множеств*]
:= [декомпозиция множества*]
∈ *квазибинарное отношение*
∈ *ориентированное отношение*
∈ *отношение декомпозиции*

разбиение* — это *квазибинарное ориентированное отношение*, областью определения которого является семейство всевозможных множеств. В результате разбиения множества получается множество попарно непересекающихся множеств, объединение которых есть исходное множество.

Семейство множеств {*S1. . . , Sn*} является разбиением множества *Si* в том и только том случае, если:
- семейство {*S1. . . , Sn*} является семейством *попарно непересекающихся множеств*;
- семейство {*S1. . . , Sn*} является покрытием множества *Si* (или другими словами, множество *Si* является *объединением* множеств, входящих в указанное выше семейство).

пересечение*
:= [пересечение множеств*]
∈ *квазибинарное отношение*
∈ *ориентированное отношение*

пересечение* — это операция над множествами, аргументами которой являются два или большее число множеств, а результатом является множество, элементами которого являются все те и только те сущности, которые одновременно принадлежат каждому множеству, которое входит в семейство аргументов этой операции.

Будем говорить, что *Множество Si* является пересечением *Множество Sj* и *Множество Sk* тогда и только тогда, когда любой элемент *Множество Si* является элементом *Множество Sj* и элементом *Множество Sk*.

пара пересекающихся множеств*
∈ *бинарное отношение*
∈ *неориентированное отношение*
⇒ *пояснение**:
[***пара пересекающихся множеств**** — *бинарное неориентированное отношение* между двумя *множествами*, имеющими непустое *пересечение**.]
⇒ *определение**:
[*пара пересекающихся множеств** — *бинарное неориентированное отношение* между двумя *множествами*, имеющими, по крайней мере, один общий для этих двух множеств элемент.]

попарно пересекающиеся множества*

:= [семейство попарно пересекающихся множеств*]
⊃ *пересекающиеся множества**
∈ *отношение*
⇒ *определение**:
[***попарно пересекающиеся множества**** — семейство множеств, каждая пара которых является парой пересекающихся множеств, то есть каждая пара которых имеет хотя бы один общий элемент]
⇒ *примечание**:
[Не каждое *семейство попарно пересекающихся множеств** является *семейством пересекающихся множеств**, хотя обратное верно.]

декартово произведение*
:= [декартово произведение множеств*]
:= [прямое произведение множеств*]
∈ *бинарное отношение*
∈ *ориентированное отношение*

декартово произведение* — это *бинарное ориентированное отношение* между *ориентированной парой* множеств и *множеством*, элементами которого являются всевозможные упорядоченные пары, первыми элементами которых являются элементы первого из указанных множеств, вторыми — элементы второго из указанных множеств.

мощность множества
:= [кардинальное число]
:= [общее число вхождений элементов в заданное множество]
:= [класс эквивалентности, элементами которого являются знаки всех тех и только тех множеств, которые имеют одинаковую мощность]
:= [класс эквивалентности, соответствующий отношению быть парой множеств, имеющих одинаковую мощность (равномощность множеств)]
:= [величина мощности множеств]
:= [трансфинитное число]
:= [мощность по Кантору]
∈ *параметр*

мощность множества — это *параметр*, элементами которых являются *множества*, имеющие одинаковое количество элементов. Значением данного параметра является числовая величина, задающая количество элементов, входящих в данный класс множеств, то есть по сути, количество *позитивных sc-дуг принадлежности*, выходящих из данного *множества*.

Для двух множеств, имеющих одинаковую мощность, существует взаимно-однозначное соответствие между ними (между множествами вхождений элементов в эти множества — на случай мультимножеств).

§ 2.4.2. Формальная онтология связок и отношений

⇒ *ключевое понятие**:
- *связь*
- *небинарная связь*
- *отношение*
- *бинарное отношение*
- *небинарное отношение*
- *ролевое отношение*
- *неролевое отношение*
- *квазибинарное отношение*
- *отношение порядка*
- *арность*

⇒ *ключевое отношение**:
- *домен**
- *атрибут отношения**
- *область определения'*
- *область прибытия'*
- *соответствие**

связь — множество, являющееся абстрактной моделью связи между описываемыми сущностями, которые или знаки которых являются элементами этого *множества*.

Напомним, что все элементы *множества*, представленного в *SC-коде*, являются знаками, но описываемыми сущностями могут быть не только сущности, обозначаемые *sc-элементами*, но и сами эти *sc-элементы*.

связь
:= [связка sc-элементов]
:= [sc-связка]
⇒ *разбиение**:
{• *бинарная связь*
• *небинарная связь*
}
⇒ *разбиение**:
{• *неориентированная связь*
• *ориентированная связь*
}

бинарная связь
⇒ *разбиение**:
{• *sc-коннектор*
• *неатомарная бинарная связь*
}

Данное разбиение осуществляется на основе синтаксического признака, а не семантического, поскольку каждый *sc-коннектор* может быть записан в памяти при помощи семантически эквивалентной конструкции, содержащей знак самой связи и пары принадлежности, ведущие к ее элементам, уточненные, при необходимости ролевыми отношениями.

Каждый ***sc-коннектор*** представлен в *sc-памяти* одним *sc-элементом* и семантически эквивалентен конструкции, содержащей знак некоторой *бинарной связи* и пары принадлежности, ведущие к элементам этой связи, уточненные, при необходимости ролевыми отношениями.

Такая конструкция может быть обозначена ***sc-коннектором*** только в случае, когда роли компонентов соответствующей бинарной связи указываются только при помощи *числовых атрибутов 1′* и *2′* или не уточняются вообще.

небинарная связь — *связь*, имеющая больше двух элементов.

неориентированная связь
⊃ *неориентированное множество*
⇒ *пояснение**:
[***неориентированная связь*** — связь, все элементы которой имеют одинаковые роли (при этом соответствующее ролевое отношение, как правило, явно не указывается).]

ориентированная связь
⊃ *кортеж*
⇒ *пояснение**:
[***ориентированная связь*** — связь, в которой с помощью ролевых отношений, указываются роли компонентов этой связи.]

отношение
:= [класс связей]
:= [класс sc-связок]
:= [множество отношений]
:= [Множество всевозможных отношений]
⇒ *определение**:
[***отношение***, *заданное на множестве M* — это подмножество *декартового произведения* этого множества самого на себя некоторое количество раз]

В более широком смысле ***отношение*** — это математическая структура, которая формально определяет свойства различных объектов и их взаимосвязи.
⇒ *разбиение**:
{• *класс равномощных связок*

- *класс связок разной мощности*

}

⇒ *разбиение**:

{• *бинарное отношение*
• *небинарное отношение*

}

⇒ *разбиение**:

{• *ориентированное отношение*
• *неориентированное отношение*

}

⇒ *разбиение**:

{• *ролевое отношение*
• *неролевое отношение*

}

бинарное отношение

:= [отношение арности два]

:= [двухместное отношение]

⊃ *квазибинарное отношение*

⊃ *отношение порядка*

⊃ *отношение толерантности*

⇒ *разбиение**:

{• *рефлексивное отношение*
• *антирефлексивное отношение*
• *частично рефлексивное отношение*

}

⇒ *разбиение**:

{• *симметричное отношение*
• *антисимметричное отношение*
• *частично симметричное отношение*

}

⇒ *разбиение**:

{• *транзитивное отношение*
• *антитранзитивное отношение*
• *частично транзитивное отношение*

}

⇒ *разбиение**:

{• *ролевое отношение*
• *неролевое бинарное отношение*

}

⇒ *определение**:

[***бинарное отношение*** — это множество таких отношений на множестве M, являющихся подмножеством *декартова произведения* множества M.]

Если *бинарное отношение* R задано на *множестве* M и два элемента этого множества a и b связаны данным отношением, то будем обозначать такую связь как aRb.

квазибинарное отношение

⇒ *пояснение**:

[***квазибинарное отношение*** — *множество ориентированных пар*, первые компоненты которых являются связками.]

Таким образом, *sc-дуги*, принадлежащие *квазибинарным отношениям*, всегда выходят из связок.

⇒ *sc-утверждение**:

[В область определения квазибинарного отношения будем включать:

- вторые компоненты ориентированных пар, принадлежащих этому отношению;
- элементы первых компонентов ориентированных пар, принадлежащих этому отношению;
- других элементов область определения квазибинарного отношения не содержит.

]

связанное отношение*

∈ *бинарное отношение*

⇒ *определение**:

[*связанное отношение* R* на *множестве A* — это *бинарное отношение*, в котором для каждой пары элементов *a* и *b* этого множества выполняется одно из двух отношений: *aRb* или *bRa*.**]**

отношение порядка

⇒ *разбиение**:

{• *отношение строгого порядка*

• *отношение нестрогого порядка*

}

⇒ *определение**:

[*отношение порядка* — это *бинарное отношение*, обладающее свойством транзитивности и антисимметричности.**]**

отношение строгого порядка

⇒ *определение**:

[*отношение строгого порядка* — это *отношение порядка*, обладающее свойством антирефлексивности.**]**

отношение нестрогого порядка

⇒ *определение**:

[*отношение нестрогого порядка* — это *отношение порядка*, обладающее свойством рефлексивности.**]**

отношение толерантности

⇒ *определение**:

[*отношение толерантности* — это *бинарное отношение*, принадлежащее классам *симметричное отношение* и *рефлексивное отношение*.**]**

отношение эквивалентности

:= **[**максимальное семейство отношений эквивалентности**]**

⊂ *отношение толерантности*

⇒ *определение**:

[*отношение эквивалентности* — это *отношение толерантности*, принадлежащее классу *транзитивных отношений***]**

⇒ *примечание**:

[Каждое отношение эквивалентности уточняет то, что мы считаем эквивалентными сущностями, то есть то, на какие сходства этих сущностей мы обращаем внимание и какие их отличия мы игнорируем (не учитываем).**]**

ролевое отношение

:= **[**атрибут**]**

:= **[**атрибутивное отношение**]**

:= **[**отношение, которое задает роль элементов в рамках некоторого множества**]**

:= **[**отношение, являющееся подмножеством отношения принадлежности**]**

⇐ *семейство подмножеств**:

*принадлежность**

⊂ *бинарное отношение*

⊃ *числовой атрибут*

⇒ *пояснение**:

[*ролевое отношение* — это отношение, являющееся подмножеством отношения принадлежности.**]**

⇒ *правило идентификации экземпляров**:

[В конце каждого *идентификатора*, соответствующего экземплярам класса ***ролевое отношение***, не являющегося системным, ставится знак “′”.

Например:
ключевой экземпляр′

Из-за ограничений в разрешенном алфавите символов, в системном идентификаторе не может быть использовать знак “′”, поэтому в начале каждого *системного идентификатора*, соответствующего экземплярам класса ***ролевое отношение*** ставится префикс “rrel_”.

Например:
*rrel_key_sc_element***]**

числовой атрибут
:= [порядковый номер]
:= [номер компонента ориентированной связки]
∋ *1'*
∋ *2'*
∋ *3'*
⇒ *пояснение**:
[***числовой атрибут*** *— ролевое отношение*, задающее порядковый номер элемента некоторой ориентированной связки, не уточняя при этом семантику такой принадлежности. Во многих случаях бывает достаточно использовать числовые атрибуты, чтобы различать компоненты связки, семантика каждого из которых дополнительно оговаривается, например, при определении отношения, которому данная связка принадлежит.]

неролевое отношение
⇒ *разбиение**:
{• *небинарное отношение*
• *неролевое бинарное отношение*
}
⇒ *пояснение**:
[***неролевое отношение*** — отношение, не являющееся подмножеством отношения принадлежности.]
⇒ *правило идентификации экземпляров**:
[В конце каждого *идентификатора*, соответствующего экземплярам класса ***неролевое отношение***, не являющегося системным, ставится знак “*”.

Например:
*включение**

Из-за ограничений в разрешенном алфавите символов, в системном идентификаторе не может быть использовать знак “*”, поэтому в начале каждого *системного идентификатора*, соответствующего экземплярам класса ***неролевое отношение*** ставится префикс “nrel_”.

Например:
nrel_inclusion]

арность
:= [арность отношения]
∈ *параметр*
⇒ *пояснение**:
[***арность*** — это параметр, каждый элемент которого представляет собой класс *отношений*, каждая связка которых имеет одинаковую *мощность*. Значение данного *параметра* совпадает со значением *мощности* каждой из таких связок.]

область определения*
:= [область определения отношения*]
∈ *бинарное отношение*
⇒ *пояснение**:
[***область определения**** — это *бинарное отношение*, связывающее отношение со множеством, являющимся его областью определения.

Областью определения отношения будем называть результат теоретико-множественного объединения всех связок этого отношения, или, другими словами, результат теоретико-множественного объединения всех множеств, являющихся доменами данного отношения.]

атрибут отношения*
:= [ролевой атрибут, используемый в связках заданного отношения*]
∈ *бинарное отношение*
⇒ *пояснение**:
[***атрибут отношения**** — это *бинарное отношение*, связывающее заданное отношение с *ролевым отношением*, используемым в данном отношении для уточнения роли того или иного элемента связок данного отношения.]

домен*
:= [домен отношения по заданному атрибуту*]
∈ *бинарное отношение*
⇒ *пояснение**:

[*домен** — это *бинарное отношение*, связывающее связку отношения *атрибут отношения** со множеством, являющимся доменом заданного отношения по заданному атрибуту. Множество *di* является доменом отношения *ri* по атрибуту *ai* в том и только том случае, если элементами этого множества являются все те и только те элементы связок отношения *ri*, которые имеют в рамках этих связок атрибут *ai*.**]**

соответствие*
:= [наличие соответствия*]
∈ *бинарное отношение*
⇒ *разбиение**:
{• *соответствие между непересекающимися множествами**
• *соответствие между строго пересекающимися множествами**
• *соответствие, область отправления и область прибытия которого совпадают**
}
⇒ *разбиение**:
{• *всюду определенное соответствие**
• *частично определенное соответствие**
}
⇒ *разбиение**:
{• *сюръекция**
• *несюръективное соответствие**
}
⇒ *разбиение**:
{• *однозначное соответствие**
• *неоднозначное соответствие**
}
⇒ *определение**:
[*соответствие** — *бинарное ориентированное отношение*, каждая пара которого связывает два множества и указывает на наличие некоторого отношения, связывающего элементы этих двух множеств.**]**

область отправления′
:= [область отправления соответствия′]
:= [область определения соответствия′]
:= [первый компонент пары в отношении соответствия′]
∈ *ролевое отношение*
⇒ *определение**:
[*область отправления′* — *ролевое отношение*, указывающее на первый компонент пары в рамках отношения *соответствие.**]**

область прибытия′
:= [область прибытия соответствия′]
:= [область значений соответствия′]
∈ *ролевое отношение*
⇒ *определение**:
[*область прибытия′* — *ролевое отношение*, указывающее на второй компонент пары в рамках отношения *соответствие.**]**

§ 2.4.3. Формальная онтология параметров, величин и шкал

⇒ *ключевое понятие**:
• *параметр*
• *величина*
• *точная величина*
• *неточная величина*
• *интервальная величина*
• *измерение с фиксированной единицей измерения*
⇒ *ключевое отношение**:
• *единица измерения**
• *измерение**
• *точность**

Каждый ***параметр*** представляет собой класс, являющийся семейством всевозможных классов эквивалентности или толерантности, задаваемых либо *отношением эквивалентности*, либо *отношением толерантности* (симметричным, рефлексивным, но частично транзитивным).

Так, например, элементами (значениями, величинами) *параметра длина* являются либо классы эквивалентности, задаваемые отношением эквивалентности “иметь точно одинаковую длину*”, либо классы толерантности, задаваемые отношением вида “иметь приблизительно одинаковую длину с указываемой точностью*”, либо интервальные классы, задаваемые бинарными отношениями вида “иметь длину, находящуюся в одном и том же указываемом интервале*” (например, от 1 метра до 2 метров).

Примерами параметров как отношений эквивалентности являются:
- равновеликость геометрических фигур (по длине, площади, объему — в зависимости от размерности этих фигур);
- иметь одинаковый цвет (быть эквивалентными по цвету);
- эквивалентность, по вкусу, запаху, твердости и так далее.

Заметим, что среди элементов (значений, величин) параметра могут встречаться пересекающиеся множества (классы), но объединение всех элементов каждого параметра есть не что иное, как класс всевозможных сущностей, обладающих этим параметром (свойством, характеристикой). Например, класс всех сущностей, имеющих длину, класс всех сущностей, обладающих цветом.

параметр
:= [характеристика]
:= [свойство]
:= [признак]
:= [класс классов]
:= [измеряемое свойство]
:= [признак классификации или покрытия некоторого класса сущностей]
:= [признак разбиения или покрытия некоторого класса сущностей]
:= [семейство множеств, разбивающих или покрывающих некоторый класс сущностей]
:= [семейство классов сущностей, обладающих одинаковым соответствующим свойством]
:= [фактор-множество, соответствующее некоторому отношению эквивалентности, или аналог фактор-множества, соответствующий некоторому отношению толерантности]
⇒ *разбиение**:
{• *измеряемый параметр*
• *неизмеряемый параметр*
}

Каждый конкретный параметр (характеристика), то есть каждый элемент класса всевозможных параметров (характеристик) есть, по сути, признак классификации сущностей, обладающих это характеристикой, по принципу эквивалентности (одинаковости значения) этой характеристики. Например, параметр *цвет* разбивает множество сущностей имеющих цвет на классы, каждый из которых включает в себя сущности, имеющие одинаковый цвет. Параметр может разбиваться на классы для уточнения некоторого свойства, например элементами параметра цвет будут классы, соответствующие конкретным цветам (синий, красный и так далее), в свою очередь каждый конкретный цвет может включать более частные классы, уточняющие данное свойство, например, темно-синий, светло-красный и так далее.

Другими словами, каждому множеству сущностей может ставиться в соответствие набор (семейство) параметров (параметрическое пространство), которыми обладают сущности этого множества — аналог семейства отношений, определенных (заданных) на этом множестве. Часто бывает важно построить такое параметрическое пространство, "точки" которого взаимнооднозначно соответствуют параметризуемым сущностям (например, набор параметров, позволяющих однозначно идентифицировать, установить личность каждого человека).

Таким образом, для каждого используемого элемента (значения) какого-либо параметра, необходимо явно указывать спецификацию этого значения (точное значение, неточное значение, интервальное значение, точность, интервал).

область определения параметра*
:= [множество всех тех и только тех сущностей, которые являются компонентами значений заданного параметра*]
:= [множество всех тех и только тех сущностей, которые обладают заданным параметром*]
⇐ *включение**:
*объединение**

измеряемый параметр
:= [количественный параметр]
:= [семейство измеряемых величин]
:= [семейство классов эквивалентности, каждому из которых может быть поставлено в соответствие числовое значение]

Каждый ***измеряемый параметр*** представляет собой *параметр*, значение (элемент, экземпляр) которого трактуется как *величина*, которой можно поставить в соответствие ее числовое значение на основании выбранной единицы измерения и/или точки отсчета (нулевой отметки выбранной шкалы).
⊃ *параметр, измеряемый по шкале*

неизмеряемый параметр
:= [качественный параметр]

ориентированный параметр
:= [упорядоченный параметр]
⊃ *параметр, измеряемый по шкале*
:= [параметр, на значениях которого может быть задано некоторое отношение порядка, семантика которого уточняется в зависимости от семантики параметра]

величина
:= [значение количественного параметра]
:= [значение измеряемого параметра]
:= [класс сущностей, имеющих одинаковое значение соответствующего параметра]
⇒ *включение**:
- *точная величина*
- *неточная величина*
- *интервальная величина*

Каждая ***величина*** представляет собой однозначный и независящий от шкалы измерения результат измерения некоторой характеристики у некоторой сущности.

Каждой *величине* можно поставить в соответствие ее числовое значение на основании выбранной единицы измерения и точки отсчета (нулевой отметки выбранной шкалы, в случае, если измерение осуществляется по шкале).

Нельзя путать значение параметра (*величину*) и значение величины по некоторой шкале, которое может быть скалярным и векторным.

точная величина
:= [точное значение параметра]
:= [множество всех точных значений параметра]
:= [значение параметра, являющееся семейством классов эквивалентности, соответствующим некоторому отношению эквивалентности]
:= [класс эквивалентности]

Каждая ***точная величина*** имеет одно фиксированное значение в некоторой единице измерения или по какой-либо шкале. При этом считается, что все элементы такого класса имеют одинаковое значение данного параметра и отклонениями можно пренебречь.

Каждой *точной величине* можно поставить в соответствие группу *неточных величин*, являющихся не разбиениями, а покрытиями того же множества, но с разной степенью точности.

неточная величина
:= [множество неточных значений параметра]
:= [приблизительная величина]
:= [приблизительное значение параметра]
:= [значение параметра в интервале с нефиксированными границами]

Каждой ***неточной величине*** ставится в соответствие ее значение в некоторой единице измерения или по какой-либо шкале, а также дополнительно указывается *точность**, то есть возможное отклонение от данного значения.

интервальная величина
:= [интервальное значение параметра]
:= [значение параметра в интервале с фиксированными границами]

:= [интервал значения параметра из множества пересекающихся интервалов разной длины, имеющих нефиксированные границы]

Каждая ***интервальная величина*** представляет собой класс сущностей, находящихся в рамках точно заданного интервала, минимальная и максимальная точка которого являются *точными величинами*. Результатом *измерения** такой величины является ориентированная пара, первым компонентом которой является левая (меньшая) граница интервала, вторым компонентом — правая (большая) граница интервала.

эталон'
:= [образец']
∈ *ролевое отношение*

Ролевое отношение *эталон'* указывает на тот элемент значения некоторого параметра, который в рамках данного класса эквивалентности считается эталонным, то есть он используется как образец при определении данного параметра.

эталон' может задаваться как для измеряемых, так и для неизмеряемых параметров, например, эталон метра или эталон красоты.

измерение*
:= [значение параметра*]
:= [значение заданной величины заданного параметра*]
:= [измерение как соответствие*]
:= [результат измерения заданной величины в заданной единице измерения и по заданной шкале*]
:= [бинарное ориентированное отношение, связывающее различные величины с результатами их измерения в различных единицах измерения и по различным шкалам*]

Связки отношения *измерение** связывают величину и ее значение в некоторой единице измерения (в том числе, в интервале) или по некоторой шкале. Конкретная единица измерения или шкала указывается дополнительно при помощи соответствующего отношения. Одной величине может соответствовать только одно значение в каждой возможной единице измерения или одна точка на некоторой шкале.

единица измерения*
∈ *бинарное отношение*
:= [единица по шкале*]
:= [единичная отметка по шкале*]

Связки отношения ***единица измерения**** связывают знак конкретного ***измерения с фиксированной единицей измерения*** и некоторую *точную величину*, входящую в тот же конкретный *параметр*, что и первый компонент связок этого конкретного измерения, и которая используется в данном случае в качестве единицы измерения.

измерение по шкале
:= [шкала]
⇐ *семейство подмножеств**:
*измерение**

Каждая ***измерение по шкале*** представляет собой подмножество отношения *измерение** и характеризуется не единицей измерения, а некоторой точкой отсчета для данной ***шкалы***. Результатом *измерения по шкале* будет некоторая точка шкалы, отстоящая от точки отсчета на определенное расстояние в нужную сторону (меньшую или большую). Понятно, что это расстояние может быть измерено любыми единицами измерения, но его величина при этом останется неизменной.

Не стоит путать измерение по *измерение по шкале*, которое зависит от *нулевой отметки**, с измерением изменения того же *параметра*, которое характеризуется единицей измерения и не зависит от точки отсчета. Например, не стоит путать дату по некоторому календарю, соответствующую *началу* какого-либо процесса, и *длительность* этого процесса, которая не зависит от выбранного календаря.

Каждое ***арифметическое выражение на величинах*** представляет собой *связку*, компонентами которой являются элементы или подмножества некоторого *количественного параметра*.

действие. измерение
:= [измерение как действие]
:= [действие, направленное на установление связи, принадлежащей отношению измерение* и связывающей величину, которая принадлежит заданному параметру, и которой принадлежит заданная сущность, и соответствующее значение этой величины на некоторой шкале]
:= [действие, направленное на решение задачи измерения заданного параметра у заданной сущности]
⇐ *включение**:
действие

задача. измерение
:= [спецификация действия измерения]
:= [спецификация действия, целью которого является измерение заданного параметра у заданной сущности]
⇐ *включение**:
задача

§ 2.4.4. Формальная онтология чисел и числовых структур

⇒ *ключевое понятие**:
- *число*
- *цифра*

⇒ *ключевое отношение**:
- *модуль**
- *арифметическая операция**
- *произведение**

число — это основное понятие математики, используемое для количественной характеристики, сравнения, нумерации объектов и их частей. Письменными знаками для обозначения чисел служат *цифры*.

цифра
:= [множество цифр]
⊂ *внутренний файл ostis-системы*
⇒ *включение**:
- *арабская цифра*
- *римская цифра*

цифра — это множество файлов, обозначающих вхождение этой цифры во всевозможные записи чисел с помощью этой цифры.

натуральное число
:= [множество натуральных чисел]
⊂ *целое число*

натуральное число — это подмножество множества *целых чисел*, которые используются при счете предметов.

рациональное число
:= [множество рациональных чисел]
⊂ *действительное число*
⊂ *целое число*

рациональное число — это число, представляемое *обыкновенной дробью*, где числитель — *целое число*, а знаменатель — *натуральное число*.

дробь
:= [множество дробей]
⇒ *включение**:
- *обыкновенная дробь*
- *десятичная дробь*

дробь — это число, состоящее из одной или нескольких равных частей (долей) единицы

обыкновення дробь
:= [множество обыкновенных дробей]
:= [множество простых дробей]

обыкновенная дробь - запись *рационального числа* в виде $\pm\frac{m}{n}$ или $\pm m/n$, где $n \neq 0$.Горизонтальная или косая черта обозначает знак деления, в результате которого получается частное. Делимое называется числителем дроби, а делитель — знаменателем.

десятичная дробь
:= [множество десятичных дробей]

десятичная дробь — разновидность дроби, которая представляет собой способ представления действительных чисел в виде $\pm d_m \ldots d_1 d_0{,}d_{-1}d_{-2}\ldots$, где , — десятичная запятая, служащая разделителем между целой и дробной частью числа, d_km — десятичные цифры.

иррациональное число
:= [множество иррациональных чисел]
⊂ *действительное число*

иррациональное число — это *вещественное число*, которое не является рациональным, то есть не может быть представлено в виде дроби, где числитель — *целое число*, знаменатель — *натуральное число*. Любое *иррациональное число* может быть представлено в виде бесконечной непериодической десятичной дроби.

модуль*
:= [модуль числа*]
∈ *бинарное отношение*
⇒ *пояснение**:
[Связки отношения ***модуль**** связывают некоторое *число* (которое может быть как *отрицательным*, так и *положительным*) и другое *число* (всегда *положительное*), которое выражает расстояние от указанного числа до *Нуля* в единицах.]

арифметическая операция*
:= [множество арифметических операций]
⇐ *семейство подмножеств**:
арифметическое выражение

Каждая ***арифметическая операция**** представляет собой *отношение*, элементами которого являются *арифметические выражения*, то есть множество *арифметических выражений* какого-либо одного вида.

сумма*
:= [сложение*]
∈ *арифметическая операция*
∈ *квазибинарное отношение*

сумма* — это арифметическая операция, в результате которой по данным числам (слагаемым) находится новое число (сумма), обозначающее столько единиц, сколько их содержится во всех слагаемых.

Первым компонентом связки отношения *сумма** является *множество чисел* (слагаемых), содержащее два или более элемента, вторым компонентом — *число*, являющееся результатом сложения.

Отдельно отметим, что каждая связка отношения *сумма** вида a = b+c может также трактоваться и как запись о вычитании чисел, например b = a-c, в связи с чем *арифметическая операция* разности чисел отдельно не вводится.

произведение*
:= [умножение*]
∈ *арифметическая операция*
∈ *квазибинарное отношение*

произведение* — это *арифметическая операция*, в результате которой один аргумент складывается столько раз, сколько показывает другой, затем результат складывается столько раз, сколько показывает третий и так далее.

Первым компонентом связки отношения *произведение** является *множество чисел* (множителей), содержащее два или более элемента, вторым компонентом — *число*, являющееся результатом произведения.

Отдельно отметим, что каждая связка отношения *произведение** вида a = b*c может также трактоваться и как запись о делении чисел, например b = a/c, в связи с чем *арифметическая операция* деления чисел отдельно не вводится.

возведение в степень*
∈ *арифметическая операция*
∈ *бинарное отношение*

возведение в степень* — это *арифметическая операция*, в результате которой число, называемое основанием степени, умножается само на себя столько раз, каков показатель степени.

Первым компонентом связки отношения *возведение в степень** является ориентированная пара, первым компонентом которой является *число*, которое является основанием степени, вторым — *число*, которое является показателем степени; Вторым компонентом связки отношения *возведение в степень** является *число*, которое является результатом возведения в степень.

больше*
∈ *бинарное отношение*
∈ *отношение строгого порядка*
⇒ *пояснение**:
[*больше** — это *бинарное отношение* сравнения чисел. Из двух чисел на координатной прямой больше то, которое расположено правее. Соответственно, первым компонентом связки *отношения больше** является большее из двух *чисел*.**]**

§ 2.4.5. Формальная онтология темпоральных сущностей

⇒ *ключевое понятие**:
- *временная сущность*
- *временная связь*
- *процесс*
- *темпоральное отношение*

⇒ *ключевой параметр**:
- *начало*^
- *завершение*^
- *длительность*^

временная сущность
:= [временно существующая сущность]
:= [нестационарная сущность]
:= [сущность, имеющая и/или начало, и/или конец своего существования]
:= [sc-элемент, являющийся знаком некоторой временно существующей сущности]
:= [сущность, обладающая темпоральными характеристиками (длительностью, начальным моментом, конечным моментом и так далее)]
⇒ *разбиение**:
{
- *прошлая сущность*
- *настоящая сущность*
- *будущая сущность*

}
⇒ *разбиение**:
{
- *временная связь*
- *темпоральная структура*
  := [структура, содержащая хотя бы одну временную сущность]
  ⇒ *включение**:
  структура
  ⇒ *примечание**:
  [Следует отличать:
  - временный характер самой структуры как sc-элемента;
  - временный характер sc-элементов, принадлежащих данной структуре, и сущностей, обозначаемых этими sc-элементами;

- временный характер пар принадлежности, связывающих структуру с ее элементами.

]

:= [структура, описывающая темпоральные свойства (свойства, связанные со временем) окружающей среды, частью которой являются также и различные базы знаний кибернетических систем (в том числе и собственная база знаний).]

⇒ *разбиение**:

{• *ситуация*

:= [статическая темпоральная структура]

• *процесс*

:= [динамическая структура]

:= [динамическая темпоральная структура]

}

- *материальная сущность*

}

⇒ *разбиение**:

{• *непрерывная временная сущность*

⇒ *разбиение**:

{• *точечная временная сущность*

:= [атомарная временная сущность]

:= [условно мгновенная временная сущность]

:= [временная сущность, длительность существования которой в данном контексте считается несущественной (пренебрежительно малой)]

• *длительная непрерывная временная сущность*

}

- *дискретная временная сущность*

:= [временная сущность, которая может быть декомпозирована на последовательность точечных временных сущностей]

:= [временная сущность, которой соответствует некоторый временной ряд параметров (состояний) точечных временных сущностей, на которые декомпозируется исходная временная сущность]

- *прерывистая временная сущность*

:= [временная сущность, являющаяся результатом соединения нескольких не только точечных временных сущностей]

:= [временная сущность с прерываниями]

}

Следует отметить, что приведенная классификация *временных сущностей* характеризует не столько сами *временные сущности*, сколько наши знания о них и степень детализации знаний об этих сущностях, с которой они описаны в базе знаний. Так, если для решения конкретных задач не важно, как изменялась некоторая *временная сущность* в рамках какого-либо периода времени, а важно только ее начальное и конечное состояние, то она может рассматриваться как *точечная временная сущность*. Впоследствии же та же *временная сущность* может быть рассмотрена и описана с большей степенью детализации, и таким образом, уже не будет точечной.

Следует отличать:

- временный характер сущности, обозначаемой *sc-элементом*;
- временный характер существования самого *sc-элемента* в рамках *sc-памяти*, поскольку в ходе обработки информации каждый *sc-элемент* может быть удален из *sc-памяти*;
- временный характер описываемых ситуаций, событий и самих процессов;
- временный характер хранения в sc-памяти тех sc-конструкций, которые являются самими описаниями соответствующих ситуаций, событий и процессов.

Следует отличать, например, *материальную сущность* (некоторый физический или, в частности, биологический объект) от различных динамических структур (*процессов*), которые с той или иной степенью детализации и в том или ином ракурсе отражают (описывают) динамику изменений этой *материальной сущности*.

При этом сам *процесс* как уточнение динамики некоторой последовательности ситуаций и событий, также является сущностью, принадлежащей к классу *временных сущностей*.

прошлая сущность

:= [сущность, существовавшая в прошлом времени]

:= [сущность прошлого времени]

:= [сущность, завершившая свое существование]

настоящая сущность
:= [сущность, существующая в текущий момент времени]
:= [сущность, существующая сейчас]
:= [сущность настоящего времени]

будущая сущность
:= [возможно будущая сущность]
:= [прогнозируемая временная сущность]
:= [временная сущность, которая может существовать в будущем]
:= [сущность, которая может или должна начать свое существование в будущем времени]
⇒ *включение**:
инициированное действие

Каждой ***будущей сущности*** можно поставить в соответствие вероятность ее возникновения.

временная связь
:= [нестационарная связь]
:= [временно существующая связь]

Каждая ***временная связь*** представляет собой *связку*, принадлежащую множеству *временных сущностей*.

Понятие ***временной связи*** не следует путать с понятием *темпоральной связи*, которая сама является *постоянной сущностью*, описывающей то, как связаны во времени некоторые *временные сущности*.

ситуация
:= [состояние]
:= [временная структура]
:= [временно существующая структура]
:= [квазистационарная структура]
:= [состояние некоторой динамической системы, описываемое с некоторой степенью детализации (подробности)]
:= [квазистационарная структура, существующая временно (в течение некоторого отрезка времени)]

Под ситуацией понимается *структура*, содержащая, по крайней мере, один элемент, который является *временной сущностью*. Наличие в рамках ситуации нескольких *временных сущностей* говорит о том, что существует момент времени (в прошлом, настоящем или будущем), в который все они существуют одновременно.

процесс
:= [процесс преобразования некоторой временной сущности из одного состояния в другое]
:= [процесс перехода от одной ситуации к другой]
:= [абстрактный процесс]
:= [информационная модель некоторых преобразований]
:= [динамическая sc-модель]
:= [динамическая структура]
⇒ *включение**:
воздействие

Каждый ***процесс*** определяется (задается) либо возникновением или исчезновением связей, связывающих заданную *временную сущность* с другими сущностями, либо возникновением или исчезновением связей, связывающих части указанной *временной сущности* с другими сущностями.

Многим *процессам* можно поставить в соответствие *ситуацию*, которая является его *начальной ситуацией** и *ситуацию*, которая является его *конечной ситуацией**, то есть указать *ситуации*, переход между которыми осуществляется в ходе *процесса*.

При этом знаки тех *временных сущностей*, с которыми связаны изменения, описываемые некоторым *процессом*, входят в данный *процесс* как элементы и при необходимости уточняются дополнительными *ролевыми отношениями*.
⇒ *разбиение**:
{• *процесс в sc-памяти*
• *процесс во внешней среде ostis-системы*
}

Каждой ***материальной сущности*** можно поставить в соответствие различные *процессы*, описывающие ее преобразование из одного состояния в другое.

Поскольку *процесс* представляет собой изменяющуюся во времени динамическую структуру, то полностью представить процесс в базе знаний в общем случае не представляется возможным. Однако, можно ввести sc-элемент, обозначающий конкретный процесс, с необходимой степенью детализации описать его декомпозицию на более частные подпроцессы и/или описать ситуации, соответствующие состояниям процесса в разные моменты времени. В данном случае можно провести некоторую аналогию с *бесконечными множествами*, все элементы которых физически не могут быть представлены в базе знаний одновременно, тем не менее, само множество и некоторые из его элементов могут быть описаны с необходимой степенью детализации.

воздействие
:= [процесс, осуществляющийся на основе (в результате) воздействия одной сущности на другую]
⇒ *включение**:
действие

Каждому ***воздействию*** может быть поставлена в соответствие (1) некоторая *воздействующая сущность**, то есть сущность, осуществляющая *воздействие* (в частности, это может быть некоторое физическое поле), и (2) некоторый *объект воздействия**, то есть сущность, на которую воздействие направлено. Если *воздействие* связано с *материальной сущностью*, то его объектом воздействия является либо сама эта *материальная сущность*, либо некоторая ее пространственная часть.

точечный процесс
:= [атомарный процесс]
:= [условно мгновенный процесс]
:= [процесс, длительность которого в данном контексте считается несущественной (пренебрежимо малой)]
⊂ *точечная временная сущность*

элементарный процесс
:= [процесс, детализация которого на входящие в него подпроцессы в текущем контексте не производится]
⊃ *точечный процесс*

Элементарные процессы могут иметь длительность и, следовательно, не обязательно являются атомарными процессами.

Понятия *точечного процесса* и *элементарного процесса*, как и понятие *точечной временной сущности* в целом, характеризуют не столько характеристики самого *процесса*, сколько степень наших знаний о нем и степень детализации описания процесса в базе знаний. Так, очевидно, что любой процесс, протекающий в компьютерной системе, может быть при необходимости детализирован до уровня команд процессора, затем до уровня микропрограмм и даже до уровня физических процессов (изменения физических характеристик сигналов), однако чаще всего такая детализация не требуется.

событие
⊂ *точечная временная сущность*
:= [точечная временная сущность, являющаяся началом и/или завершением какой-либо временной сущности (например, процесса)]
:= [граничная точка временной сущности]

детализация процесса*
:= [бинарное ориентированное отношение, каждая связка которого связывает некоторый процесс с более детальным его описанием, что предполагает представление декомпозиции этого процесса на систему взаимосвязанных его подпроцессов (в том числе элементарных).]
⇒ *пример**:
Переход от процесса, соответствующего какой-либо программе, к рассмотрению декомпозиции этого процесса (протокола) в терминах языка программирования высокого уровня, затем переход для каждого из полученных подпроцессов (операторов языка высокого уровня) к детализации выполнения этих подпроцессов на уровне машинных операций, выполняемых процессором компьютера (на уровне ассемблера), и далее к детализации выполнения подпроцессов уровня машинных операций к подпроцессам на уровне языка микропрограммирования. Таким образом, детализация процесса может быть иерархической, вплоть до уровня элементарных процессов.

темпоральное отношение
⇐ *семейство подмножеств**:
темпоральная связь
:= [класс темпоральных связей]

:= [отношение, задающее темпоральные связи между временными сущностями]
∋ *темпоральное включение**
∋ *темпоральное объединение**
∋ *темпоральная декомпозиция**
∋ *темпоральная последовательность**
⇒ *разбиение**:
{• *темпоральная смежность**
• *темпоральная последовательность с промежутком**
• *темпоральная последовательность с пересечением**
}

темпоральное включение*
⇒ *пояснение**:
[Связки отношения ***темпоральное включение**** связывают две *временные сущности*, период существования одной из которых полностью включается в период существования второй.
Первым компонентом каждой связки отношения *темпоральное включение** является знак *временной сущности*, *длительность* существования которой больше.]
⊃ *темпоральная часть**
⊃ *темпоральное включение без совпадения начальных и конечных моментов**
⊃ *темпоральное совпадение**
⊃ *темпоральное включение с совпадением начальных моментов**
⊃ *темпоральное включение с совпадением конечных моментов**

темпоральная часть*
:= [этап (период) заданной временной сущности*]
:= [этап процесса существования временной сущности*]
⊃ *начальный этап**
⊃ *конечный этап**
⊃ *промежуточный этап**
⊃ *подпроцесс**
⇒ *первый домен**:
процесс
⇒ *второй домен**:
процесс

Связки отношения *темпоральная часть** связывают две *временные сущности*, одна из которых является частью другой, например, действие и одно из его поддействий. Соответственно, период существования одной из этих сущностей всегда будет включаться в период существования другой (большей).

В отличие от более общего отношения *темпоральное включение**, связки которого могут связывать любые *временные сущности*, связки отношения *темпоральная часть** связывают только *временные сущности*, одна из которых является частью другой.

следует отличать*
∋ {• *темпоральная часть**
⊃ *подпроцесс**
• *темпоральное включение**
⇒ *примечание**:
[Связь *темпорального включения** может связывать абсолютно разные *временные сущности*, существующие в общем случае в разных местах, а не только *временные сущности*, одна из которых является частью другой. Хотя формально и можно объединить любые разные *временные сущности* в одну общую *временную сущность*, далеко не всегда имеет смысл это делать.]
}

темпоральное включение без совпадения начальных и конечных моментов*
:= [строгое темпоральное включение*]

начало^
:= [одновременность начинаний^]
:= [класс одновременно начавшихся сущностей^]
∈ *параметр*

Каждый элемент множества **начало** представляет собой класс *временных сущностей*, у которых совпадает момент начала их существования. Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

завершение^
:= [конец^]
:= [одновременность завершений^]
:= [класс одновременно завершившихся сущностей^]
∈ *параметр*

Каждый элемент множества ***завершение*** представляет собой класс *временных сущностей*, у которых совпадает конечный момент их существования (момент завершения существования). Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

одновременность^
:= [параметр, значениями (элементами) которого являются классы либо одновременно существующих (происходящих) *точечных временных сущностей*, одновременность которых рассматривается с заданной степенью точности, либо одновременно начинающихся и заканчивающихся длительных процессов]

Важно отметить, что элементами некоторого значения параметра *одновременности* с заданной точностью могут быть только те временные сущности, которые и начались, и завершились в течение периода времени, заданного указанным значением этого параметра, но при этом начало и завершение этих временных сущностей не обязательно должно совпадать с началом и завершением указанного периода времени. Так, например, можно ввести значение параметра *одновременности* "*2022 год по Григорианскому календарю*", элементами которого будут все временные сущности, начавшие и закончившие свое существовавшие в рамках 2022 года. При этом не обязательно, чтобы эти временные сущности начались именно в полночь 1 января 2022 года и закончились в полночь 1 января 2023 года, это могут быть временные сущности, существовавшие, например, в течение июля 2022 года.

следует отличать*
∋ {• *темпоральное совпадение**
∈ *отношение эквивалентности*
• *одновременность^*
:= [фактор-множество для отношения темпоральное совпадение*]
}

длительность^
:= [класс временных сущностей, имеющих одинаковую длительность^]
∈ *параметр*
∋ *тысячелетие*
∋ *век*
∋ *год*
∋ *месяц*
∋ *день*
∋ *час*
∋ *минута*
∋ *секунда*

Каждый элемент множества ***длительность*** представляет собой класс *временных сущностей*, у которых совпадает длительность их существования. Конкретное значение данного *параметра* может быть как *точной величиной*, так и *неточной величиной* или *интервальной величиной*.

§ 2.4.6. Формальная онтология ситуаций и событий, описывающих динамику баз знаний ostis-систем

⇒ *ключевое понятие**:
- *sc-элемент*
- *событие в sc-памяти*
- *понятие, переходящее из основного в неосновное*
- *понятие, переходящее из неосновного в основное*

Обработка информации в *sc-памяти* (то есть динамика базы знаний, хранимой в *sc-памяти*) в конечном счете сводится:

- к появлению в *sc-памяти* новых актуальных *sc-узлов* и *sc-коннекторов*;
- к логическому удалению актуальных *sc-элементов*, то есть к переводу их в неактуальное состояние (это необходимо для хранения протокола изменения состояния базы знаний, в рамках которого могут описываться действия по удалению *sc-элементов*);
- к возврату логически удаленных *sc-элементов* в статус актуальных (необходимость в этом может возникнуть при откате базы знаний к какой-нибудь ее прошлой версии);
- к физическому удалению *sc-элементов*;
- к изменению состояния актуальных (логически не удаленных *sc-элементов*) — *sc-узел* может превратиться в *sc-ребро*, *sc-ребро* может превратиться в *sc-дугу*, *sc-дуга* может поменять направленность, *sc-дуга* общего вида может превратиться в *константную стационарную sc-дугу принадлежности*, и так далее;

Подчеркнем, что временный характер самого *sc-элемента* (так как он может появиться или исчезнуть) никак не связан с возможно временным характером сущности, обозначаемой этим *sc-элементом*. То есть временный характер самого sc-элемента и временный характер сущности, которую он обозначает — абсолютно разные вещи.

Таким образом, следует четко отличать динамику внешнего мира, описываемого базой знаний, а динамику самой базы знаний (динамику внутреннего мира). При этом очень важно, чтобы описание динамики базы знаний также входило в состав каждой базы знаний.

К числу понятий, используемых для описания динамики базы знаний относятся:

- логически удаленный sc-элемент;
- сформированное множество;
- вычисленное число;
- сформированное высказывание;

sc-элемент

⇒ *разбиение**:

{• *настоящий sc-элемент*

• *логически удаленный sc-элемент*

}

настоящий sc-элемент

∈ *ситуативное множество*

логически удаленный sc-элемент

∈ *ситуативное множество*

основное понятие

:= [основное понятие для данной ostis-системы]

∈ *ситуативное множество*

К ***основным понятиям*** относятся те понятия, которые активно используются в системе и могут быть ключевыми элементами sc-агентов. К *основным понятиям* относятся также все неопределяемые понятия.

неосновное понятие

:= [дополнительное понятие]

:= [вспомогательное понятие]

:= [неосновное понятие для данной ostis-системы]

∈ *ситуативное множество*

Каждое ***неосновное понятие*** должно быть строго определено на основе *основных понятий*. Такие *неосновные понятия* используются только для понимания и правильного восприятия вводимой информации, в том числе, для выравнивания онтологий. Ключевым элементом *sc-агентов неосновные понятия* быть не могут.

⇒ *правило идентификации экземпляров**:

[В случае, когда некоторое понятие полностью перешло из *основных понятий* в неосновные, то есть стало *неосновным понятием*, и соответствующее ему *основное понятие* (через которое оно определяется) в рамках некоторого внешнего языка имеет одинаковый с ним основной идентификатор, то к идентификатору *неосновного понятия* спереди добавляется знак #. Если при этом соответствующее *основное понятие* имеет в идентификаторе знак $, добавленный в процессе перехода, то этот знак удаляется. Если указанные понятия имеют разные основные идентификаторы в рамках этого внешнего языка, то никаких дополнительных средств идентификации не используется.

Например:
#трансляция sc-текста
#scp-программа]

понятие, переходящее из основного в неосновное
∈ *ситуативное множество*

понятие, переходящее из неосновного в основное
∈ *ситуативное множество*
⇒ *правило идентификации экземпляров**:
[В случае, когда текущее *основное понятие* и соответствующее ему ***понятие, переходящее из неосновного в основное*** в рамках некоторого внешнего языка имеют одинаковый основной идентификатор, то к идентификатору понятия, переходящего из неосновного в основное спереди добавляется знак $. Если указанные понятия имеют разные основные идентификаторы в рамках этого внешнего языка, то никаких дополнительных средств идентификации не используется.

Например:
$трансляция sc-текста
$scp-программа]

специфицированная сущность
⇒ *разбиение**:
{• *недостаточно специфицированная сущность*
• *достаточно специфицированная сущность*
}

К ***достаточно специфицированным сущностям*** предъявляются следующие требования:
- если сущность не является понятием, то для нее должны быть указаны
  - различные варианты обозначающих ее внешних знаков;
  - классы, которым она принадлежит;
  - связки, которыми она связана с другими сущностями (с указанием соответствующего отношения);
  - значения параметров, которыми она обладает;
  - те разделы базы знаний, в которых указанная сущность является ключевой;
  - предметные области, в которые данная сущность входит.
- если специфицированная сущность является понятием, то для нее должны быть указаны:
  - различные варианты внешних обозначений этого понятия;
  - предметные области, в которых это понятие исследуется;
  - определение понятия;
  - пояснения
  - разделы базы знаний, в которых это понятие является ключевым;
  - описание примера — пример экземпляра понятия.

событие в sc-памяти
⊃ *событие*

элементарное событие в sc-памяти
⊂ *событие в sc-памяти*
⇒ *разбиение**:
{• *событие добавления sc-дуги, выходящей из заданного sc-элемента*
• *событие добавления sc-дуги, входящей в заданный sc-элемент*
• *событие добавления sc-ребра, инцидентного заданному sc-элементу*
• *событие удаления sc-дуги, выходящей из заданного sc-элемента*
• *событие удаления sc-дуги, входящей в заданный sc-элемент*
• *событие удаления sc-ребра, инцидентного заданному sc-элементу*
• *событие удаления sc-элемента*
• *событие изменения содержимого файла*
}

Под ***элементарным событием в sc-памяти*** понимается такое *событие*, в результате выполнения которого изменяется состояние только одного *sc-элемента*.

точечный процесс
:= [атомарный процесс]
:= [условно мгновенный процесс]
:= [процесс, длительность которого в данном контексте считается несущественной (пренебрежимо малой)]

элементарный процесс
:= [процесс, детализация которого на входящие в него подпроцессы в текущем контексте не производится]

Глава 2.5.

Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий

⇒ *авторы**:
- *Голенков В. В.*
- *Банцевич К. А.*

⇒ *аннотация**:
[Глава посвящена онтологическому подходу к проектированию баз знаний интеллектуальных компьютерных систем нового поколения. Данный подход основан на представлении базы знаний как иерархической структуры взаимосвязанных предметных областей и их онтологий, построенных на базе онтологий верхнего уровня.]

⇒ *подраздел**:
- *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*
- *§ 2.5.2. Формализация понятия структуры*
- *§ 2.5.3. Формализация понятия семантической окрестности*
- *§ 2.5.4. Формализация понятия предметной области*
- *§ 2.5.5. Формализация понятия онтологии*
- *§ 2.5.6. Онтологии верхнего уровня*

⇒ *ключевое понятие**:
- *база знаний*
- *знание*
- *структура*
- *семантическая окрестность*
- *предметная область*
- *онтология*
- *онтология верхнего уровня*

⇒ *библиографическая ссылка**:
- *Давыденко.И.Т..СредсССМБЗ-2016cm*
- *Sowa.J.F.Top-lOC-1995art*
- *Davydenko I.T.OntolBDoIS-2017art*
- *Bantsevich K.A.Struc oKBoNGI-2022art*
- *ISOIECITTLOPR-2021el*
- *BasicFOB-2021el*
- *Arp R..BuildOwBFO-2015bk*
- *SuggestedUMO-2016el*
- *DOLCE-2009el*

Введение в Главу 2.5.

Развитие информационных технологий привело к расширению многообразия используемой информации, и в следствии, к необходимости создания *интеллектуальных компьютерных систем*, способных оперировать объемными информационными ресурсами. Важнейшими видами таких ресурсов являются *базы знаний*.

база знаний представляет собой систематизированную совокупность *знаний*, хранимую в памяти *интеллектуальной компьютерной системы* и достаточную для обеспечения целенаправленного (целесообразного, адекватного) функционирования (поведения) этой системы как в своей внешней среде, так и в своей внутренней среде (в собственной *базе знаний*).

Важным этапом разработки *баз знаний интеллектуальных компьютерных систем* является их структуризация. Структуризация *базы знаний*, то есть выделение в ней различных связанных между собой подструктур, необходима по целому ряду причин (см. *Давыденко.И.Т..СредсССМБЗ-2016cm*). В частности, это необходимо для обеспечения их синтаксической совместимости, что подразумевает унификацию формы представления *знаний*.

На сегодняшний день существуют десятки моделей представления *знаний*. Каждая из которых адаптирована для представления *знаний* определенного вида, в то время как при создании *интеллектуальных компьютерных*

систем часто возникает необходимость представить различные *виды знаний* в рамках одной *базы знаний*. Однако, в настоящее время ни одна из существующих моделей, взятых в отдельности, не может этого обеспечить.

В связи с этим возникает необходимость в создании универсальной структурированной модели представления *знаний*, которая позволила бы представлять любые *виды знаний* в унифицированном виде.

На сегодняшний день наиболее эффективным средством структуризации различных областей *знаний* являются *онтологии*. Суть онтологического подхода при проектировании *базы знаний* заключается в рассмотрении структуры *базы знаний* как иерархической системы выделенных *предметных областей* и соответствующих им *онтологий*. Однако, онтологически существует множество способов, которыми можно описать реальный мир таким, каким он есть. Решением данной проблемы является использование при проектировании *баз знаний интеллектуальных компьютерных систем онтологий верхнего уровня*.

Грамотно построенная *онтология верхнего уровня* позволит обеспечить широкую синтаксическую совместимость между большим количеством *онтологий* для различных предметных областей. Поскольку термины предметно-ориентированных *онтологий* подчинены терминам *онтологии* более высокого уровня (см. *Sowa.J.F.Top-lOC-1995art*).

На сегодняшний день был предпринят ряд попыток по созданию *онтологии верхного уровня*, которая бы позволила обеспечить широкую синтаксическую совместимость разрабатываемых *баз знаний интеллектуальных компьютерных систем*. Однако после анализа существующих *онтологий верхнего уровня*, который подробно представлен в *§ 2.5.6. Онтологии верхнего уровня*, был сделан вывод, что попытки создать универсальную *онтологию верхнего уровня*, способную обеспечить совместимость *интеллектуальных компьютерных систем*, не привели к ожидаемым результатам, поскольку имеют ряд ключевых недостатков.

Отсутствие *онтологии верхнего уровня*, которая бы позволила решить проблему проектирования и разработки *баз знаний интеллектуальных компьютерных систем*, приводит к несовместимости разрабатываемых *интеллектуальных компьютерных систем*. Исходя из вышесказанного, возникает необходимость в построении такой системы *онтологий верхнего уровня*, которая могла бы обеспечить синтаксическую совместимость между большим количеством *онтологий* различных *предметных областей баз знаний интеллектуальных компьютерных систем*.

Для решения представленных проблем необходимо согласовать трактовку таких понятий, как *структура*, *семантическая окрестность*, *предметная область*, *онтология*, поскольку данные понятия являются базовыми классами сущностей, составляющими основу для структуризации *баз знаний* интеллектуальных систем (см. *Davydenko I.T.OntolBDoIS-2017art*). .

Онтологическая модель, построенная на основе данных понятий, станет *Ядром базы знаний*, обеспечивающим совместимость интеллектуальных системы, за счет унифицированного представления знаний (см. *Bantsevich K.A.Struc oKBoNGI-2022art*). Следует отметить, что в зависимости от специфики разрабатываемых систем их *базы знаний* могут расширяться, однако, онтологическая модель, лежащая в основе *Ядра*, позволит обеспечить дальнейшую совместимость разрабатываемых систем.

§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем

⇒ *ключевой знак**:
- *Предметная область знаний и баз знаний ostis-систем*

⇒ *ключевое понятие**:
- *знание*
- *вид знаний*
- *отношение, заданное на множестве знаний*
- *база знаний*

В рамках модели *баз знаний* ostis-систем выделяются синтаксически корректные (для соответствующего языка) и семантически целостные информационные конструкции. Такие конструкции будем называть *знаниями*.

знание
:= [синтаксически корректная (для соответствующего языка) и семантически целостная информационная конструкция]
⊂ *информационная конструкция*
⇒ *покрытие**:
вид знаний
:= [Множество всевозможных видов знаний]

Тот факт, что семейство *видов знаний* является покрытием *Множества всевозможных знаний*, означает то, что каждое знание принадлежит по крайней мере одному выделенному нами *виду знаний*.

вид знаний
- ∋ *спецификация*
  - ≔ [описание заданной сущности]
  - ⊃ *спецификация материальной сущности*
  - ⊃ *спецификация обратной сущности, не являющейся множеством*
    - ⊃ *спецификация геометрической точки*
    - ⊃ *спецификация числа*
  - ⊃ *спецификация множества*
    - ⊃ *спецификация связи*
    - ⊃ *спецификация структуры*
    - ⊃ *спецификация класса*
      - ⊃ *спецификация класса сущностей, не являющихся множествами*
      - ⊃ *спецификация отношения*
        - ≔ [спецификация класса связей (связок)]
      - ⊃ *спецификация класса классов*
        - ⊃ *спецификация параметра*
      - ⊃ *спецификация класса структур*
      - ⊃ *спецификация понятий*
        - ⊃ *пояснение*
        - ⊃ *определение*
        - ⊃ *утверждение*
          - ≔ [утверждение, описывающее свойства экземпляров (элементов) специфицируемого понятия]
          - ≔ [закономерность]
  - ⊃ *семантическая окрестность*
  - ⊃ *однозначная спецификация*
  - ⊃ *сравнительный анализ*
  - ⊃ *достоинства*
  - ⊃ *недостатки*
  - ⊃ *структура специфицируемой сущности*
  - ⊃ *принципы, лежащие в основе*
  - ⊃ *обоснование предлагаемого решения*
    - ≔ [аргументация предлагаемого решения]
- ∋ *сравнение*
- ∋ *высказывание*
  - ⊃ *фактографическое высказывание*
  - ⊃ *закономерность*
- ∋ *формальная теория*
- ∋ *предметная область*
- ∋ *предметная область и онтология*
  - ≔ [предметная область и ее онтология]
  - ≔ [предметная область и соответствующая ей объединенная онтология]
- ∋ *метазнание*
  - ≔ [спецификация знания]
  - ⊃ *аннотация*
  - ⊃ *введение*
  - ⊃ *предисловие*
  - ⊃ *заключение*
  - ⊃ *онтология*
    - ⊃ *онтология предметной области*
      - ⊃ *структурная онтология предметной области*
      - ⊃ *теоретико-множественная онтология предметной области*
      - ⊃ *логическая онтология предметной области*
      - ⊃ *терминологическая онтология предметной области*
      - ⊃ *объединенная онтология предметной области*
- ∋ *задача*
  - ≔ [спецификация действия]
- ∋ *план*
- ∋ *протокол*

∋ *результативная часть протокола*
∋ *метод*
∋ *технология*
∋ *база знаний*

Даже небольшой перечень *видов знаний* свидетельствует об огромном многообразии *видов знаний*.

знания разделяются на *декларативные* и *процедурные*. Под *декларативными знаниями* понимаются *знания*, которые имеют только *денотационную семантику*, которая представляется в виде семантической *спецификации* системы понятий, используемых в этом *знании*. Под *процедурным знанием* понимается *знание*, имеющее не только *денотационную семантику*, но и *операционную семантику*, которая представляется в виде семейства *спецификаций агентов*, осуществляющих интерпретацию *процедурного знания*, направленную на решение некоторой инициированной задачи.

В рамках *Технологии OSTIS* также выделяются *отношения, заданные на множестве знаний*.

отношение, заданное на множестве знаний
∋ *дочернее знание**
  ≔ [знание, которое от "материнского" знания наследует все описанные там свойства объектов исследования]
  ⊃ *дочерний раздел**
    ≔ [частный раздел*]
  ⊃ *дочерняя предметная область и онтология**
∋ *спецификация**
  ≔ [быть знанием, которое является спецификацией (описанием) заданной сущности]
∋ *онтология**
  ≔ [быть семантической спецификацией заданного знания*]
∋ *семантическая эквивалентность**
∋ *следовательно**
  ≔ [логическое следствие*]
∋ *логическая эквивалентность**

Важным *видом знаний* являются *базы знаний*.

база знаний
≔ [совокупность знаний, хранимых в памяти интеллектуальной компьютерной системы и достаточных для того, чтобы указанная система удовлетворяла соответствующим предъявляемым к ней требованиям (в частности, чтобы она имела соответствующий уровень интеллекта)]
≔ [систематизированная совокупность знаний, хранимая в памяти интеллектуальной компьютерной системы и достаточная для обеспечения целенаправленного (целесообразного, адекватного) функционирования (поведения) этой системы как в своей внешней среде, так и в своей внутренней среде (в собственной базе знаний)]

Представленным понятиям соответствует *Предметная область знаний и баз знаний ostis-систем*, в которой они подробно рассматриваются.

Предметная область знаний и баз знаний ostis-систем
∈ *предметная область*
∋ *максимальный класс объектов исследования′*:
  знание
∋ *исследуемый класс классов′*:
  - *вид знаний*
  - *отношение, заданное на множестве знаний*

Важно отметить, что одним из основных факторов, определяющими качество *интеллектуальной компьютерной системы*, является качественная структуризация (систематизация) и стратификация *базы знаний интеллектуальной компьютерной системы*.

§ 2.5.2. Формализация понятия структуры

⇒ *ключевое понятие**:
  - *структура*

⇒ *ключевое знание**:

- *Классификация структур*

⇒ *ключевой знак**:
- *Предметная область структур*

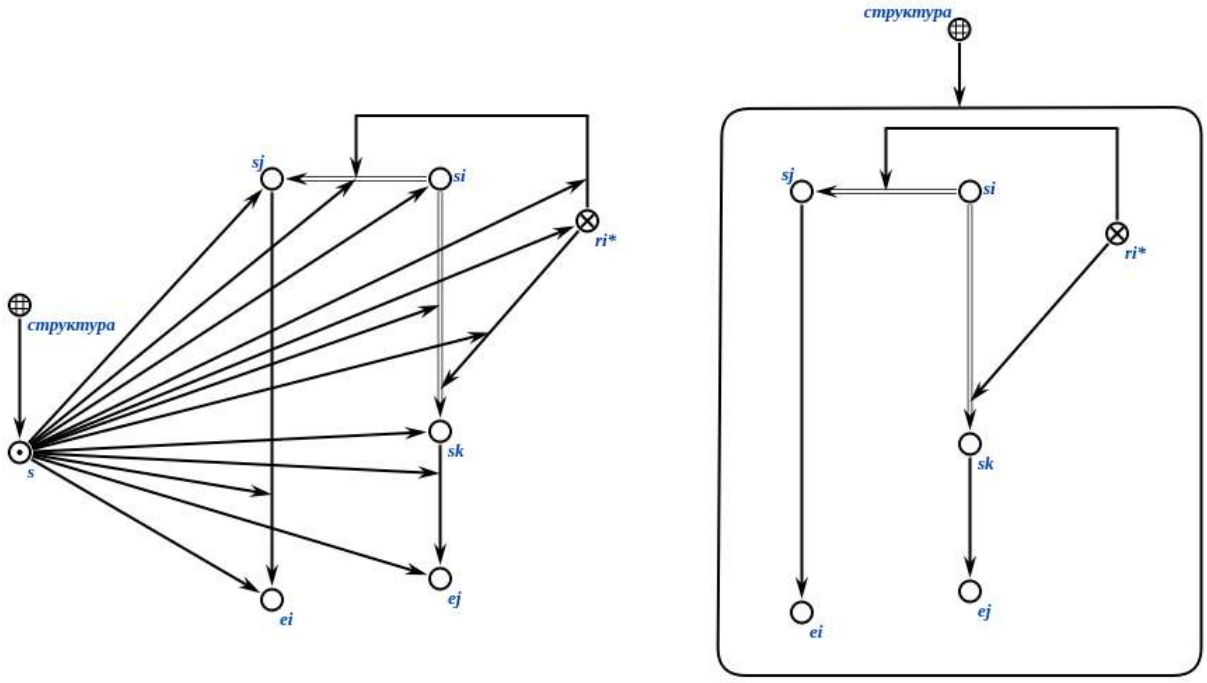
Существующие подходы к разработке *баз знаний* основываются на рассмотрении в качестве объектов спецификации конкретных элементов *базы знаний* (классов, экземпляров, отношений и так далее). Однако при накоплении больших объемов информации в *базе знаний* возникает необходимость выделять целые фрагменты *базы знаний* и иметь возможность их специфицировать, рассматривая как отдельные сущности.

Такой фрагмент базы знаний назван *структурой (sc-структурой)*. Под *структурой* будем понимать множество *sc-элементов*, удаление одного из которых может привести к нарушению целостности этого множества. Каждая *структура* представляет собой *sc-элемент*, обозначающий некоторый текст *SC-кода*, который может быть объектом спецификации, в том числе входить в состав других *структур*, быть связанным с другими сущностями различными отношениями.

Понятие *структуры* является основой для представления *знаний*, *метазнаний* и их структуризации, а также является одним из наиболее общих (с точки зрения уточнения семантики) понятий при описании свойств какого-либо объекта.

структура может изображаться путем явного указания всех пар принадлежности элементов этой *структуре*, а также в виде контура, содержащего все элементы, входящие в состав этой *структуры*, что продемонстрировано на рисунке *SCg-текст. Пример полного и сокращенного представления структуры в SCg-коде*.

SCg-текст. Пример полного и сокращенного представления структуры в SCg-коде
=



Рассмотрим типологию *структур*, описываемых в *базе знаний*.

структуре, представленной в *SC-коде*, поставим в соответствие орграф, вершинами которого являются *sc-элементы*, а дугами — связки отношений инцидентности, связывающие sc-коннекторы с инцидентными им *sc-элементами*, которые являются компонентами указанных sc-коннекторов. Если полученный таким способом орграф является связным орграфом, то исходную структуру будем считать *связной структурой*. Если полученный таким способом орграф не является связным орграфом, то исходную структуру будем считать *несвязной структурой*.

структура
⇒ *разбиение**:
- **{** *связная структура*
- *несвязная структура*

}

Под *тривиальной структурой* понимается *структура*, не содержащая в качестве элементов связок. В свою очередь, под *нетривиальной структурой* понимается *структура*, среди элементов которой есть хотя бы одна связка.

структура
⇒ *разбиение**:
{• *тривиальная структура*
• *нетривиальная структура*
}

По признаку стационарности/нестационарности выделяются *динамические структуры* (процессы) — *структуры*, состав которых меняется с течением времени, и *статические структуры* — *структуры*, состав которых не меняется с течением времени.

структура
⇒ *разбиение**:
{• *процесс*
:= [динамическая структура]
:= [нестационарная структура]
• *статическая структура*
:= [стационарная структура]
:= [структура, не изменяющаяся во времени]
}

По признаку времени существования выделяются *временные структуры* и *постоянно существующие структуры*.

структура
⇒ *разбиение**:
{• *временная структура*
:= [(структура ∩ временная сущность)]
• *постоянно существующая структура*
:= [(структура ∩ постоянная сущность)]
}

В рамках заданной *структуры* ее элементы можно классифицировать по заданным признакам:

- насколько полно в рамках заданной *структуры* представлено множество, обозначаемое *заданным sc-элементом* вместе с соответствующими дугами принадлежности;
- существуют ли в рамках заданной *структуры* *sc-элементы*, обозначающие множества, являющиеся надмножествами того множества, которое обозначается заданным *sc-элементом*;
- уровень («этаж») иерархии перехода от знаков к метазнакам для заданного *sc-элемента* в рамках заданной *структуры*.

Для формального представления *структур* используются понятия, описывающие роли элементов в рамках структуры. ***элемент структуры'*** — *неосновное понятие*, *ролевое отношение*, указывающее на все элементы каждой *структуры*. Пример описания элементов структуры представлен на рисунке *SCg-текст. Пример описания элементов структуры на SCg-коде*.

элемент структуры'
⇒ *разбиение**:
{• *непредставленное множество'*
:= [множество, не представленное в рамках данной структуры']
• *полностью представленное множество'*
:= [множество, полностью представленное в рамках данной структуры']
• *частично представленное множество'*
:= [множество, частично представленное в рамках данной структуры']
• *элемент структуры, не являющийся множеством'*
}

⇒ *разбиение**:
{• *максимальное множество′*
• *немаксимальное множество′*
}

максимальное множество′
⇒ *пояснение**:
[***максимальное множество′*** — *ролевое отношение*, связывающее *структуру* со знаком множества, для которого не существует множества, которое было бы надмножеством указанного множества и знак которого был бы элементом этой же структуры.]

немаксимальное множество′
⇒ *пояснение**:
[***немаксимальное множество′*** — *ролевое отношение*, связывающее *структуру* со знаком множества, для которого в рамках данной *структуры* существует множество, являющееся надмножеством указанного множества.]

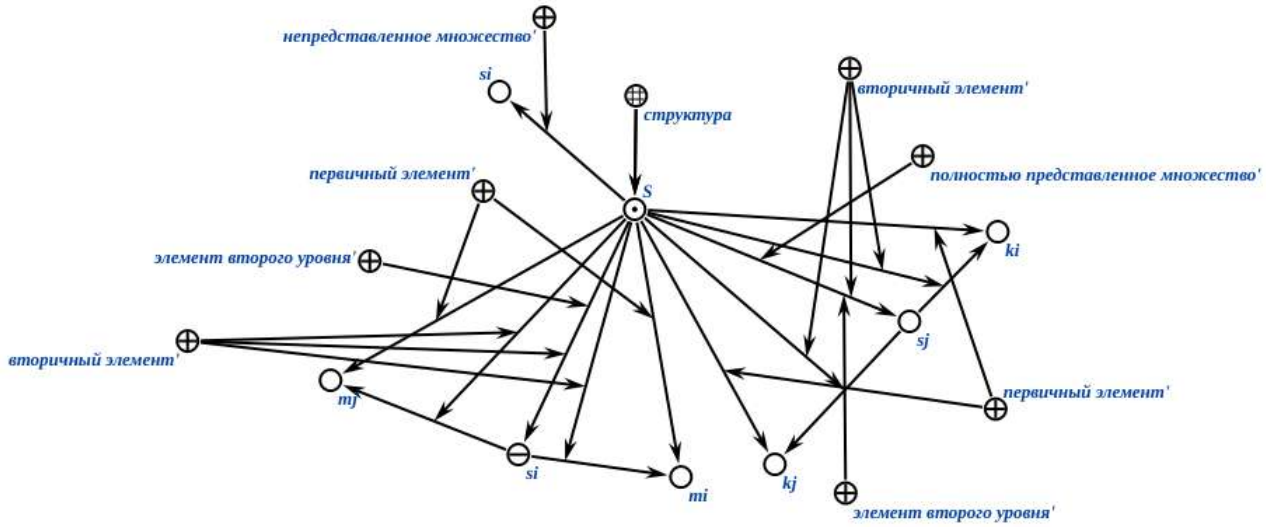
первичный элемент′
:= [первичный элемент данной структуры′]
:= [sc-элемент первого уровня в рамках данной структуры′]
∈ *ролевое отношение*
⊂ *элемент структуры′*
⇒ *пояснение**:
[***первичный элемент′*** — ролевое отношение, указывающее на элемент *структуры*, являющийся либо терминальным элементом, либо знаком множества, такого что не существует другого элемента этой же структуры, который был бы элементом множества, обозначаемого первым из указанных элементов структуры. При этом соответствующая пара принадлежности может существовать, но в состав данной структуры не входить.]

вторичный элемент′
:= [вторичный элемент данной структуры′]
:= [элемент данной структуры имеющий семантический уровень более 2′]
:= [непервичный элемент′]
∈ *ролевое отношение*
⊂ *элемент структуры′*
⇒ *пояснение**:
[***вторичный элемент′*** — ролевое отношение, указывающее на элемент структуры, обозначающий множество, все или некоторые элементы которого являются элементами указанной структуры.]
⊃ *элемент второго уровня′*

элемент второго уровня′
∈ *ролевое отношение*
⇒ *пояснение**:
[***элементом второго уровня′*** в рамках заданной *структуры* может быть связка первичных элементов, тривиальная структура из первичных элементов или класс первичных элементов.]

структура второго уровня′
⇒ *пояснение**:
[***структура второго уровня*** — *структура*, среди элементов которой есть хотя бы один *элемент второго уровня′*.]

SCg-текст. Пример описания элементов структуры на SCg-коде
=



Тот факт, что в качестве формальной основы представления *знаний* в *SC-коде* лежит теория графов и теория множеств, позволяет анализировать не только внешние связи рассматриваемого фрагмента *базы знаний* с другими элементами *базы знаний*, но и внутреннюю структуру этих фрагментов с необходимой степенью детализации, то есть выявлять в *базе знаний* аналогии, сходства, различия, строить различные виды соответствий между фрагментами.

Между *структурами* можно определять ряд соответствий, таких как *гомоморфизм*, *полиморфизм*, *автоморфизм*, *изоморфизм*, а также *аналогичность структур*, что позволяет фиксировать факт наличия некоторой аналогии, сходства и различия некоторых подструктур рассматриваемых *структур*.

полиморфность*
⊂ *соответствие**
∈ *бинарное отношение*
⇒ *пояснение**:
[*полиморфность** — это *соответствие*, заданное на *структурах*, при котором каждому элементу из области определения соответствия (первой *структуры*) ставится в соответствие один или более элемент из области значения соответствия (второй *структуры*), при этом существует хотя бы один элемент области определения соответствия, которому соответствуют два или более элемента из области значения соответствия.**]**

полиморфизм*
∈ *бинарное отношение*

гомоморфность*
:= [гомоморфность структур*]
⊂ *соответствие**
∈ *бинарное отношение*
⇒ *пояснение**:
[*гомоморфность** — это *соответствие*, заданное на *структурах*, при котором каждому элементу из области определения соответствия (первой *структуры*) ставится в соответствие только один элемент из области значения соответствия (второй *структуры*).**]**

гомоморфизм*
∈ *бинарное отношение*

изоморфность*
:= [изоморфное соответствие*]
:= [изоморфность структур*]
⊂ *гомоморфность**
∈ *бинарное отношение*
⇒ *пояснение**:

[*изоморфность** — это *гомоморфность**, при которой для каждого элемента из области значения существует ровно один соответствующий элемент из области определения.**]**

изоморфизм*
∈ *бинарное отношение*

автомоморфность*
⊂ *гомоморфность**
∈ *бинарное отношение*
⇒ *пояснение**:
[*автоморфность** — это *изоморфность**, у которой область определения соответствия и область значения соответствия совпадают.**]**

автоморфизм*
∈ *бинарное отношение*

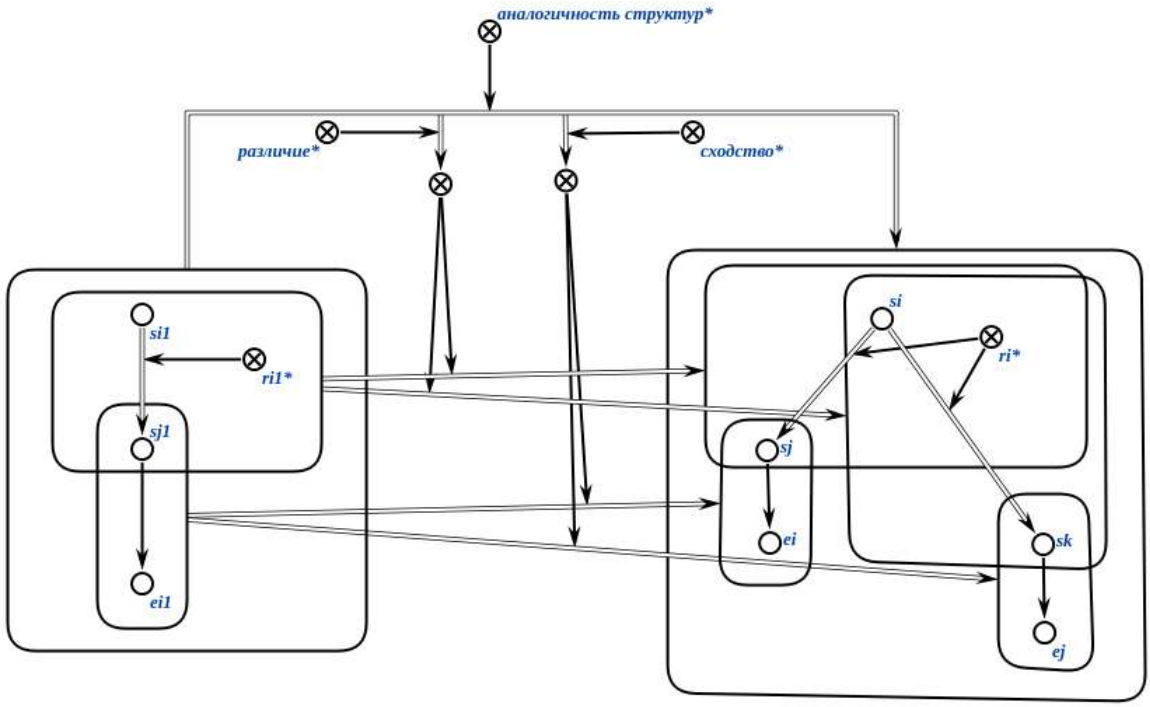
Отдельное внимание стоит уделить соответствию *аналогичность структур*, которое фиксирует факт наличия некоторой аналогии на подструктурах (подмножествах) указанных *структур*. Каждой ориентированной паре, принадлежащей *аналогичности структур*, может быть поставлено в соответствие множество пар, задающих *сходства* некоторых подструктур и *различия* некоторых подструктур исходных структур. Пример данного отношения представлен на рисунке *SCg-текст. Пример отношения аналогичности структур*.

аналогичность структур*
⊂ *соответствие**
∈ *бинарное отношение*
⇒ *пояснение**:
[*аналогичность структур** — *соответствие**, задаваемое на структурах, и фиксирующее факт наличия некоторой аналогии на подструктурах (подмножествах) указанных структур. Каждой ориентированной паре, принадлежащей ***аналогичности структур**** может быть поставлено в соответствие множество пар, задающих *сходства** некоторых подструктур и *различия** некоторых подструктур исходных структур.**]**

сходство*
∈ *бинарное отношение*

различие*
∈ *бинарное отношение*

SCg-текст. Пример отношения аналогичности структур
=



Соответствия на *структурах* являются частным случаем *метазнаний*. Используя такие соответствия, можно описывать в *базе знаний* и анализировать в дальнейшем, например, сходства и отличия различных фрагментов *базы знаний*, в том числе различных *предметных областей*.

Понятие *структуры* подробно рассматривается в *Предметной области структур*.

Предметная область структур
- ∈ *предметная область*
- ∋ *максимальный класс объектов исследования'*:
  структура
- ∋ *класс объектов исследования'*:
  - *связная структура*
  - *несвязная структура*
  - *тривиальная структура*
  - *нетривиальная структура*
- ∋ *исследуемое отношение'*:
  - *элемент структуры'*
  - *непредставленное множество'*
  - *полностью представленное множество'*
  - *частично представленное множество'*
  - *элемент структуры, не являющийся множеством'*
  - *максимальное множество'*
  - *немаксимальное множество'*
  - *первичный элемент'*
  - *вторичный элемент'*
  - *элемент второго уровня'*
  - *метасвязь'*
  - *полиморфность**
  - *полиморфизм**
  - *гомоморфность**
  - *гомоморфизм**
  - *изоморфность**
  - *изоморфизм**
  - *автоморфность**

- *автоморфизм**
- *аналогичность структур**
- *сходство**
- *различие**

§ 2.5.3. Формализация понятия семантической окрестности

⇒ *ключевой знак**:
- *Предметная область семантических окрестностей*

⇒ *ключевое понятие**:
- *семантическая окрестность*

⇒ *ключевое знание**:
- *Классификация семантических окрестностей*

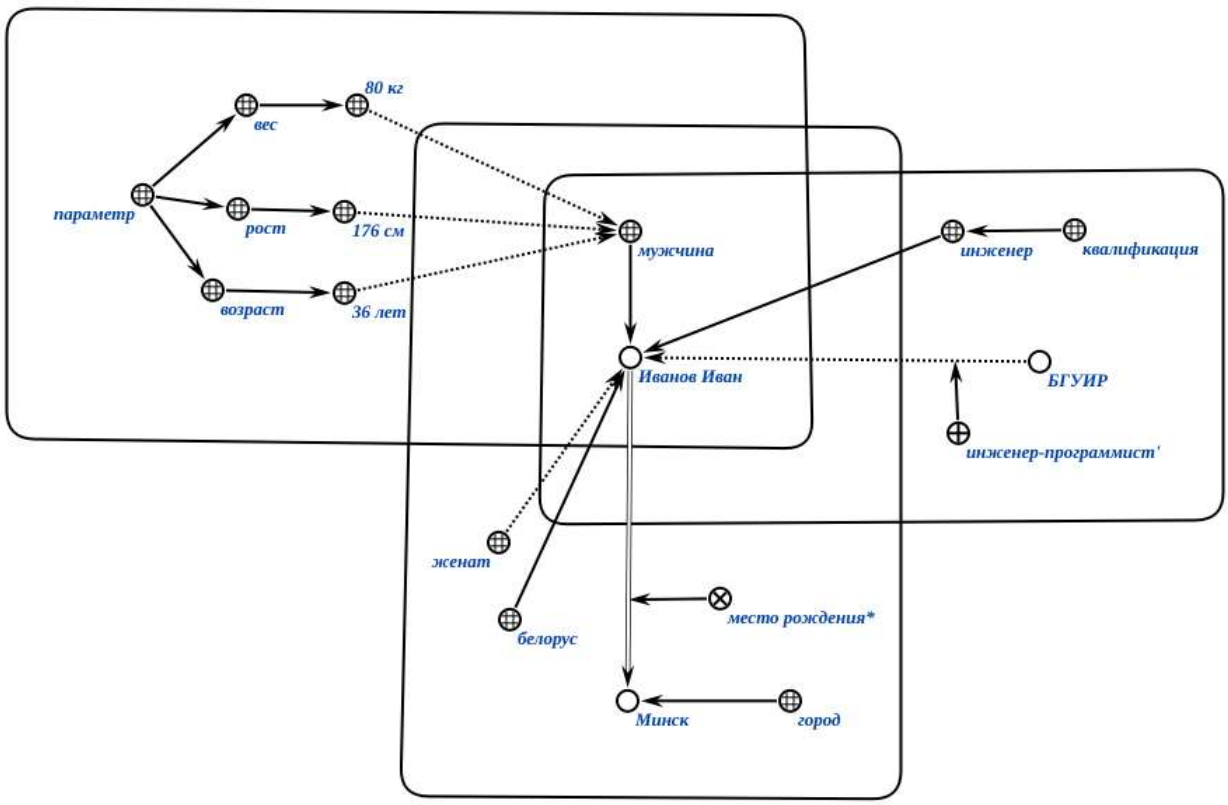
Для спецификации отдельных сущностей в рамках *базы знаний* вводится понятие *семантической окрестности*.

семантическая окрестность представляет собой спецификацию заданной сущности, знак которой указывается как ключевой элемент этой спецификации. В отличие от других *видов знаний*, *семантическая окрестность* имеет только один ключевой элемент.

Набор признаков, по которым можно специфицировать сущности, различен. Кроме того, может возникнуть необходимость специфицировать одну и ту же сущность в различных аспектах и явно фиксировать эти аспекты в *базе знаний*.

Так, например, одну и ту же персону можно описывать с профессиональной, медицинской, гражданской и других точек зрения, как представлено на рисунке *SCg-текст. Пример описания семантических окрестностей в SC-коде*.

SCg-текст. Пример описания семантических окрестностей в SC-коде
=



Понятие *семантической окрестности*, как и любой другой семантически выделяемый класс *знаний*, абсолютно не зависит от *языка представления знаний*. Этим *языком* может быть не только *SC-код* или другой *формальный язык*

представления знаний или даже *естественный язык*, тексты которых в *памяти ostis-системы* представляются в виде *файлов*.

Возможность описания различных свойств одного и того же объекта достигается за счет выделения различных классов *семантических окрестностей* и выделения набора признаков, определяющих тот или иной класс *семантических окрестностей*.

Перечислим основные виды *семантических окрестностей*.

семантическая окрестность
:= [sc-окрестность]
:= [семантическая окрестность, представленная в виде sc-текста]
:= [sc-текст, являющийся семантической окрестностью некоторого sc-элемента]
:= [спецификация заданной сущности, знак которой указывается как ключевой элемент этой спецификации]
:= [описание заданной сущности, знак которой указывается как ключевой элемент этой спецификации]
⊂ *знание*
⊃ *семантическая окрестность по инцидентным коннекторам*
⊃ *полная семантическая окрестность*
⊃ *базовая семантическая окрестность*
⊃ *специализированная семантическая окрестность*

семантическая окрестность по инцидентным коннекторам
⊃ *семантическая окрестность по выходящим дугам*
⊃ *семантическая окрестность по входящим дугам*
:= *пояснение**:
[вид *семантической окрестности*, в которую входят все коннекторы, инцидентные заданному элементу, а также все элементы, инцидентные указанным коннекторам.]

семантическая окрестность по выходящим дугам
⊃ *семантическая окрестность по выходящим дугам принадлежности*
:= *пояснение**:
[вид *семантической окрестности*, в которую входят все дуги, выходящие из заданного sc-элемента и вторые компоненты этих дуг. Также указывается факт принадлежности этих дуг каким-либо отношениям.]

семантическая окрестность по выходящим дугам принадлежности
:= *пояснение**:
[вид *семантической окрестности*, в которую входят все дуги принадлежности, выходящие из заданного *sc-элемента*, а также их вторые компоненты. При необходимости может указывается факт *принадлежности* этих дуг каким-либо *ролевым отношениям*.]

семантическая окрестность по входящим дугам
⊃ *семантическая окрестность по входящим дугам принадлежности*
:= *пояснение**:
[вид *семантической окрестности*, в которую входят все дуги, входящие в заданный sc-элемент, а также их первые компоненты. Также указывается факт принадлежности этих дуг каким-либо отношениям.]

семантическая окрестность по входящим дугам принадлежности
:= *пояснение**:
[вид *семантической окрестности*, в которую входят все дуги принадлежности, входящие в заданный sc-элемент, а также их первые компоненты. При необходимости может указывается факт принадлежности этих дуг каким-либо ролевым отношениям.]

Многообразие видов *семантических окрестностей* свидетельствует о многообразии семантических видов описаний различных сущностей.

Различают также *полную* и *базовую семантические окрестности*.

полная семантическая окрестность
:= [полная спецификация некоторой описываемой сущности]

Структура *полной семантической окрестности* определяется прежде всего семантической типологией описываемой сущности. Так, например, для понятия в *полную семантическую окрестность* необходимо включить следующую информацию (при наличии):
- варианты идентификации на различных внешних языках (sc-идентификаторы);
- принадлежность некоторой предметной области с указанием роли, выполняемой в рамках этой предметной области;
- теоретико-множественные связи заданного понятия с другими sc-элементами;
- определение или пояснение;
- высказывания, описывающие свойства указанного понятия;
- задачи и их классы, в которых данное понятие является ключевым;
- описание типичного примера использования указанного понятия;
- экземпляры описываемого понятия.

Для понятия, являющегося отношением дополнительно указываются:
- домены;
- область определения;
- схема отношения;
- классы отношений, которым принадлежит описываемое отношение.

базовая семантическая окрестность
:= [минимально достаточная семантическая окрестность]
:= [минимальная спецификация описываемой сущности]

Структура *базовой семантической окрестности* определяется прежде всего семантической типологией описываемой сущности. Так, например, для понятия в *базовую семантическую окрестность* необходимо включить следующую информацию (при наличии):
- варианты идентификации на различных внешних языках (sc-идентификаторы);
- принадлежность некоторой предметной области с указанием роли, выполняемой в рамках этой предметной области;
- определение или пояснение.

Для понятия, являющегося отношением дополнительно указываются:
- домены;
- область определения;
- описание типичного примера связки указанного отношения (спецификация типичного экземпляра).

Также выделяется *специализированная семантическая окрестность* — вид *семантической окрестности*, набор связей для которой уточняется отдельно для каждого типа такой окрестности.

специализированная семантическая окрестность
⊃ *пояснение*
  := [sc-пояснение]
  := *пояснение**:
    [знак sc-текста, поясняющего описываемую сущность.]
⊃ *примечание*
  := [sc-примечание]
  := *пояснение**:
    [знак sc-текста, являющегося примечанием к описываемой сущности. В примечании обычно описываются особые свойства и исключения из правил для описываемой сущности.]
⊃ *правило идентификации экземпляров*
  := [правило идентификации экземпляров заданного класса]
  := *пояснение**:
    [sc-текст являющийся описанием правил построения идентификаторов элементов заданного класса.]
⊃ *терминологическая семантическая окрестность*
  := *пояснение**:
    [*семантическая окрестность*, описывающая внешнюю идентификацию указанной сущности, то есть ее sc-идентификаторы]
⊃ *теоретико-множественная семантическая окрестность*
  := *пояснение**:
    [описание связи описываемого множества с другими множествами с помощью теоретико-множественных отношений]
⊃ *логическая семантическая окрестность*
  := *пояснение**:

[*семантическая окрестность*, описывающая семейство высказываний, описывающих свойства данного *понятия* или какого-либо конкретного экземпляра некоторого понятия]

⊃ *описание типичного экземпляра*

⊃ *описание декомпозиции*

Понятие *семантической окрестности*, дополненное уточнением таких понятий, как семантическое расстояние между знаками (семантическая близость знаков), радиус семантической окрестности, является перспективной основой для исследования свойств смыслового пространства.

Понятие *семантической окрестности*, предназначенное для спецификации отдельных сущностей в *базе знаний*, подробно рассматривается в *Предметной области семантических окрестностей*.

Предметная область семантических окрестностей

∈ *предметная область*

∋ *максимальный класс объектов исследования'*:
семантическая окрестность

∋ *класс объектов исследования'*:
- *семантическая окрестность по инцидентным коннекторам*
- *семантическая окрестность по выходящим дугам*
- *семантическая окрестность по выходящим дугам принадлежности*
- *семантическая окрестность по входящим дугам*
- *семантическая окрестность по входящим дугам принадлежности*
- *полная семантическая окрестность*
- *базовая семантическая окрестность*
- *специализированная семантическая окрестность*
- *терминологическая семантическая окрестность*
- *пояснение*
- *примечание*
- *правило идентификации экземпляров*
- *терминологическая семантическая окрестность*
- *теоретико-множественная семантическая окрестность*
- *логическая семантическая окрестность*

§ 2.5.4. Формализация понятия предметной области

⇒ *ключевой знак**:
- *Предметная область предметных областей*

⇒ *ключевое понятие**:
- *предметная область*

Важнейшим этапом разработки баз знаний является процесс выделения описываемых *предметных областей* и их представления в *базе знаний*.

Понятие ***предметной области*** является важнейшим методологическим приемом, позволяющим выделить из всего многообразия исследуемого Мира только определенный класс исследуемых сущностей и только определенное семейство отношений, заданных на указанном классе. То есть осуществляется локализация, фокусирование внимания только на этом, абстрагируясь от всего остального исследуемого Мира.

Каждой *предметной области* можно поставить в соответствие:
- семейство соответствующих ей онтологий разного вида;
- множество семантических окрестностей, описывающих объекты исследования этой предметной области.

предметные области являются основой структуризации смыслового пространства, средством локализации, фокусирования внимания на свойствах наиболее важных классов описываемых сущностей, которые становятся классами объектов исследования в *предметных областях*.

предметная область

:= [sc-модель предметной области]

:= [sc-текст предметной области]

:= [sc-граф предметной области]

:= [представление предметной области в *SC-коде*]

$\subset$ *знание*
$\subset$ *бесконечное множество*

предметная область — это результат интеграции (объединения) частичных семантических окрестностей, описывающих все исследуемые сущности заданного класса и имеющих одинаковый (общий) предмет исследования (то есть один и тот же набор отношений, которым должны принадлежать связки, входящие в состав интегрируемых семантических окрестностей).

предметная область — *структура*, в состав которой входят:
- основные исследуемые (описываемые) объекты — первичные и вторичные;
- различные классы исследуемых объектов;
- различные связки, компонентами которых являются исследуемые объекты (как первичные, так и вторичные), а также, возможно, другие такие связки — то есть связки (как и объекты исследования) могут иметь различный структурный уровень;
- различные классы указанных выше связок (то есть отношения);
- различные классы объектов, не являющихся ни объектами исследования, ни указанными выше связками, но являющихся компонентами этих связок.

При этом все классы, объявленные исследуемыми понятиями, должны быть полностью представлены в рамках данной *предметной области* вместе со своими элементами, элементами элементов и так далее вплоть до терминальных элементов.

Каждому типу знаний можно поставить в соответствие *предметную область*, которая является результатом интеграции всех знаний данного типа. Эти знания и становятся объектами исследования в рамках указанной *предметной области*.

предметная область
:= [система связей некоторого множества объектов исследования, ключевыми элементами которой являются:
- классы (точнее, знаки классов) объектов исследования (объектов, описываемых этой предметной областью);
- конкретные объекты исследования, обладающие особыми свойствами;
- классы связей, входящих в состав рассматриваемой системы – отношения, заданные на множестве элементов рассматриваемой системы;
- параметры, заданные на множестве элементов рассматриваемой системы;
- классы структур, являющихся фрагментами рассматриваемой системы.

]
:= [структура, представляющая собой множество связей (точнее, знаков связей) и соответствующее множество компонентов этих связей, к числу которых относится:
- элементы (экземпляры) некоторых заданных классов объектов исследования (первичных исследуемых сущностей);
- сами связи, входящие в состав указанной структуры;
- введенные классы объектов исследования;
- введенные отношения (классы связей);
- введенные параметры (классы классов эквивалентных сущностей);
- значения параметров (и, в частности, величины для измеряемых параметров);
- введенные структуры, являющиеся фрагментами (подструктурами) рассматриваемой структуры;
- введенные классы подструктур рассматриваемой структуры.

]

Выделяемые в рамках *базы знаний* интеллектуальной системы *предметные области* и соответствующие им *онтологии* — это, своего рода, семантические страты, кластеры, позволяющие "разложить" все хранимые в памяти *знания* по "семантическим полочкам" при наличии четких критериев, позволяющих однозначно определить то, на какой "полочке" должны находиться те или иные *знания*.

По уровню исследовательского внимания понятия в рамках *предметной области* могут выполнять следующие роли:

роль элемента предметной области
:= [ролевое отношения, связывающее предметные области с их ключевыми знаками]
:= [роль ключевого элемента (знака ключевой сущностей) предметной области]
:= [роль ключевого знака предметной области]
∋ *класс объектов исследования'*
:= [быть классом первичных (для данной предметной области) объектов исследования']
∋ *максимальный класс объектов исследования'*
:= [класс объектов исследования, для которого в заданной (!) предметной области отсутствует другой класс объектов исследования, который был бы его надмножеством']
∋ *ключевой объект исследования'*
:= [особый объект исследования']
:= [быть знаком особого исследуемого объекта в рамках заданной предметной области']
:= [объект исследования, обладающий особыми свойствами']
∋ *понятие, используемое в предметной области'*
:= [понятие, используемое в заданной предметной области не в качестве одного из объектов исследования, а в качестве ключевого понятия']
∋ *первичный исследуемый элемент предметной области'*
:= [знак первичного объекта исследования в рамках заданной предметной области']
∋ *вторичный исследуемый элемент предметной области'*
:= [знак вторичного объекта исследования в рамках предметной области']
∋ *неисследуемый элемент предметной области'*
:= [вспомогательный элемент предметной области, исследуемый в другой (смежной) предметной области']

Выделяются следующие типы *предметных областей*:

предметная область
⇒ *разбиение**:
{• *статическая предметная область*
:= [стационарная предметная область]
:= [*предметная область*, в которой связи между сущностями, входящими в ее состав, не зависят от времени (не меняются во времени), элементами ***статической предметной области*** не могут быть *временные сущности*]
• *квазистатическая предметная область*
:= [*предметная область*, решение задач в которой не требует учета темпоральных свойств объектов исследования]
• *динамическая предметная область*
:= [нестационарная предметная область]
:= [*предметная область*, которая описывает изменение состояния (в том числе внутренней структуры) объектов исследования и/или изменение конфигурации связей между объектами исследования]
:= [*предметная область*, в которой некоторые связи между сущностями, входящими в ее состав, меняются со временем (то есть носят ситуационный, нестационарный характер, другими словами, являются *временными сущностями*)]
}
⇒ *разбиение**:
{• *первичная предметная область*
:= [*предметная область*, объектами исследования которой являются внешние сущности (обозначаемые первичными *sc-элементами*)]
• *вторичная предметная область*
:= [метапредметная область]
:= [*предметная область*, объектами исследования которой являются *sc-множества* (отношения, параметры, структуры, классы структур, знания, языки и так далее)]
}

Во всем многообразии предметных областей особое место занимают:
- ***Предметная область предметных областей***, объектами исследования которой являются всевозможные *предметные области*, а предметом исследования являются — всевозможные ролевые отношения, связывающие *предметные области* с их элементами, отношения, связывающие *предметные области* между собой, отношение, связывающее *предметные области* с их *онтологиями*;
- ***Предметная область сущностей***, являющаяся *предметной областью* самого высокого уровня и задающая базовую семантическую типологию *sc-элементов* (знаков, входящих в тексты *SC-кода*);

- Семейство *предметных областей*, каждая из которых задает семантику и синтаксис некоторого *sc-языка*, обеспечивающего представление *онтологий* соответствующего вида (например, теоретико множественных онтологий терминологических онтологий);
- Семейство *предметных областей* верхнего уровня, в которых классами объектов исследования являются весьма "крупные" классы сущностей. К таким классам, в частности, относятся:
  - класс всевозможных материальных сущностей,
  - класс всевозможных множеств,
  - класс всевозможных связей,
  - класс всевозможных отношений,
  - класс всевозможных структур,
  - класс всевозможных темпоральных (нестационарных) сущностей,
  - класс всевозможных действий (воздействий, акций),
  - класс всевозможных параметров (характеристик),
  - класс знаний всевозможного вида и тому подобное.

Важно отметить, что *предметную область* также можно считать *семантической окрестностью*, если считать ее центром знак сущности, которая является максимальным классом объектов исследования.

Понятие *предметной области* подробно рассматривается в *Предметной области предметных областей*. В состав ***Предметной области предметных областей*** входят структурные спецификации всех *предметных областей*, входящих в состав базы знаний *ostis-системы*, в том числе, самой ***Предметной области предметных областей***. Таким образом, ***Предметная область предметных областей*** является, во-первых, *рефлексивным множеством*, во-вторых, рефлексивной предметной областью, то есть *предметной областью*, одним из объектов исследования которой является она сама.

Предметная область предметных областей
:= [Предметная область, объектами исследования которой являются предметные области]
∈ *рефлексивное множество*
∋ *максимальный класс объектов исследования'*:
предметная область
∋ *класс объектов исследования'*:
- *статическая предметная область*
- *динамическая предметная область*
- *понятие*
- *sc-язык*

∋ *исследуемое отношение'*:
- *понятие предметной области'*
- *исследуемое понятие'*
- *максимальный класс объектов исследования'*
- *немаксимальный класс объектов исследования'*
- *исследуемый класс первичных элементов'*
- *исследуемое отношение'*
- *класс исследуемых структур'*
- *дочерняя предметная область**
- *понятие, исследуемое в дочерней предметной области'*
- *понятие, исследуемое в материнской предметной области'*

§ 2.5.5. Формализация понятия онтологии

⇒ *ключевой знак**:
- *Предметная область онтологий*

⇒ *ключевое понятие**:
- *онтология*

Для формальной спецификации соответствующей предметной области, ориентированной на описание свойств и взаимосвязей понятий, входящих в состав указанной *предметной области*, используется такой *вид знаний*, как *онтология*.

онтологии являются важнейшим *видом знаний*, обеспечивающих семантическую систематизацию *знаний*, хранимых в памяти *интеллектуальных компьютерных систем* (в т.ч. *ostis-систем*), и, соответственно, семантическую структуризацию *баз знаний*.

онтология
:= [sc-онтология]
:= [семантическая *спецификация* некоторого достаточно информативного ресурса (знания)]
⊂ *спецификация*
⊂ *метазнание*
∈ *вид знаний*
:= [важнейший вид *метазнаний*, входящих в состав базы знаний]
:= [спецификация (уточнение) системы *понятий*, используемых в соответствующем (специфицируемом) *знании*]

онтология включает в себя:
- типологию специфицируемого знания;
- связи специфицируемого знания с другими знаниями;
- спецификацию ключевых понятий, используемых в специфицируемом знании, а также ключевых экземпляров некоторых таких понятий.

Важно отметить, что если *спецификация* может специфицировать (описывать) любую *сущность*, то *онтология* специфицирует только различные *знания*. При этом наиболее важными объектами такой спецификации являются *предметные области*.

Основная *цель* построения *онтологии* — семантическое уточнение (пояснение, а в идеале — определение) такого семейства *знаков*, используемых в заданном *знании*, которых достаточно для понимания смысла всего специфицируемого *знания*. Как выясняется, количество знаков, смысл которых определяет смысл всего специфицируемого *знания*, не является большим.

онтология
⇒ *разбиение**:
{• *неформальная онтология*
• *формальная онтология*
:= [онтология, представленная на формальном языке]
:= [формальное описание денотационной семантики (семантической интерпретации) специфицируемого знания]
}

Очевидно, что при отсутствии достаточно полных формальных *онтологий* невозможно обеспечить семантическую совместимость (интегрируемость) различных знаний, хранимых в *базе знаний*, а также приобретаемых извне.

онтология чаще всего трактуется как спецификация концептуализации (спецификация системы *понятий*) заданной *предметной области*. Здесь имеется в виду описание теоретико-множественных связей (прежде всего, классификации) используемых *понятий*, а также описание различных закономерностей для сущностей, принадлежащих этим *понятиям*. Тем не менее, важными видами спецификации *предметной области* являются также:

- описание связей специфицируемой *предметной области* с другими *предметными областями*;
- описание терминологии специфицируемой *предметной области*.

онтология предметной области
:= [описание *денотационной семантики* языка, определяемого (задаваемого) соответствующей (специфицируемой) *предметной области*]
:= [информационная надстройка (метаинформация) над соответствующей (специфицируемой) *предметной областью*, описывающая различные аспекты этой *предметной области* как достаточно крупного, самодостаточного и семантически целостного фрагмента *база знаний*]
:= [метаинформация (метазнание) о некоторой *предметной области*]

онтологию предметной области можно трактовать, с одной стороны, как *семантическую окрестность* соответствующей *предметной области*, с другой стороны, как *объединение* определенного вида *семантических окрестностей* всех *понятий*, используемых в рамках указанной *предметной области*, а также, возможно, ключевых экземпляров указанных *понятий*, если таковые экземпляры имеются.

Каждая конкретная *онтология* заданного вида представляет собой *семантическую окрестность* соответствующей (специфицируемой) *предметной области*. Каждому *виду онтологий* однозначно соответствует *предметная область*, фрагментами которые являются конкретные *онтологии* этого вида. Следовательно, каждому *виду онтологий* соответствует свой специализированный sc-язык, обеспечивающий представление *онтологий* этого вида.

онтология предметной области
⇒ *разбиение**:
{• *частная онтология предметной области*
:= [*онтология*, представляющая спецификацию соответствующей предметной области в том или ином аспекте]
• *объединенная онтология предметной области*
:= [онтология *предметной области*, являющаяся результатом объединения всех известных *частных онтологий* этой предметной области]
}

Каждая *частная онтология* является фрагментом *предметной области*, включающей в себя все(!) частные онтологии, принадлежащие соответствующему *виду онтологии*. При этом указанная *предметная область*, в свою очередь, также имеет соответствующую ей *онтологию*, которая уже является не метазнанием (как любая онтология), а метаметазнанием (спецификацией метазнания).

частная онтология предметной области
⇒ *разбиение**:
{• *структурная спецификация предметной области*
:= [вид *метазнаний*, описывающих соответствующие этому виду метазнаний свойства *предметных областей*]
:= [схема предметной области]
• *теоретико-множественная онтология предметной области*
:= [sc-спецификация заданной предметной области в рамках *Предметной области множеств*]
• *логическая онтология предметной области*
:= [sc-текст формальной теории заданной предметной области]
• *терминологическая онтология предметной области*
}

структурная спецификация предметной области
:= [структурная онтология предметной области]
:= [ролевая структура ключевых элементов предметной области]
:= [схема ролей понятий предметной области и ее связи со смежными предметными областями]
:= [схема предметной области]
:= [спецификация предметной области с точки зрения теории графов и теории *алгебраических систем*]
:= [описание внутренней (ролевой) структуры *предметной области*, а также ее внешних связей с другими *предметными областями*]
:= [описание ролей ключевых элементов предметной области (прежде всего, понятий – концептов), а также "место" специфицируемой предметной области в множестве себе подобных]
:= [*семантическая окрестность* знака *предметной области* в рамках самой этой *предметной области*, включающая в себя все *ключевые знаки*, входящие в состав *предметной области* (ключевые понятия и ключевые объекты исследования предметной области) с указанием их ролей (свойств) в рамках этой *предметной области* и *семантическая окрестность* знака специфицируемой *предметной области* в рамках *Предметной области предметных областей*, включающая в себя связи специфицируемой *предметной области* с другими семантически близкими ей *предметными областями* (дочерними и родительскими, аналогичными в том или ином смысле (например, изоморфными), имеющими одинаковые *классы объектов исследования* или одинаковые наборы *исследуемых отношений*)]

теоретико-множественная онтология предметной области
:= [*семантическая окрестность* специфицируемой *предметной области* в рамках *Предметной области множеств*, описывающая теоретико-множественные связи между *понятиями* специфицируемой *предметной области*, включая связи *отношений* с их *областями определения* и *доменами*, связи используемых *параметров* и классов структур их *областями определения*]
:= [онтология описывающая:

- классификацию объектов исследования специфицируемой предметной области;

- соотношение областей определения и доменов используемых отношений с выделенным классами объектов исследования, а также с выделенными классами вспомогательных (смежных) объектов, не являющихся объектами исследования в специфицируемой предметной области;
- спецификацию используемых отношений и, в том числе, указание того, все ли связки этих отношений входят в состав специфицируемой предметной области.

]

теоретико-множественная онтология предметной области включает в себя:
- теоретико-множественные связи (в т.ч. таксономию) между всеми используемыми понятиями, входящими в состав специфицируемой предметной области;
- теоретико-множественную спецификацию всех *отношений*, входящих в состав специфицируемой предметной области (ориентированность, арность, область определения, домены и так далее);
- теоретико-множественную спецификацию всех параметров, используемых в предметной области (области определения параметров, шкалы, единицы измерения, точки отсчета);
- теоретико-множественную спецификацию всех используемых классов структур.

логическая онтология предметной области
:= [формальная теория заданной (специфицируемой) предметной области, описывающая с помощью переменных, кванторов, логических связок, формул различные свойства экземпляров понятий, используемых в специфицируемой предметной области]

логическая онтология предметной области включает в себя:
- формальные определения всех понятий, которые в рамках специфицируемой предметной области являются определяемыми;
- неформальные пояснения и некоторые формальные спецификации (как минимум, примеры) для всех понятий, которые в рамках специфицируемой предметной области являются неопределяемыми;
- иерархическую систему понятий, в которой для каждого понятия, исследуемого в специфицируемой предметной области либо указывается факт неопределяемости этого понятия, либо указываются все понятия, на основе которых дается определение данному понятию. В результате этого множество исследуемых понятий разбивается на ряд уровней:
  - неопределяемые понятия;
  - понятия 1-го уровня, определяемые только на основе неопределяемых понятий;
  - понятия 2-го уровня, определяемые на основе понятий, изменяющих 1-й уровень и ниже;
  - и так далее.
- формальную запись всех аксиом, то есть высказываний, которые не требуют доказательств;
- формальную запись высказываний, истинность которых требует обоснования (доказательства);
- формальные тексты доказательства истинности высказываний, представляющие собой спецификацию последовательности шагов соответствующих рассуждений (шагов логического вывода, применения различных правил логического вывода);
- иерархическую систему высказываний, в которой для каждого высказывания, истинного по отношению к специфицируемой предметной области, либо указывается аксиоматичность этого высказывания, либо перечисляются все высказывания, на основе которых доказывается данное высказывание. В результате этого множество высказываний, истинных по отношению к специфицируемой предметной области, разбивается на ряд уровней:
  - аксиомы;
  - высказывания 1-го уровня, доказываемые только на основе аксиом;
  - высказывания 2-го уровня, доказываемые на основе высказываний, находящихся на 1-м уровне и ниже.
- формальная запись гипотетических высказываний;
- формальное описание логико-семантической типологии высказываний – высказываний о существовании, о несуществовании, об однозначности, высказывания определяющего типа (которые можно использовать в качестве определений соответствующих понятий);
- формальное описание различного вида логико-семантических связей между высказываниями (например, между высказыванием и его обобщением);
- формальное описание аналогии
  - между определениями;
  - между высказываниями любого вида;
  - между доказательствами различных высказываний.

терминологическая онтология предметной области
:= [онтология, описывающая правила построения терминов (sc-идентификаторов), соответствующих sc-элементам, принадлежащим специфицируемой предметной области, а также описывающая различного рода терминологические связи между используемыми терминами, характеризующие происхождение этих терминов]
:= [система терминов заданной предметной области]
:= [тезаурус соответствующей предметной области]
:= [словарь соответствующей (специфицируемой) предметной области]
:= [фрагмент глобальной *Предметной области sc-идентификаторов* (внешних идентификаторов sc-элементов), обеспечивающий терминологическую спецификацию некоторой предметной области]

Теперь подробнее рассмотрим понятие *объединенной онтологии предметной области*.

объединенная онтология предметной области
:= [объединение всех частных онтологий, соответствующих одной предметной области]
⇐ *обобщенное объединение**:
{• *структурная спецификация предметной области*
• *теоретико-множественная онтология предметной области*
• *логическая онтология предметной области*
• *терминологическая онтология предметной области*
}

предметная область и онтология
:= [интеграция некоторой *предметной области* с соответствующей ей *объединенной онтологией*]
:= [предметная область & онтология]
⇐ *обобщенное объединение**:
{• *предметная область*
• *объединенная онтология предметной области*
}
:= [sc-текст, являющийся объединением некоторой предметной области, представленной в *SC-коде*, и объединенной онтологии этой предметной области, также представленной в *SC-коде*]
:= [интеграция предметной области и всех онтологий, специфицирующих эту предметную область]
:= [совокупность различных *фактов* о структуре некоторой области деятельности некоторых *субъектов*, а также различного вида *знаний*, специфицирующих эту область деятельности]
:= [факты и знания о некоторой области деятельность]
:= [sc-модель предметной области и всевозможных онтологий, специфицирующих эту предметную область (и, в первую очередь, ее ключевых понятий) в разных ракурсах]
:= [целостный с логико-семантической точки зрения фрагмент базы знаний ostis-системы, акцентирующий внимание на конкретном классе объектов исследования и на конкретном аспекте их рассмотрения]

предметные области и онтологии являются основным *видом разделов баз знаний*, обладающих высокой степенью их независимости друг от друга и четкими правилами их согласования, что обеспечивает их семантическую (понятную) совместимость в рамках всей *базы знаний*.

онтология и связанные понятия подробно специфицируются в *Предметной области онтологий*.

Предметная область онтологий
:= [Предметная область теории онтологий]
:= [Предметная область, объектами исследования которой являются *онтологии*]
∈ *предметная область*
∋ *максимальный класс объектов исследования'*:
онтология
∋ *класс объектов исследования'*:
• *объединенная онтология*
• *структурная спецификация предметной области*
• *теоретико-множественная онтология предметной области*
• *логическая онтология предметной области*
• *логическая иерархия понятий предметной области*
• *логическая иерархия высказываний предметной области*
• *терминологическая онтология предметной области*
∋ *исследуемое отношение'*:
• *онтология**

- *используемые константы**
- *используемые утверждения**

§ 2.5.6. Онтологии верхнего уровня

⇒ *ключевое понятие**:
- *онтология верхнего уровня*

⇒ *библиографическая ссылка**:
- *ISOIECITTLOPR-2021el*
- *BasicFOB-2021el*
- *Arp R..BuildOwBFO-2015bk*
- *SuggestedUMO-2016el*
- *DOLCE-2009el*

Решением проблемы обеспечения совместимости различных *видов знаний* является использование при проектировании *баз знаний интеллектуальных компьютерных систем онтологий верхнего уровня*.

Грамотно построенная *онтология верхнего уровня* позволит обеспечить широкую семантическую совместимость между большим количеством *онтологий* для различных *предметных областей*. Поскольку термины предметно-ориентированных *онтологий* подчинены терминам *онтологии* более высокого уровня.

онтология верхнего уровня
⊃ *онтология*
:= [онтология, описывающая фундаментальные понятия, которые являются общими для всех предметных областей]
:= [онтология, систематизирующая знания о реальном мире безотносительно к какой-либо конкретной предметной области]
⇒ *цель**:
[поддержка семантической совместимости онтологий предметных областей и прикладных онтологий]

Согласно стандарту ISO/IEC 21838-1:2021 (см. *ISOIECITTLOPR-2021el*) были сформулированы требования, согласно которым *онтология* может считаться *онтологией верхнего уровня*. Первой принятой по ISO/IEC 21838-1:2021 *онтологией верхнего уровня* является BFO (Basic Formal Ontology) (см. *BasicFOB-2021el*).

BFO — *онтология верхнего уровня*, разработанная Барри Смитом и его коллегами для обеспечения совместимости между суб-онтологиями предметных областей, построенными в ее терминах.

Руководство по построению суб-онтологий предметных областей, соответствующих BFO, было опубликовано MIT Press в 2015 году (см. *Arp R..BuildOwBFO-2015bk*).

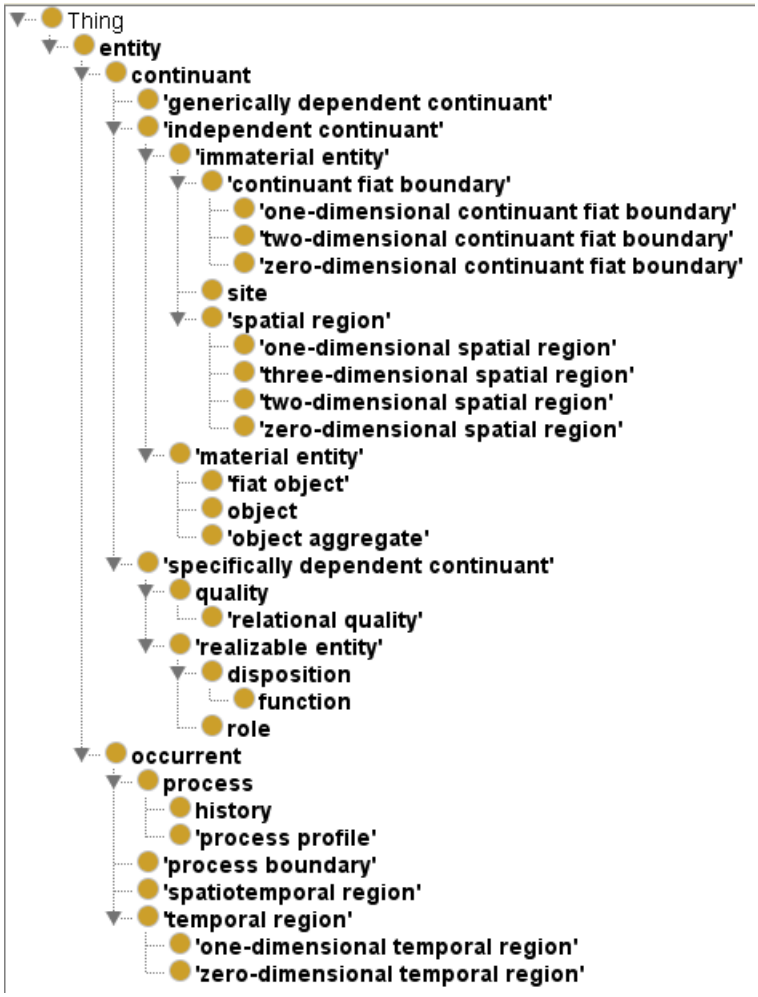
В поддержку BFO разработан стандарт ISO/IEC 21838-2.

Структура BFO основана на разделении сущностей на две категории:
- непрерывные сущности, такие как трехмерные объекты;
- возникающие сущности, такие как процессы.

BFO содержит определения своих терминов и выражений отношения и форм реализации на языках OWL2 и Общей логики (Common Logic, CL).

Иерархия понятий, используемых в BFO, представлена на рисунке *Рисунок. Иерархия понятий, используемых в BFO*.

Рисунок. Иерархия понятий, используемых в BFO
=



BFO — *онтология*, разработанная для поддержки совместимости данных и информационных систем, связанных с *онтологиями*, содержащими более конкретные термины, относящиеся к конкретным *предметным областям*.

Основная цель BFO — поддерживать разработку таких *онтологий предметной области* с тем, чтобы способствовать координации работы различных групп специалистов, добиваться непротиворечивости и предотвращать избыточность *онтологий*.

Стоит отметить, что многие *предметные области и онтологии*, описанные в *Главе 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*, пересекаются с областями выделенными в BFO, но при этом BFO является более конкретной предметной онтологией.

При этом онтология BFO не учитывает, например, ситуации и события, которые бы описывали динамику *базы знаний*, что определенно необходимо для интеллектуальной системы.

Также имеет смысл рассмотреть следующие онтологии верхнего уровня:
- **The Standard Upper Ontology** (SUMO) (см. *SuggestedUMO-2016el*)
- **Descriptive Ontology for Linguistic and Cognitive Engineering** (DOLCE) (см. *DOLCE-2009el*)

SUMO — это онтология верхнего уровня, предназначена для использования в качестве базовой онтологии для различных компьютерных систем обработки информации. Данный проект образует большую формальную общедоступную онтологию и претендует на статус формирования стандарта для онтологий верхнего уровня, используется для исследований и приложений в области поиска, лингвистики и рассуждений. Назначение SUMO — содействовать улучшению интероперабельности данных, извлечению и поиску информации, автоматическому выводу (доказательствам), обработке естественного языка.

Структура SUMO основана на разделении сущностей на две категории:
- абстрактные сущности;
- физические сущности.

Иерархия понятий, используемых в SUMO, представлена на рисунке *Рисунок. Иерархия понятий, используемых в SUMO*.

Рисунок. Иерархия понятий, используемых в SUMO
=

Первая категория “Абстрактное” включает в себя все, что имеет положение в пространстве и времени, а вторая категория включает в себя все остальное.

В свою очередь онтология DOLCE ориентирована на то, чтобы охватить онтологические категории, лежащие в основе естественного языка.

Иерархия понятий, используемых в DOLCE, представлена на рисунке *Рисунок. Иерархия понятий, используемых в DOLCE*.

Рисунок. Иерархия понятий, используемых в DOLCE
=

Верхний объект помечен как “Сущность”, что указывает на то, что все экземпляры этого объекта и его подтипов являются частными по отношению к понятию “Сущность”. У рассмотренной онтологии несомненно есть свои плюсы, такие как разделение объектов на классы во времени, однако деление классов не шаблонизировано.

Стоит отметить, что представленный список не является окончательным. Существует более пятнадцати онтологий, претендующих на звание *онтологий верхнего уровня*, целью создания которых является использование их при создании онтологий более низкого уровня.

Однако попытки создать универсальную *онтологию верхнего уровня*, способную обеспечить совместимость *интеллектуальных компьютерных систем*, не привели к ожидаемым результатам, поскольку данные *онтологии* имеют ряд ключевых недостатков:

- Каждая из представленных *онтологий* является монолитной структурой, в которой нет четкой локализации на отдельные небольшие *онтологии*.
  Главной задачей при проектировании фрагментов *баз знаний* с использованием онтологического подхода является выделение *онтологий* таким образом, чтобы они давали возможность относительно независимой эволюции каждого фрагмента. Структура данных *онтологий верхнего уровня* представляет собой иерархию, состоящую из большого количества различных понятий. Данный вид структуризации приводит к ситуации, когда необходимость внесения изменений в одном месте обязательно повлечет за собой невозможность редактирования другой части *онтологии*. В силу вышесказанного данный вид структуризации делает *онтологии* неудобными для их использования при разработке различных интеллектуальных систем;
- Рассматриваемые *онтологии верхнего уровня* не являются частью комплексной технологии.
  Поскольку рассматриваемые *онтологии* не являются частью какой-то комплексной технологии, они не могут рассматриваться как часть *библиотеки многократно используемых компонентов*. Что, в свою очередь, приводит к неудобствам в виде необходимости адаптации используемых *онтологий* для каждой конкретной системы.
- Не существует технологий проектирования *баз знаний* на основе данных *онтологий верхнего уровня*.
  Отсутствие технологий проектирования *баз знаний* затрудняет разработку интеллектуальных систем.

В свою очередь, *Технология OSTIS* позволяет построить четкую иерархию *предметных областей и онтологий*, что подразумевает легкость расширения *баз знаний* и, соответственно, легкость расширения онтологий и предметных областей. Данный факт позволяет обеспечить возможность коллективной разработки *баз знаний*, построенных на основе *Технологии OSTIS*, а также позволяет использовать результаты, полученные в ходе работы над описанными онтологиями верхнего уровня, в ostis-системах, и собрать все лучшее от существующих ныне решений, исключая их недостатки.

Заключение к Главе 2.5.

Онтологический подход к проектированию *баз знаний интеллектуальных компьютерных систем* является ключевым в технологии проектирования *баз знаний интеллектуальных компьютерных систем* и заключается в представлении *базы знаний интеллектуальных компьютерных систем* на основе *Технологии OSTIS* как иерархической структуры взаимосвязанных *предметных областей* и их *онтологий*, построенных на базе *онтологий верхнего уровня*.

Использование данного подхода при проектировании *баз знаний интеллектуальных компьютерных систем* позволяет обеспечить совместимость интеллектуальных систем, за счет унифицированного представления *знаний*. А также повысить эффективность разработки *интеллектуальных компьютерных систем нового поколения*, за счет компонентного подхода к разработке *баз знаний* и средств автоматизации их разработки.

Глава 2.6.

Смысловое представление логических формул и высказываний в различного вида логиках

⇒ *автор**:
- *Шункевич Д. В.*
- *Василевская А. П.*
- *Орлов М. К.*
- *Ивашенко В. П.*

⇒ *аннотация**:
[В главе рассмотрено представление логических формул и высказываний как для классических логик и их приложений (прикладных логик), так и представление некоторых понятий неклассических логик.]

⇒ *подраздел**:
- *§ 2.6.1. Смысловое представление логических формул и формальных теорий классической логики*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

⇒ *ключевой знак**:
- *Отношение выводимости*
- *Отношение выводимости на конечных множествах*
- *Отношение выводимости на конечных множествах полносвязно представленных множеств*
- *Отношение выводимости на секвенциях*

⇒ *ключевое понятие**:
- *логическая формула*
- *высказывание*
- *фактографическое высказывание*
- *нефактографическое высказывание*
- *атомарное существование*
- *слотовое бинарное отношение*
- *неслотовое бинарное отношение*
- *секвенция*
- *метаструктура*
- *модальный оператор*
- *модальное правило вывода*
- *отношение становления структур*
- *последовательность мышления*
- *немонотонный вывод на конечном sc-множестве посылок*
- *выводимое множество*

⇒ *ключевое отношение**:
- *высказывание**
- *подформула**
- *логическая связка**
- *квантор**
- *нечеткая истинность**
- *верное высказывание**
- *конструктивно истинное высказывание**
- *монотонное бинарное отношение**
- *неискаженное высказывание**

⇒ *ключевое знание**:
- *Описание понятия формальной теории*
- *Описание понятия утверждения*
- *Описание понятия определение*

⇒ *библиографическая ссылка**:
- *Клини С.К.МатемЛ-1973кн*
- *Тейз А..ЛогичПкИИоК-1990кн*
- *Тейз А..ЛогичПкИИоМ-1998кн*

- *Драгалин А.Г.КонстТДиНА-2003кн*
- *Вагин В.Н..Досто иПВвИС-2008кн*
- *Голенков В.В..ИнтелОСиВУ-2001кн*
- *Батыршин И.З..НечетГСТиП-2007кн*
- *Behounek L.Intro tMFL-2011bk*
- *Ивашенко В.П.Модел иАИЗнООСС-2014дс*
- *Ивашенко В.П.ОнтолМПВОСиЯвПОЗ-2017ст*
- *Летичевский А.А..ИнсерП-2003ст*

§ 2.6.1. Смысловое представление логических формул и формальных теорий классической логики

Появление *формальных систем* было обусловлено осознанием того факта, что совершенно различные системы, будь то технические, социальные, экономические или биологические, обладают глубоким сходством. Действительно, каждая конкретная система состоит из каких-то первичных (базовых) элементов, обладающих какими-то свойствами. Затем, исходя из наличия исходных описаний, можно логическим путем вывести описание новых свойств, причем утверждения о наличии исходных или выведенных свойств воспринимаются как истинные на основании смысла определений данных элементов.

формальная теория — это *множество высказываний*, которые считаются истинными в рамках данной ***формальной теории***.

Аксиоматические системы — это системы с наличием определенного числа исходных заранее выбранных и фиксированных высказываний, называемых аксиомами.

высказывания могут быть как фактографическими, так и нефактографическими. Некоторые *высказывания* считаются аксиомами, а другие доказываются на основе других высказываний в рамках этой же ***формальной теории***.

Каждая формальная теория интерпретируется (то есть ее *высказывания* являются истинными) на какой-либо *предметной области*, которая является максимальным из *фактографических высказываний* (их *объединением**), входящих в состав этой ***формальной теории***.

Каждой ***формальной теории*** соответствует одна *предметная область*, которая входит в нее под атрибутом *предметная область'*.

Каждая *формальная теория* может рассматриваться как *конъюнктивное высказывание*, априори истинное (с чьей-то точки зрения) при интерпретации на соответствующей предметной области.

Каждая *формальная теория* задается *алфавитом*, *формулами*, *аксиомами*, *правилами вывода*.

предметная область является *максимальным фактографическим высказыванием формальной теории*, которая интерпретируется на данной *предметной области*.

аксиома — это *высказывание*, истинность которого не требует доказательства в рамках рассматриваемой *формальной теории*.

теорема — это *высказывание*, истинность которого доказывается в рамках рассматриваемой *формальной теории*.

логическая формула
⇒ *разбиение**:
{• *атомарная логическая формула*
• *неатомарная логическая формула*
}
⇒ *разбиение**:
{• *замкнутая логическая формула*
• *открытая логическая формула*
}

высказывание
⊂ *логическая формула*
⇒ *разбиение**:
{• *атомарное высказывание*
• *неатомарное высказывание*
}

⇒ *разбиение**:
{• *фактографическое высказывание*
⊂ *замкнутая логическая формула*
• *нефактографическое высказывание*
}

Под ***высказыванием*** понимается некоторая *структура* (в которую входят *sc-константы* из некоторой предметной области и/или *sc-переменные*) или *логическая связка*, которая может трактоваться как истинная или ложная в рамках какой-либо *предметной области*.

Истинность ***высказывания*** задается путем указания принадлежности знака этого высказывания *формальной теории*, соответствующей данной *предметной области*. Ложность высказывания задается путем указания принадлежности знака *отрицания** этого высказывания данной *формальной теории*.

Явно указанная непринадлежность ***высказывания*** *формальной теории* может говорить как о его ложности в рамках данной теории (если это указано рассмотренным выше образом), так и о том, что данное ***высказывание*** вообще не рассматривается в данной *формальной теории* (например, использует понятия, не принадлежащие данной *предметной области*).

Одно и то же ***высказывание*** может быть истинно в рамках одной *формальной теории* и ложно в рамках другой.

атомарное высказывание
⊂ *структура*
⇒ *разбиение**:
{• *атомарное фактографическое высказывание*
• *атомарное нефактографическое высказывание*
}

атомарное высказывание — это *высказывание*, которое не является *неатомарным высказыванием*.

неатомарное высказывание — это *высказывание*, в состав которого входят только знаки *логических формул* или множества связываемых переменных. Следует отметить, что невозможно говорить об истинности либо ложности ***неатомарного высказывания*** в рамках какой-либо *формальной теории*, в случае, когда невозможно установить истинность либо ложность любого из его элементов в рамках этой же *формальной теории* или интерпретации этих элементов.

Под *фактографическим высказыванием* понимается:
- *атомарное высказывание*, в состав которого не входит ни одна *sc-переменная*;
- *неатомарное высказывание*, все элементы которого также являются ***фактографическими высказываниями***.

высказывание*
:= [бинарное ориентированное отношение, каждая *пара* которого связывает (1) знак некоторой *предметной области* и (2) знак некоторого *высказывания*.]
⇒ *разбиение**:
{• *ложное высказывание**
• *неразрешимое высказывание**
• *истинное высказывание**
}

⇒ *предъявляемое требование**:
[Все *sc-константы*, входящие в состав всех *атомарных логических формул*, входящих в состав всех *высказываний*, описывающих некоторую *предметную область* должны входить в состав описываемой *предметной области*.]

следует отличать*
∋ {• *высказывание**
∈ *бинарное ориентированное отношение*
:= [быть высказыванием, описывающим заданную предметную область*]
⇒ *сокращение**:
[быть высказыванием*]
• *высказывание*
⊂ *логическая формула*
:= [Второй домен отношения “быть высказыванием”]
}

нефактографическое высказывание
⇒ *разбиение**:
{• *атомарное нефактографическое высказывание*
• *неатомарное нефактографическое высказывание*
}

Под *нефактографическим высказыванием* понимается:
- *атомарное нефактографическое высказывание*, в состав которого входит хотя бы одна *sc-переменная*;
- *неатомарное нефактографическое высказывание*, хотя бы один элемент которого является ***нефактографическим высказыванием***.

Под ***атомарным нефактографическим высказыванием*** понимается *атомарное высказывание*, которое является *нефактографическим высказыванием*.

атомарное нефактографическое высказывание — это *нефактографическое высказывание*, которая не является *неатомарным нефактографическим высказыванием*.

атомарная логическая формула — это *логическая формула*, которая не является *неатомарной логической формулой*.

По умолчанию ***атомарное нефактографическое высказывание*** трактуется как *высказывание* о существовании, то есть наличия в памяти значений, соответствующих всем *sc-переменным*, входящим в состав данной формулы и не попадающих под действие какого-либо другого *квантора* (указанного явно или по умолчанию). Таким образом, на все *sc-переменные*, входящие в состав *атомарного нефактографического высказывания* и не попадающие под действие какого-либо другого *квантора*, неявно накладывается квантор *существования**.

логическая формула
⇒ *разбиение**:
{• *выполнимая логическая формула*
⇒ *разбиение**:
{• *общезначимая логическая формула*
• *нейтральная логическая формула*
}
• *невыполнимая логическая формула*
}

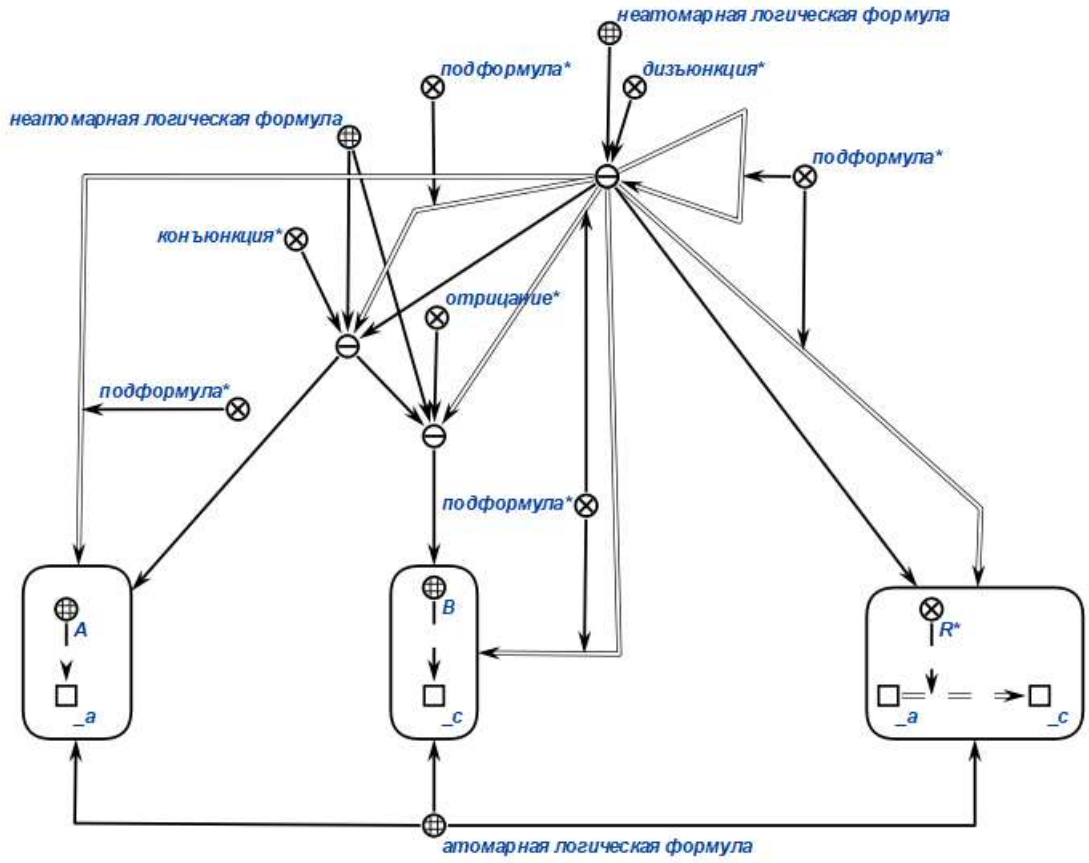
Под ***неатомарным нефактографическим высказыванием*** понимается *неатомарное высказывание*, которое является *нефактографическим высказыванием*.

Для того, чтобы рассмотреть типологию ***неатомарных логических формул***, будем говорить, что исследуется истинность самой ***неатомарной логической формулы*** и всех ее *подформул** в рамках одной и той же *формальной теории*, при этом не важно, какой именно. Также считается, что в рассматриваемой *формальной теории* каждая *подформула** рассматриваемой ***неатомарной логической формулы*** в рамках этой *формальной теории* может однозначно трактоваться как либо истинная, либо ложная. В противном случае мы не можем говорить об истинности либо ложности исходной ***неатомарной логической формулы*** в рамках этой *формальной теории*.

Будем называть ***подформулой**** *неатомарной логической формулы* ***fi*** любую *логическую формулу* ***fj***, являющуюся элементом исходной формулы ***fi***, а также любую ***подформулу**** формулы ***fj***.

подформула*
:= [частная формула*]
∈ *бинарное отношение*
∈ *ориентированное отношение*
∈ *транзитивное отношение*

SCg-текст. Формализация примера подформулы
=



утверждение — это *семантическая окрестность* некоторой *логической формулы*, в которую входит полный текст этой *логической формулы*, а также факт принадлежности этой *логической формулы* некоторой *формальной теории*.

Знак *логической формулы*, семантическая окрестность которой представляет собой утверждение, является *главным ключевым sc-элементом'* в рамках этого ***утверждения***. Знаки понятий соответствующей *предметной области*, которые входят в состав какой-либо *подформулы** указанной *логической формулы*, будут *ключевыми sc-элементами'* в рамках этого ***утверждения***.

Полный текст некоторой *логической формулы* включает в себя:
- знак самой этой *логической формулы*;
- знаки всех ее *подформул**;
- элементы всех *логических формул*, знаки которых попали в данную структуру;
- все пары принадлежности, связывающие *логические формулы*, знаки которых попали в данную структуру, с их компонентами.

Таким образом, факт принадлежности (истинности) логической формулы нескольким *формальным теориям* будет порождать новое утверждение для каждой такой *формальной теории*. Текст ***утверждения*** входит в состав *логической онтологии*, соответствующей *предметной области*, на которой интерпретируется *главный ключевой sc-элемент'* данного утверждения.

Правило идентификации экземпляров ***утверждения*** в рамках *Русского языка* именуются по следующим правилам:
- в начале идентификатора пишется сокращение **Утв.**;
- далее в круглых скобках через точку с запятой перечисляются основные идентификаторы *ключевых sc-элементов'* данного ***утверждения***. Порядок определяется в каждом конкретном случае в зависимости от того, свойства каких из этих *понятий* описывает данное ***утверждение*** в большей или меньшей степени.

Могут быть исключения для ***утверждений***, названия которых закрепились исторически, например, *Теорема Пифагора*, *Аксиома о прямой и точке*.

определение — это *утверждение*, *главным ключевым sc-элементом'* которого является связка *эквиваленции**, однозначно определяющая некоторое понятие на основе других понятий.

Каждое определение имеет ровно один *ключевой sc-элемент'* (не считая *главного ключевого sc-элемента'*).

Для одного и того же понятия в рамках одной *формальной теории* может существовать несколько *утверждений об эквиваленции**, однозначно задающих некоторое понятие на основе других, однако только одно такое *утверждение* в рамках этой *формальной теории* может быть отмечено как ***определение***. Остальные *утверждения об эквиваленции** могут трактоваться как *пояснения* данного понятия.

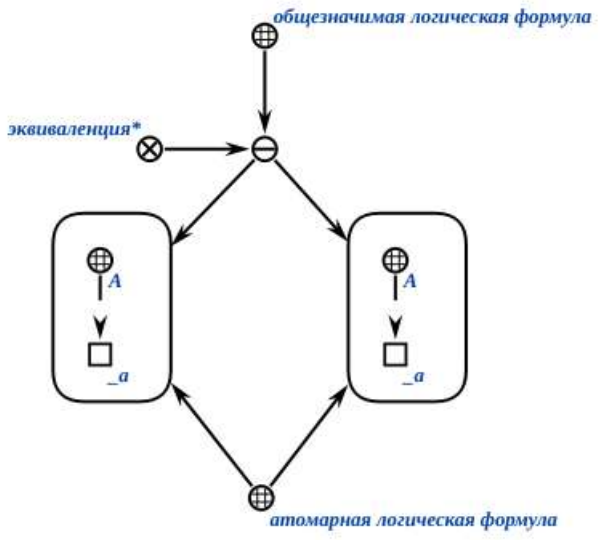
Правило идентификации экземпляров ***определения*** в рамках *Русского языка* именуются по следующим правилам:

- в начале идентификатора пишется сокращение **Опр.**;
- далее в круглых скобках через точку с запятой записывается основной идентификатор *ключевого sc-элемента'* данного ***определения***.

общезначимая логическая формула
:= [тождественно истинная логическая формула]
⊂ *выполнимая логическая формула*
⊂ *логическая формула, равнозначная логической константе*

общезначимая логическая формула — это *логическая формула*, для которой не существует *формальной теории*, в рамках которой она была бы ложной (или имела бы ложную интерпретацию) с учетом истинности и ложности всех ее *подформул** (или их интерпретаций) в рамках этой же *формальной теории*.

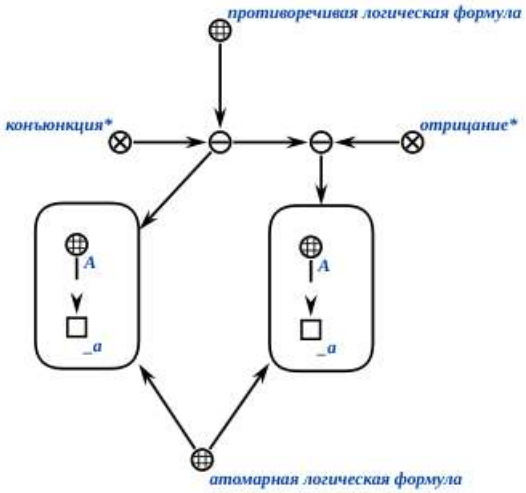
SCg-текст. Формализация закона тождества
=



противоречивая логическая формула
:= [тождественно ложная логическая формула]
⊂ *невыполнимая логическая формула*
⊂ *логическая формула, равнозначная логической константе*

противоречивая логическая формула — это *логическая формула*, для которой не существует *формальной теории*, в рамках которой она была бы истинной (или имела бы истинную интерпретацию) с учетом истинности и ложности всех ее *подформул** (или их интерпретаций) в рамках этой же *формальной теории*.

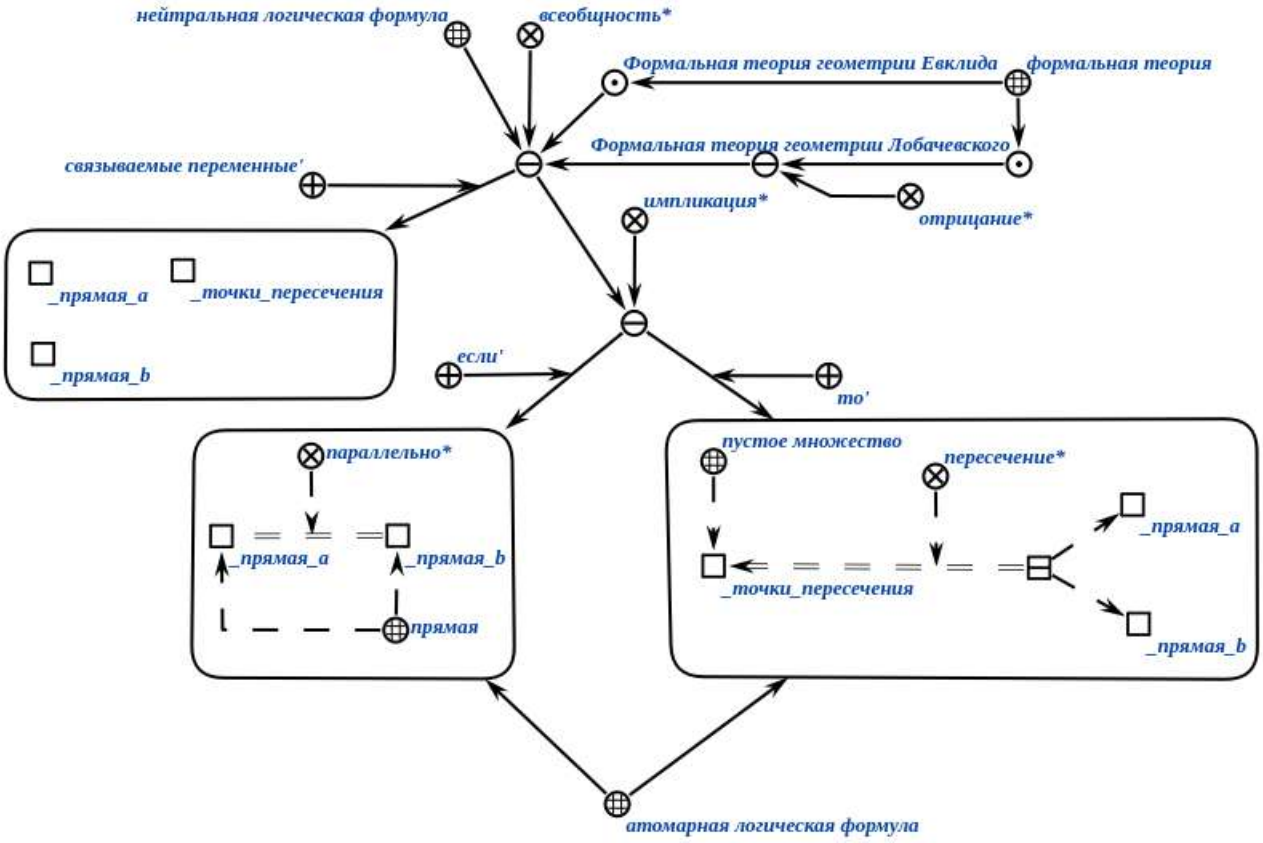
SCg-текст. Формализация закона противоречия
=



нейтральная логическая формула
⊂ *выполнимая логическая формула*

нейтральная логическая формула — это *логическая формула*, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула ложна (или имеет ложную интерпретацию), и хотя бы одна *формальная теория*, в рамках которой эта формула истинна (или имеет ложную интерпретацию).

SCg-текст. Формализация нейтральной логической формулы
=



В *Геометрии Евклида* в плоскости через точку, не лежащую на данной прямой, можно провести одну и только одну прямую, параллельную данной. В *Геометрии Лобачевского* данный постулат является ложным. В *Сферической геометрии* все прямые пересекаются.

непротиворечивая логическая формула
:= [выполнимая логическая формула]
⇐ *объединение**:
{• *нейтральная логическая формула*
• *общезначимая логическая формула*
}

непротиворечивая логическая формула — это *логическая формула*, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула истинна (или имеет истинную интерпретацию).

необщезначимая логическая формула
⇐ *объединение**:
{• *нейтральная логическая формула*
• *противоречивая логическая формула*
}

необщезначимая логическая формула — это *логическая формула*, для которой существует хотя бы одна *формальная теория*, в рамках которой эта формула ложна (или имеет ложную интерпретацию).

логическая формула, равнозначная логической константе — это *логическая формула*, которая является либо только истинной (имеет только истинные интерпретации), либо только ложной (имеет только ложные интерпретации) в рамках всех *формальных теорий*, в которых можно установить ее истинность или ложность. ***логическая формула, равнозначная логической константе*** — это такая *логическая формула*, которая является либо *общезначимой логической формулой*, либо *противоречивой логической формулой*.

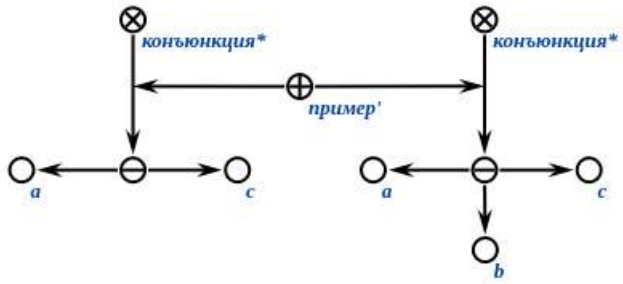
логическая связка*
:= [неатомарная логическая формула]
:= [логический оператор*]
:= [пропозициональная связка*]
∈ *класс связок разной мощности*
⇐ *семейство подмножеств**:
неатомарное высказывание

логическая связка* — это отношение (класс связок), связками которого являются *высказывания*, а областью определения которого является множество *высказываний*, при этом само это отношение и некоторые его подмножества могут быть *классами связок разной мощности*.

конъюнкция*
:= [логическое и*]
:= [логическое умножение*]
⊂ *логическая связка**
∈ *неориентированное отношение*
∈ *класс связок разной мощности*
∈ *неунарное отношение*
⇒ *область определения**:
логическая формула

конъюнкция* — это множество конъюнктивных *логических формул*, каждая из которых истинна (имеет истинные интерпретации) в рамках некоторой *формальной теории* только в том случае, когда все ее компоненты истинны (имеют только соответствующие истинные интерпретации) в рамках этой же *формальной теории*.

SCg-текст. Формализация примера конъюнкции
=

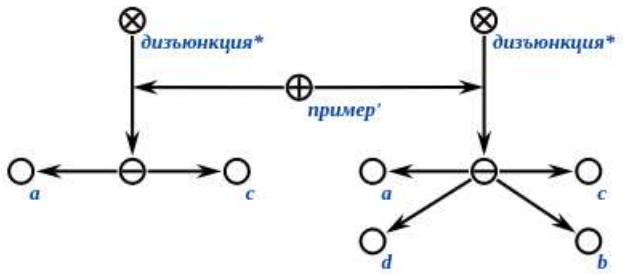


дизъюнкция*
:= [логическое или*]
:= [логическое сложение*]
:= [включающее или*]
⊂ *логическая связка**
∈ *неориентированное отношение*
∈ *класс связок разной мощности*
∈ *неунарное отношение*
⇒ *область определения**:
логическая формула

дизъюнкция* — это множество дизъюнктивных *логических формул*, каждая из которых истинна (имеет истинные интерпретации) в рамках некоторой *формальной теории* только в том случае, когда хотя бы один его компонент является истинным (имеет соответствующую истинную интерпретацию) в рамках этой же *формальной теории*.

Следует отметить, что каждая конъюнктивная и дизъюнктивная формула представляют собой связку (множество) своих непосредственных подформул, причем эти множества могут быть равными, однако, такие множества будем полагать различными, так как одно множество будет представлять конъюнкцию, а другое – дизъюнкцию. Наличие равных, но различных множеств не допускается в классической математике, которая основывается, в том числе, на абстракции обобщения (все равные множества (абстракции) – тождественны). Однако, такие множества могут существовать в неклассических математических моделях. Таким образом, в рамках языка SC будем использовать неклассическую математическую модель для представления логических формул классической математической логики.

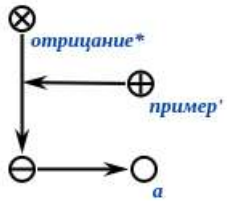
SCg-текст. Формализация примера дизъюнкции
=



отрицание*
⊂ *логическая связка**
⊂ *синглетон*
∈ *унарное отношение*
⇒ *область определения**:
логическая формула

отрицание* — это множество *логических формул* об отрицании, каждое из которых истинно (имеет истинную интерпретацию) в рамках некоторой *формальной теории* только в том случае, когда его единственный элемент является ложным (имеет ложную эту же интерпретацию) в рамках этой же *формальной теории*.

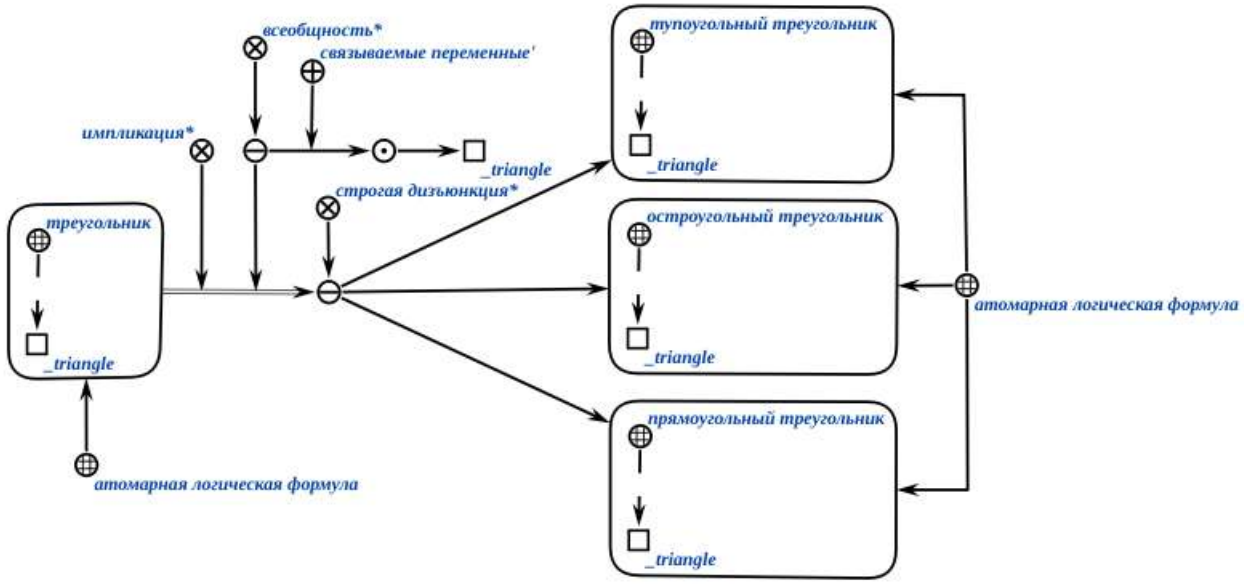
SCg-текст. Формализация примера отрицания
=



строгая дизъюнкция*
:= [исключающее или*]
:= [альтернатива*]
$\subset$ *логическая связка**
$\in$ *неориентированное отношение*
$\in$ *класс связок разной мощности*

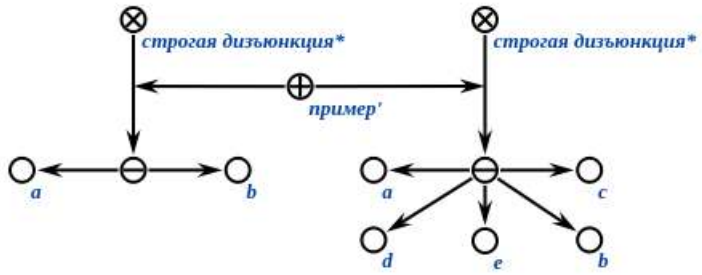
строгая дизъюнкция* — это множество строго дизъюнктивных *логических формул*, каждое из которых истинно в рамках некоторой *формальной теории* только в том случае, когда ровно один его компонент является истинным (имеет соответствующую истинную интерпретацию) в рамках этой же *формальной теории*.

SCg-текст. Формализация примера строгой дизъюнкции в геометрии
=



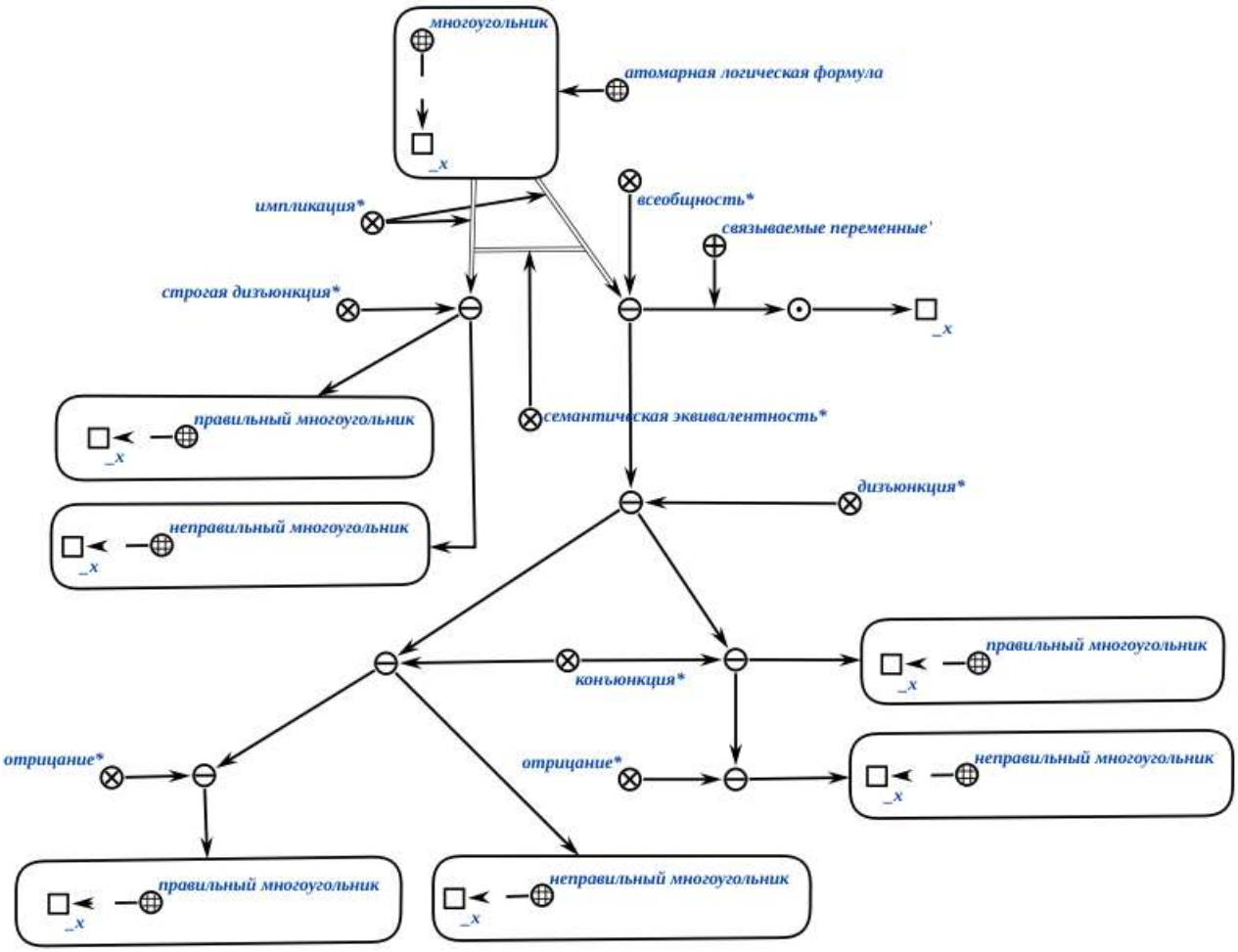
$\Rightarrow$ *пояснение**:
[Данная неатомарная логическая формула содержит следующую информацию: для любых переменных _triangle если _triangle является треугольником, то _triangle является или тупоугольным треугольником, или остроугольным треугольником, или прямоугольным треугольником.**]**

SCg-текст. Формализация примера строгой дизъюнкции
=



строгая дизъюнкция* может быть представлена как *дизъюнкция конъюнкции отрицания* первой логической формулы и второй логической формулы и *конъюнкции* первой логической формулы и *отрицания* второй логической формулы. Также она может быть представлена и в виде *конъюнкции дизъюнкций* двух логических формул и их *отрицаний*.

SCg-текст. Формализация примера строгой дизъюнкции
=



импликация*
:= [логическое следование*]
⊂ *логическая связка**
∈ *бинарное отношение*
∈ *ориентированное отношение*
⇒ *область определения**:

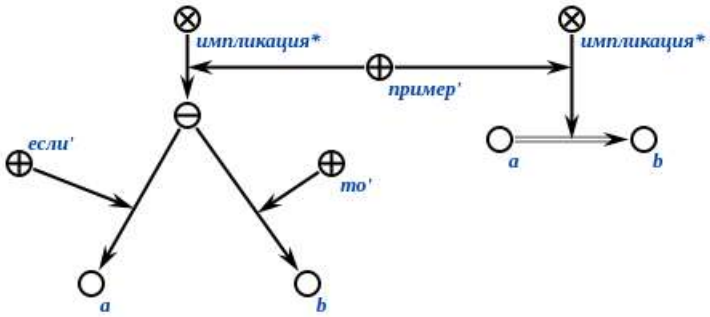
логическая формула

импликация* — это множество импликативных *логических формул*, каждая из которых состоит из посылки (первый компонент *высказывания*) и следствия (второй компонент *высказывания*).

Каждое импликативное *высказывание* ложно в рамках некоторой *формальной теории* в том случае, когда его посылка истинна, а заключение ложно в рамках этой же *формальной теории*. В других случаях такое *высказывание* истинно.

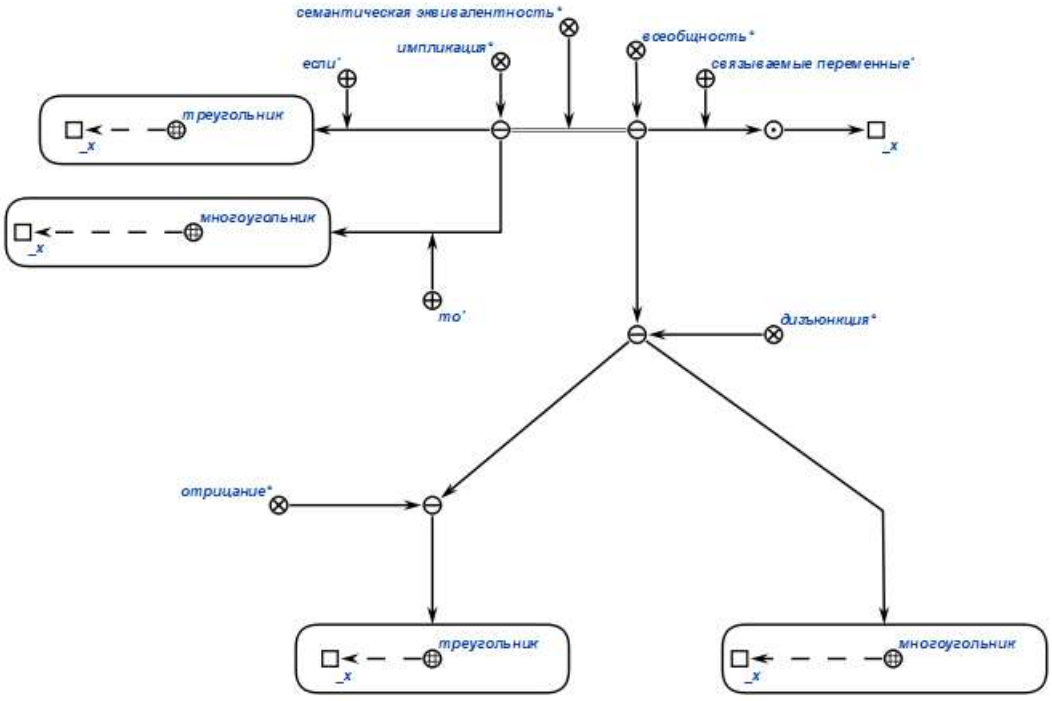
По умолчанию на все переменные, входящие в обе части высказывания об ***импликации**** (или хотя бы одну из *подформул** каждой части) неявно накладывается квантор *всеобщности**, при условии, что эти переменные не связаны другим *квантором*, указанным явно.

SCg-текст. Формализация примера импликации
=

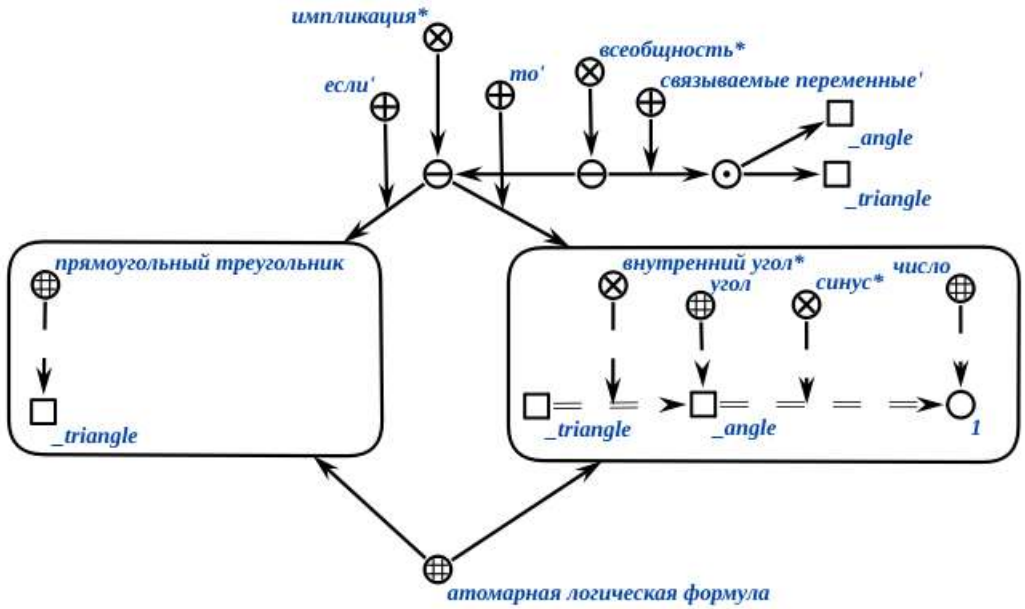


импликация* может быть представлена как *дизъюнкция отрицания* первой логической формулы и второй логической формулы или же как *отрицание конъюнкции* первой логической формулы и *отрицания* второй логической формулы.

SCg-текст. Формализация примера импликации
=



SCg-текст. Формализация примера импликации
=



⇒ *пояснение**:
[Данная неатомарная логическая формула содержит следующую информацию: для любых переменных _triangle и _angle если _triangle является прямоугольным треугольником, то синус его внутреннего угла _angle равен единице.**]**

если'
:= **[**посылка'**]**
⊂ *1'*
∈ *ролевое отношение*

если' — это *ролевое отношение*, используемое в связках *импликации** для указания посылки.

то'
:= **[**следствие'**]**
⊂ *2'*
∈ *ролевое отношение*

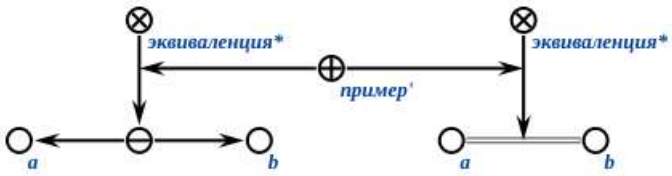
то' — это *ролевое отношение*, используемое в связках *импликации** для указания следствия.

эквиваленция*
:= **[**эквивалентность***]**
⊂ *логическая связка**
∈ *бинарное отношение*
∈ *неориентированное отношение*
⇒ *область определения**:
логическая формула

эквиваленция* — это множество *логических формул* об эквивалентности, каждое из которых истинно (имеет истинную интерпретацию) в рамках некоторой *формальной теории* только в тех случаях, когда оба его компонента одновременно либо истинны (имеют соответствующие истинные интерпретации) в рамках этой же *формальной теории*, либо ложны (имеют соответствующие ложные интерпретации).

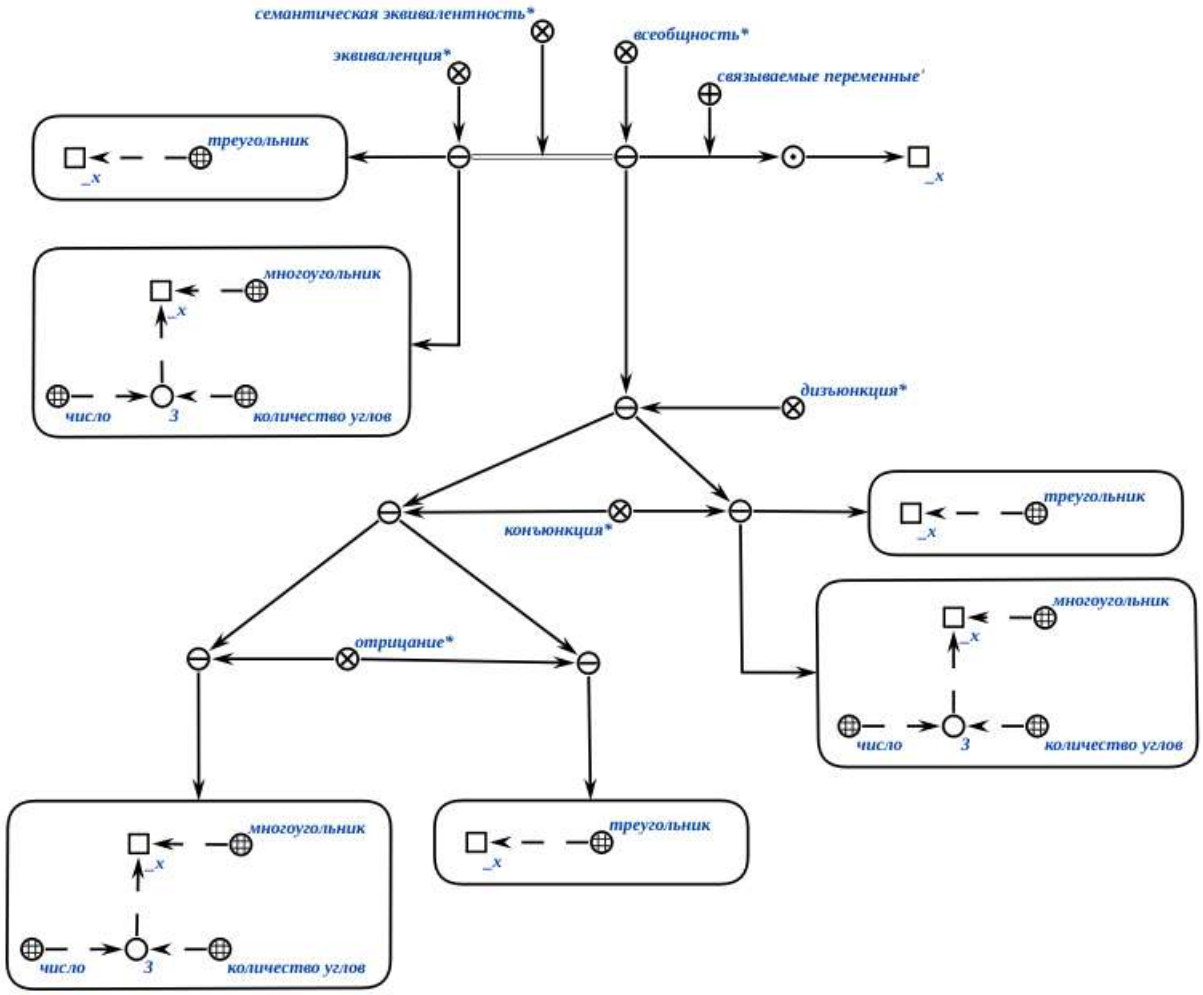
По умолчанию на все переменные, входящие в обе части высказывания об ***эквиваленции**** (или хотя бы одну из *подформул** каждой части) неявно накладывается квантор *всеобщности**, при условии, что эти переменные не связаны другим *квантором*, указанным явно.

SCg-текст. Формализация примера эквиваленции
=



эквиваленция* двух логических формул может быть представлена как *дизъюнкция конъюнкции* этих двух логических формул и *конъюнкции отрицаний* этих двух логических формул.

SCg-текст. Формализация примера эквиваленции
=



квантор
⊂ *логическая связка**

квантор — это *отношение*, каждая связка которого истинна (или имеет истинную интерпретацию) при выполнении дополнительных условий, связанных с некоторыми из переменных, входящих в состав *логических формул*, входящих в ее состав.

Будем говорить, что переменные связаны ***квантором*** или попадают под область действия данного ***квантора*** (имея в виду конкретную связку конкретного ***квантора***).

В состав каждой связки каждого ***квантора*** входит *атомарная формула*, являющаяся *тривиальной структурой*, в которой перечислены переменные, связанные данным ***квантором***.

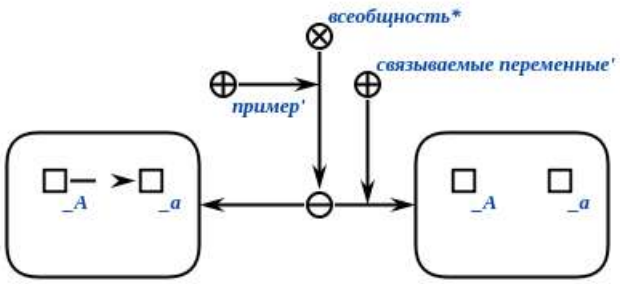
всеобщность*
:= [квантор всеобщности*]
:= [квантор общности*]
∈ *квантор*
∈ *ориентированное отношение*
∈ *класс связок разной мощности*

всеобщность* — это *квантор*, для каждой связки которого, истинной в рамках некоторой *формальной теории* (или имеющей истинную интерпретацию), выполняется следующее утверждение: все формулы, входящие в состав этой связки имеют соответствующую истинную интерпретацию в рамках этой же *формальной теории* при всех (любых) возможных значениях всех элементов множества *связываемых переменных'* входящего в эту связку.

Каждая связка *квантора* ***всеобщность**** может быть представлена как *конъюнкция** (потенциально бесконечная) исходных *логических формул*, входящих в состав этой связки, в каждой из которых все *связанные переменные'* заменены на их возможные значения.

Квантор ***всеобщности**** зачастую обозначается "∀" и читается как "для всех", "для каждого", "для любого" или "все", "каждый", "любой".

SCg-текст. Формализация примера всеобщности
=



формула существования
:= [существование*]
⇒ *разбиение**:
{• *атомарное существование*
• *неатомарное существование**
}

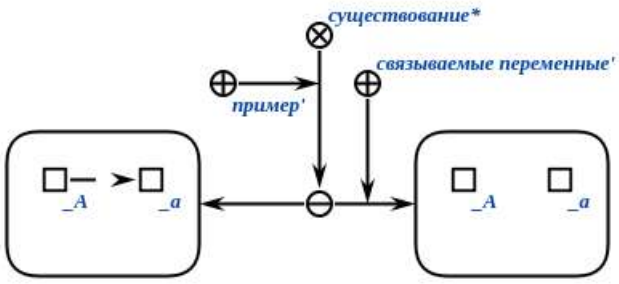
неатомарное существование*
:= [квантор неатомарного существования*]
∈ *квантор*
∈ *ориентированное отношение*
∈ *класс связок разной мощности*

неатомарное существование* — это *квантор*, для каждой связки которого, истинной в рамках некоторой *формальной теории* (или имеющей истинную интерпретацию), выполняется следующее утверждение: существуют значения всех элементов множества *связываемых переменных'* входящего в эту связку, такие, что все формулы, входящие в состав этой связки имеют соответствующую истинную интерпретацию в рамках этой же *формальной теории*.

Каждая связка *квантора* ***неатомарное существование**** может быть представлена как *дизъюнкция** (потенциально бесконечная) исходных *логических формул*, входящих в состав этой связки, в каждой из которых все *связанные переменные'* заменены на их возможные значения.

Квантор ***существования**** зачастую обозначается "∃" и читается как "существует", "для некоторого", "найдется".

SCg-текст. Формализация примера неатомарного существования
=



число значений переменной
∈ *параметр*

Каждый элемент *параметра* ***число значений переменной*** представляет собой класс ориентированных пар, первым компонентом которых является знак *логической формулы*, вторым — *sc-переменная*, имеющая в рамках данной *логической формулы* ограниченное известное число значений, при которых данная формула является истинной в рамках соответствующей *формальной теории*.

Отметим, что в случае *атомарной логической формулы* каждая такая связка связывает знак формулы и знак принадлежащей ей *sc-переменной*, то есть является, по сути, частным случаем пары принадлежности. В случае *неатомарной логической формулы* указанная *sc-переменная* может принадлежать любой из *подформул** исходной формулы.

*измерением** значения параметра ***число значений переменной*** является некоторое *число*, задающее количество значений *sc-переменных* в рамках *логической формулы*.

кратность существования
∈ *параметр*
⇒ *область определения параметра**:
формула существования
∋ *единственное существование*

Каждый элемент *параметра* ***кратность существования*** представляет собой класс логических *формул существования*, для которых при интерпретации на соответствующей *предметной области* существует ограниченное общее для всех таких формул число комбинаций значений переменных, при которых указанные формулы являются истинными в рамках соответствующей *формальной теории*. *измерением** каждого значения ***кратности существования*** является некоторое *число*, задающее количество таких комбинаций.

единственное существование
:= [однократное существование]
:= [формула существования и единственности]

Единственное существование зачастую обозначается “∃!” и читается как “существует и единственный”.

логическая формула и единственность
⊂ *логическая формула*
⊂ *единственное существование*

Каждый элемент множества ***логическая формула и единственность*** представляет собой *логическую формулу* (*атомарную* или *неатомарную*), для которой дополнительно уточняется, что при ее интерпретации на некоторой предметной области существует только один набор значений переменных, входящих в эту формулу (или ее *подформулы**), при котором указанная логическая формула истинна в рамках *формальной теории*, в которую входит данная *предметная область*.

связываемые переменные' — это *ролевое отношение*, которое связывает связку конкретного *квантора* с множеством переменных, которые связаны этим квантором.

открытая логическая формула — это *логическая формула*, в рамках которой (и всех ее *подформул**) существует хотя бы одна переменная, не связанная никаким *квантором*.

замкнутая логическая формула — это *логическая формула*, в рамках которой (и всех ее *подформул**) не существует переменных, не связанных каким-либо *квантором*.

§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках

⇒ *ключевой знак**:
- *Отношение выводимости*
- *Отношение выводимости на конечных множествах*
- *Отношение выводимости на конечных множествах полносвязно представленных множеств*
- *Отношение выводимости на секвенциях*

⇒ *ключевое понятие**:
- *иррефлексивное слотовое бинарное отношение*
- *иррефлексивное неслотовое бинарное отношение*
- *рефлексивное слотовое бинарное отношение*
- *рефлексивное неслотовое бинарное отношение*
- *транзитивное слотовое бинарное отношение*
- *транзитивное неслотовое бинарное отношение*
- *симметричное слотовое бинарное отношение*
- *симметричное неслотовое бинарное отношение*
- *антисимметричное слотовое бинарное отношение*
- *антисимметричное неслотовое бинарное отношение*
- *монотонное бинарное отношение**
- *отношение порядка монотонного отношения'*
- *монотонное бинарное отношение'*
- *монотонное слотовое бинарное отношение**
- *монотонное неслотовое бинарное отношение**
- *слотовое бинарное отношение*
- *неслотовое бинарное отношение*
- *слотовое отношение эквивалентности*
- *неслотовое отношение эквивалентности*
- *слотовое отношение нестрогого порядка*
- *неслотовое отношение нестрогого порядка*
- *секвенция*
- *метаструктура*
- *модальный оператор*
- *модальное правило вывода*
- *отношение становления структур*
- *последовательность мышления*
- *последовательность рационального мышления*
- *последовательность иррационального мышления*
- *последовательность рационального мышления классической логики*
- *последовательность рационального дедуктивного мышления классической логики*
- *последовательность рационального дедуктивного мышления классической логики на конечных sc-множествах*
- *последовательность классического рационального дедуктивного познания*

Прикладные логики (см. *Клини С.К.МатемЛ-1973кн*, *Тейз А..ЛогичПкИИоК-1990кн*, *Драгалин А.Г.КонстТДиНА-2003кн*, *Вагин В.Н..Досто иПВвИС-2008кн*, *Тейз А..ЛогичПкИИоМ-1998кн*, *Голенков В.В..ИнтелОСиВУ-2001кн*) рассматривают приложения классической логики к абстрактным и предметным областям, описывающим действительность: логические теории об отношениях равенства и порядка (см. *Клини С.К.МатемЛ-1973кн* , *Тейз А..ЛогичПкИИоК-1990кн*), логические теории арифметики (см. *Клини С.К.МатемЛ-1973кн*), логические теории времени (см. *Тейз А..ЛогичПкИИоМ-1998кн*, *Вагин В.Н..Досто иПВвИС-2008кн*), логические теории доказательств (см. *Драгалин А.Г.КонстТДиНА-2003кн*, *Клини С.К.МатемЛ-1973кн*), теории графов и геометрические теории (см. *Голенков В.В..ИнтелОСиВУ-2001кн*), теории природных и социальных систем (см. *Тейз А..ЛогичПкИИоК-1990кн*).

Классификация логических теорий соответствует классификации предметных областей. Рассмотрим некоторые понятия и примеры, которые рассматриваются в рамках прикладных логик.

слотовое бинарное отношение

⇒ *примечание**:

[Слотовое бинарное отношение — слотовое sc-отношение, являющееся множеством.**]**

неслотовое бинарное отношение

⇒ *примечание**:

[Неслотовое бинарное отношение — бинарное sc-отношение, являющееся множеством, но не являющееся слотовым sc-отношением.**]**

иррефлексивное слотовое бинарное отношение

⊂ *иррефлексивное бинарное отношение*

⇒ *примечание**:

[Иррефлексивное слотовое бинарное отношение — слотовое (бинарное) отношение, любая связка которого не является связкой,обозначенной петлевой дугой (дугой с совпадающим началом и концом).**]**

иррефлексивное неслотовое бинарное отношение

⊂ *иррефлексивное бинарное отношение*

⇒ *примечание**:

[Иррефлексивное неслотовое бинарное отношение — неслотовое бинарное sc-отношение, для любой связки которого ее различные принадлежности являются принадлежностями различных элементов.**]**

рефлексивное слотовое бинарное отношение

⊂ *рефлексивное бинарное отношение*

⇒ *примечание**:

[Рефлексивное слотовое бинарное отношение — слотовое бинарное sc-отношение, для любого элемента связки которого найдется связка, обозначенная петлевой дугой (дугой с совпадающим началом и концом).**]**

рефлексивное неслотовое бинарное отношение

⊂ *рефлексивное бинарное отношение*

⇒ *примечание**:

[Рефлексивное неслотовое бинарное отношение — неслотовое бинарное sc-отношение, для любого элемента связки которого найдется связка, имеющая две различные принадлежности этого элемента.**]**

транзитивное слотовое бинарное отношение

⊂ *транзитивное бинарное отношение*

⇒ *примечание**:

[Транзитивное слотовое бинарное отношение — слотовое бинарное отношение, для любых двух связок которого, конец одной из которых является началом второй, существует связка началом которой является начало первой связки, а концом является конец второй связки.**]**

транзитивное неслотовое бинарное отношение

⊂ *транзитивное бинарное отношение*

⇒ *примечание**:

[Транзитивное неслотовое бинарное отношение — неслотовое бинарное отношение, для которого существует ролевое отношение, первым доменом которого является это бинарное отношение такое, что для любых двух связок этого бинарного отношения, принадлежность элемента одной из которых не принадлежит этому ролевому отношению, а принадлежность этого же элемента второй связке принадлежит этому ролевому отношению, существует связка с принадлежностью элемента, принадлежащей ролевому отношению, принадлежность которого первой связке принадлежит ролевому отношению, и с принадлежностью элемента, не принадлежащей ролевому отношению, принадлежность которого второй связке не принадлежит этому ролевому отношению.**]**

симметричное слотовое бинарное отношение

⊂ *симметричное бинарное отношение*

⇒ *примечание**:

[Симметричное слотовое бинарное отношение — слотовое бинарное отношение, для любой связки которого существует связка, конец которой является началом первой связки, а начало — ее концом (то есть эти связки обозначены встречными дугами).**]**

симметричное неслотовое бинарное отношение
⊂ *симметричное бинарное отношение*
⇒ *примечание**:
[Симметричное неслотовое бинарное отношение — неслотовое бинарное отношение, для которого существует ролевое отношение, первым доменом которого является это бинарное отношение, для любой связки которого существует связка с принадлежностью элемента первой связке, принадлежащая ролевому отношению, принадлежность которого второй связке не принадлежит этому ролевому отношению, и с принадлежностью элемента первой связке, не принадлежащая ролевому отношению, принадлежность которого второй связке принадлежит этому же ролевому отношению.]

антисимметричное слотовое бинарное отношение
⊂ *антисимметричное бинарное отношение*
⇒ *примечание**:
[Антисимметричное слотовое бинарное отношение — слотовое бинарное отношение, для любой связки которого у которой различным начало и конец не существует связки, конец которой является началом первой связки, а начало — ее концом (то есть эти связки обозначены встречными дугами).]

антисимметричное неслотовое бинарное отношение
⊂ *антисимметричное бинарное отношение*
⇒ *примечание**:
[Антисимметричное неслотовое бинарное отношение — неслотовое бинарное отношение, для которого существует ролевое отношение, первым доменом которого является это бинарное отношение, для любой связки которого не существует связки с принадлежностью элемента первой связке, принадлежащая ролевому отношению, принадлежность которого второй связке не принадлежит этому ролевому отношению, и с принадлежностью другого элемента первой связке, не принадлежащая ролевому отношению, принадлежность которого второй связке принадлежит этому же ролевому отношению.]

монотонное слотовое бинарное отношение*
⊂ *монотонное бинарное отношение**
⇒ *примечание**:
[Монотонное слотовое бинарное отношение* — слотовое бинарное отношение по отношению к отношению порядка, если есть связка этого бинарного отношения, то для любой его связки, начало которой связано связкой этого отношения порядка с началом первой, найдется третья связка бинарного отношения, начало которой совпадает с началом втором связки, а конец совпадает с концом первой.]

отношение порядка монотонного отношения'
⇒ *первый домен**:
*монотонное бинарное отношение**
⇒ *второй домен**:
отношение порядка

монотонное бинарное отношение'
⇒ *первый домен**:
*монотонное бинарное отношение**
⇒ *второй домен**:
монотонное бинарное отношение

монотонное неслотовое бинарное отношение*
⊂ *монотонное бинарное отношение**
⇒ *примечание**:
[Монотонное неслотовое бинарное отношение* — неслотовое бинарное отношение по отношению к отношению порядка, для которого существует ролевое отношение, первым доменом которого является это бинарное отношение, такое, что если есть связка этого бинарного отношения, то для любой его связки элемент принадлежащий ей под этим ролевым отношением в отличие от другого связан связкой этого отношения порядка с элементом принадлежащим первой связки под этим же ролевым отношением, в отличие от другого элемента первой связки, найдется третья связка бинарного отношения такая, что элемент, принадлежащий ей под ролевым отношением, принадлежит под этим же ролевым отношением второй связке, а элемент, принадлежащий не под ролевым отношением третьей связке, принадлежит не под ролевым отношением первой связке.]

слотовое отношение эквивалентности
⊂ *sc-отношение эквивалентности*
⇒ *примечание**:
[Слотовое отношение эквивалентности — слотовое транзитивное бинарное отношение, которое является слотовым рефлексивным и симметричным отношением.**]**

неслотовое отношение эквивалентности
⊂ *sc-отношение эквивалентности*
⇒ *примечание**:
[Неслотовое отношение эквивалентности — неслотовое транзитивное бинарное отношение, которое является неслотовым рефлексивным и симметричным отношением (по соответствующим доменам).**]**

слотовое отношение нестрогого порядка
⊂ *sc-отношение нестрогого порядка*
⇒ *примечание**:
[Слотовое отношение нестрогого порядка — транзитивное бинарное отношение, которое является рефлексивным и антисимметричным.**]**

неслотовое отношение нестрогого порядка
⊂ *sc-отношение нестрогого порядка*
⇒ *примечание**:
[Неслотовое отношение нестрогого порядка — транзитивное бинарное отношение, которое является рефлексивным и антисимметричным (по соответствующим доменам).**]**

Отношение выводимости
⊃ *Отношение выводимости на конечных множествах*
⊃ *Отношение выводимости на конечных множествах полносвязно представленных множеств*
∈ *рефлексивное бинарное отношение*
∈ *транзитивное бинарное отношение*
∈ *монотонное бинарное отношение*
⇒ *примечание**:
[Отношение выводимости — рефлексивное, транзитивное, монотонное бинарное отношение на множествах посылок (высказываний, логических формул). Свойствами отношения выводимости являются правила вывода по Генцену.**]**

секвенция
⇒ *примечание**:
[секвенция — связка (импликативного вида) между конъюнктивным множеством логических формул (конъюнкцией) и дизъюнктивным множеством логических формул (дизъюнкцией). Примером секвенции является выражение вида: $A_1 \wedge A_2 \wedge ... \wedge A_n \Rightarrow C_1 \vee C_2 \vee ... \vee C_m$.**]**

антецедент'
⇒ *первый домен**:
секвенция
⇒ *второй домен**:
конъюнкция

консеквент'
⇒ *первый домен**:
секвенция
⇒ *второй домен**:
дизъюнкция

Отношение выводимости на секвенциях
⇒ *примечание**:
[Отношение выводимости на секвенциях удовлетворяет правилам вывода исчисления секвенций.**]**

метаструктура
⇒ *примечание**:
[метаструктура — структура, полносвязно представленным элементом которой является другая структура.]

модальный оператор
⇒ *примечание**:
[модальный оператор — логическая связка, которая связывает логическую формулу со структурой (и иногда — другими элементами) в метаструктуре. Примером модального оператора является оператор знания: Δ.]

модальное правило вывода
⇒ *примечание**:
[модальное правило вывода — связка модального оператора формулы истинна (имеет истинную интерпретацию) в структуре, если и только если формула истинна (имеет истинную интерпретацию) в предшествующей ей структуре. Примером правила вывода является оператор знания: $\Gamma \cup \{\alpha\} \vdash \Gamma \cup \{\Delta\alpha\}$.]

отношение становления структур
⇒ *примечание**:
[Отношение становления структур — бинарное отношение на множестве структур, имеющих непустой общий носитель. Ролями в связке отношения становления являются ролевые отношения предшествующей структуры и последующей структуры.]

последовательность мышления
:= [судьба мышления]
:= [мысль]
⇒ *примечание**:
[последовательность мышления — последовательность sc-множеств высказываний (логических формул).]
⇒ *разбиение**:
{• *последовательность иррационального мышления*
• *последовательность рационального мышления*
}

последовательность рационального мышления классической логики
:= [судьба рационального мышления классической логики]
⊂ *последовательность рационального мышления*
⇒ *примечание**:
[последовательность рационального мышления — последовательность (заданная отношением становления структур) (классически) непротиворечивых sc-множеств (sc-подмножеств или sc-надмножеств) высказываний.]

последовательность классического рационального дедуктивного мышления
⊂ *последовательность рационального мышления классической логики*
⊃ *последовательность рационального дедуктивного мышления классической логики на конечных sc-множествах*
⇒ *примечание**:
[последовательность классического рационального дедуктивного мышления — последовательность рационального мышления, последовательность (становления) непротиворечивых sc-множеств высказываний, дедуктивно логически следующих (по классическим правилам) друг за другом.]

последовательность классического рационального дедуктивного познания
:= [воля]
⇒ *примечание**:
[последовательность классического рационального дедуктивного познания — последовательность (заданная отношением становления структур) непротиворечивых sc-надмножеств высказываний, логически следующих друг за другом.]

§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках

⇒ *ключевое понятие**:
- *немонотонный вывод на конечном sc-множестве посылок*
- *выводимое множество*
- *нечеткая истинность**
- *конструктивно истинное высказывание**
- *верное высказывание**
- *неискаженное высказывание**

неклассические логики (см. *Тейз А..ЛогичПкИИоК-1990кн*, *Тейз А..ЛогичПкИИоМ-1998кн*, *Драгалин А.Г.КонстТДиНА-2003кн*, *Батыршин И.З..НечетГСТиП-2007кн*, *Behounek L.Intro tMFL-2011bk*, *Вагин В.Н..Досто иПВвИС-2008кн*) рассматривают (1) неклассический вывод, в котором отношение выводимости обладает иными свойствами (см. *Тейз А..ЛогичПкИИоК-1990кн* , *Тейз А..ЛогичПкИИоМ-1998кн* , *Вагин В.Н..Досто иПВвИС-2008кн*, *Драгалин А.Г.КонстТДиНА-2003кн*) и при котором можно или нельзя вывести то, что выводимо в классической логике, а также (2) — другие шкалы признаков логических формул, их интерпретаций и значений (см. *Батыршин И.З..НечетГСТиП-2007кн*, *Behounek L.Intro tMFL-2011bk*), отличных от ложных и (достоверно) истинных.

немонотонный вывод на конечном sc-множестве посылок*

⇒ *примечание**:
[*немонотонный вывод на конечном sc-множестве посылок** — отношение* между (конечными) sc-множествами истинных *логических высказываний* (посылок). Если не существует вложения структуры *атомарной логической формулы* в реляционную структуру (sc-подмножество предметной области) *sc-множества* истинных (непротиворечивых) посылок и относительно них истинно отрицание этой *атомарной формулы*, то существует *sc-множество*, с принадлежащей ему реляционной структурой, включающей все элементы ранее упомянутой реляционной структуры и константы этой *атомарной формулы*, которому принадлежат все посылки ранее упомянутого sc-множества истинных (непротиворечивых) посылок и упомянутая атомарная логическая формула.**]**

выводимое множество

⇒ *примечание**:
[*выводимое множество — ситуативное sc-множество* (см. *Ивашенко В.П.Модел иАИЗнООСС-2014дс* , *Ивашенко В.П.ОнтолМПВОСиЯвПОЗ-2017ст*), (временная) принадлежность *логических формул* которому устанавливается в порядке становления процесса вывода этих *логических формул*.**]**

нечеткая истинность*

⇒ *примечание**:
[*нечеткая истинность** связывает конечное sc-множество с временными принадлежностями на конечном множестве конечных явлений принадлежности с высказыванием. На явлениях принадлежности задано конечное sc-подмножество sc-отношения становления (непосредственно прежде, непосредственно после), которое задает структуру соответствующих sc-подмножеств. Эта структура является ориентированным деревом. *нечеткая истинность** — *бинарное отношение* между (нечеткой) принадлежностью связки *высказывания формальной теории* и конечного sc-множества и действительным числом от 0.0 до 1.0. Нечеткая истинность отрицания высказывания равна разности 1.0 и нечеткой истинности высказывания, принадлежащего отрицанию. *нечеткая истинность** *конъюнкции** высказываний не превышает минимума нечеткой истинности элементов этой конъюнкции и не ниже (граничного или драстического) произведения *нечеткой истинности** этих же элементов конъюнкции. *нечеткая истинность** *дизъюнкции** не превышает (граничной или драстической) суммы *нечеткой истинности** элементов этой конъюнкции и не ниже максимума нечеткой истинности этих же элементов конъюнкции. *нечеткая истинность** *атомарных высказываний* равна среднему арифметическому изоморфного вложения структуры *высказывания* в каждое из sc-подмножеств конечного sc-множества, которые входят в структуру заданную sc-отношением становления.**]**

конструктивно истинное высказывание*

⇒ *примечание**:
[***конструктивно истинное высказывание**** — подмножество *истинного высказывания**. Истинные *атомарные логические формулы* или их истинные интерпретации — конструктивно истинные, если и только если они имеют изоморфное вложение в предметную область, где все элементы вложения полносвязно представлены. *конъюнкция** *конструктивно истинных логических формул* (или имеющих соответствую-

щие конструктивно истинные интерпретации) конструктивно истинна (или имеет конструктивно истинную соответствующую интерпретацию). *дизъюнкция** хотя бы одной *конструктивно истинной логической формулы* (или имеющей соответствующую полную конструктивно истинную интерпретацию) конструктивно истинна (или имеет конструктивно истинную соответствующую интерпретацию). *отрицание** *ложной логической формулы* (или имеющей ложную соответствующую интерпретацию) конструктивно истинное (или имеет конструктивно истинную интерпретацию). Если все *логические формулы* в *дизъюнкции** ложны (имеют соответствующие ложные интерпретации), то и дизъюнкция ложна (имеет соответствующую ложную интерпретацию). *отрицание** *ложной логической формулы* (или имеющей ложную соответствующую интерпретацию) конструктивно истинное (или имеет конструктивно истинную интерпретацию). *импликация** с ложной посылкой (или имеющей соответствующую ложную интерпретацию) конструктивна истинна (или имеет конструктивно истинную интерпретацию). *импликация** с конструктивно истинным следствием (или имеющим соответствующую конструктивно истинную интерпретацию) конструктивна истинна (или имеет конструктивно истинную интерпретацию). *конструктивно истинная импликация** (или имеющая конструктивно истинную интерпретацию) с конструктивно истинной посылкой (или имеющей соответствующую конструктивно истинную интерпретацию) имеет конструктивно истинное следствие (или имеющее соответствующую конструктивно истинную интерпретацию). *конструктивно истинная импликация** (или имеющая конструктивно истинную интерпретацию) с ложным следствием (или имеющим соответствующую ложную интерпретацию) имеет ложную посылку (или имеющую соответствующую ложную интерпретацию). Существование значений переменных для *логической формулы* конструктивно истинно (или имеет соответствующую конструктивную истинную интерпретацию), если всеобщность значений переменных для этой *логической формулы* конструктивно истинна (или имеет соответствующую конструктивную истинную интерпретацию). Если *логическая формула* имеет только конструктивно истинные соответствующие интерпретации, то всеобщность значений переменной для этой *логической формулы* конструктивно истинна (или имеет соответствующую конструктивно истинную интерпретацию).]

верное высказывание*

⇒ *примечание**:

[*верное высказывание** — *высказывание*, которое является истинным или неискаженным.]

неискаженное высказывание*

⇒ *примечание**:

[*неискаженое высказывание** — *высказывание*, верность или неверность которого не приводит к противоречию.]

Глава 2.7.

Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах

⇒ *автор**:
- *Никифоров С.А.*
- *Гойло А.А.*

⇒ *аннотация**:
[Глава посвящена языковым средствам формального описания *синтаксиса* и *денотационной семантики* различных *языков* в *ostis-системах*. Предложена *онтология* различных *языков*. Формализованы базовые *синтаксические правила* построения конструкций *естественных языков* и *правила соответствия синтаксических конструкций семантическому представлению*.]

⇒ *подраздел**:
- *§ 2.7.1. Формализация синтаксиса естественных языков*
- *§ 2.7.2. Формализация денотационной семантики естественных языков*

⇒ *ключевое понятие**:
- *язык*
- *лексема*
- *грамматическая категория*
- *часть речи*
- *словоформа*
- *составляющая*
- *синтаксическая группа*
- *вершина*
- *комплемент*
- *адъюнкт*
- *спецификатор*

⇒ *ключевое знание**:
- *Общие правила синтаксической структуры конструкций естественных языков*
- *Правила построения синтаксических групп*
- *Базовые правила денотационной семантики естественных языков*

⇒ *библиографическая ссылка**:
- *Golenkov V.V..MethoPotCSoW-2021art*
- *Pileggi S..Ontol iSE-2018art*
- *Lando P..Towar aGOoCP-2007art*
- *Farrar S..aCommoOfLC-2002art*
- *Chiarcos C..Inter oCaA-2012art*
- *TextEI-2022el*
- *EAGLERftMAoC-2022el*
- *Ide N..WhatDIMA-2010art*
- *Schalley A.C.Ontol aOMiL-2019art*
- *McCrae J.P..OneOtBTA-2015art*
- *tGenerOoLD-2022el*
- *Pease A..tSuggeUMOaL-2002art*
- *Farrar S..aLinguOftSW-2003art*
- *Chiarcos C..Ontol oLASaP-2012art*
- *Bateman J.A.tTheorSoOiNLP-1997art*
- *Bateman J.A..tGenerUMKBO-2002art*
- *Buitelaar P..TowarLGO-2009art*
- *Kostareva T..UsingODMtD-2016art*
- *Nevzorova O..OntolDPoUT-2019art*
- *Cimiano P..ExploOLfGNL-2013art*
- *Bouayad-Agha N..NaturLGitCotSW-2014art*
- *Saha D..Athen aOSfNL-2016art*
- *Shamsfard M..LearnOfNLT-2004art*

- *Bateman J.A..aLinguOoSfNLP-2010art*
- *Moens M..TempoOiNL-1987art*
- *Dobrov A..CompuOoTfMD-2018art*
- *What-2016el*
- *VerbNet-2022el*
- *Frame-2022el*
- *Huang C.ed.FormalOaItSaW-2010bk*
- *Matsukawa T..Devel otCDIoLK-1991art*
- *Calzolari N.Acqui aRSIiaLK-1991art*
- *Buitelaar P..LingiDaAoaMftIoL-2006art*
- *Cimiano P..Lexon aMfOLfON-2007art*
- *Buitelaar P..aMultiMLMfO-2006art*
- *McCrae J..InterLRotSW-2012art*
- *SemanW-2022el*
- *NLTKNLPL-2022el*
- *SpaCyNL-2022el*
- *Erekhinskaya T.N..TenWoLOfR-2020art*
- *Голенков В.В..СтандОТОП-2021кн*
- *SIL:GoLT-2022el*
- *Adger D.CoreSaMA-2003bk*
- *Jackendoff R..XS aSoPS-1977bk*
- *Haegeman L.Intro tGaBT-1994bk*
- *Carnie A.Syntax aGL-2012bk*
- *Heim I..Seman iGG-1998bk*
- *Winter Y.Eleme oFS-2016bk*
- *Portner P.H..FormaStER-2008bk*

Введение в Главу 2.7.

В настоящее время научные исследования в области искусственного интеллекта развиваются по большому спектру различных направлений, однако между ними отсутствует согласованность систем понятий и, как следствие этого, совместимость разрабатываемых систем (см. *Golenkov V.V..MethoPotCSoW-2021art*).

Так в области создания программного обеспечения в силу его значительной сложности остро стоит проблема обеспечения интероперабельности различных программных сущностей, а также переиспользования результатов предыдущих аналогичных разработок.

Одним из путей решения данных проблем является создание *онтологий* программного обеспечения (см. *§ 2.5.5. Формализация понятия онтологии*, *§ 3.2.2. Существующие онтологии языков программирования*), к которым предъявляются следующие требования (см. *Pileggi S..Ontol iSE-2018art*):

- путем спецификация формальной семантики для избежания двусмысленных определений, а также нежелательных интерпретаций с целью обеспечения интероперабельности;
- созданные *онтологии* должны быть применены в иной или же более широкой предметной области, что позволило бы избежать дорогостоящей специальной разработки и может повысить качество конечного продукта;
- возможность применения на их базе механизмов логического вывода.

В качестве примера *онтологии* в данной области можно привести COPS (см. *Lando P..Towar aGOoCP-2007art*). Целью данной *онтологии* являлась формализация общих понятий из области программного обеспечения с целью упрощения его разработки и использования.

Проблема совместимости результатов исследований также остро стоит и в лингвистике — науке, в которой существует множество различных теорий, часто несовместимых друг с другом. В лингвистических исследованиях используются разные варианты разметки данных, нет одного подхода к структуризации корпусов текстов и различаются способы представления данных в них (см. *Farrar S..aCommoOfLC-2002art*, *Chiarcos C..Inter oCaA-2012art*).

В качестве решения проблемы несовместимости различных способов описания данных в лингвистике предлагались варианты стандартизации форматов такого описания. Примером могут служить *Text Encoding Initiative* — консорциум по стандартизации представления текстов в цифровом виде (см. *TextEI-2022el*) и гайдлайны экспертной группы по стандартизации представления языковых данных *EAGLES* (например, рекомендации по разметке корпусов текстов (см. *EAGLERftMAoC-2022el*). Однако ни один из таких стандартов не получил распространения и не стал использоваться лингвистами повсеместно (см. *Ide N..WhatDIMA-2010art* (с. 4))).

Вместо создания рекомендаций по разметке языкового материала в качестве более эффективного средства решения указанных выше проблем предлагается создание *онтологий* (см. *Schalley A.C.Ontol aOMiL-2019art*, *McCrae J.P..OneOtBTA-2015art*). Помимо того, что *онтология верхнего уровня* для предметной области языкознания может служить связующим звеном между различными лингвистическими теориями, она также представляет собой формализованное описание лингвистических концептов, представленное в удобном для компьютеров формате, что обусловливает ее применимость в системах, способных понимать аннотированные языковые данные, совершать интеллектуальный поиск по корпусам текстов, а также потенциально выполнять анализ существующих лингвистических исследований (см. *Farrar S..aCommoOfLC-2002art*).

В качестве такой *онтологии* в предметной области лингвистики выступает *The General Ontology of Linguistic Description* (*GOLD*) (см. *tGenerOoLD-2022el*). В этой *онтологии* формализованы наиболее базовые категории и отношения, используемые в лингвистике, а сама онтология интегрирована с онтологией верхнего уровня *Suggested Upper Merged Ontology* (*SUMO*) (см. *Pease A..tSuggeUMOaL-2002art*). Авторы *GOLD* пишут, что создавали онтологию в первую очередь для того, чтобы решить проблему интероперабельности данных лингвистической типологии и для того, чтобы с ее помощью экспертные системы могли обрабатывать научные данные по естественным языкам — то есть целью создателей *онтологии* не являлось непосредственно решение задач из области обработки текстов на естественном языке (см. *Farrar S..aLinguOftSW-2003art* (с.4)).

онтологией естественных языков, нацеленной непосредственно на использование при решении задач по обработке естественного языка, является *Ontologies of Linguistic Annotation* (*OLiA*) (см. *Chiarcos C..Ontol oLASaP-2012art*). Основной идеей *онтологии* является обеспечение совместимости разметки языковых данных, полученных в результате выполненного компьютерными системами анализа текстов на *естественном языке* с соответствующими им лингвистическими концептами из *онтологии* — в отличие от других лингвистических *онтологий*, *OLiA* предоставляет не только инвентарь концептов и отношений, но и необходимую спецификацию интеграции этих элементов с разметкой языковых данных (например, в корпусах (см. *Chiarcos C..Ontol oLASaP-2012art* (с. 4)).

При создании *онтологий естественного языка*, встает вопрос о статусе спецификации лингвистической информации в таких онтологиях. Дж. Бейтман выделяет три типа онтологий в зависимости от интегрированности в них естественно-языковой информации (см. *Bateman J.A.tTheorSoOiNLP-1997art*):

- *онтологии*, представляющие собой абстрактную семантико-концептуальную репрезентацию знаний о мире, которая используется непосредственно в качестве *денотационной семантики* для *синтаксиса* и *лексики* естественного языка;
- *онтологии*, в которых есть отдельная спецификация *денотационной семантики естественного языка*, которая служит интерфейсом между *синтаксисом* естественных *языков* и концептуальной онтологией;
- *онтологии*, представляющие собой абстрактную спецификацию *знаний* о реальном мире вне зависимости от ограничений *естественного языка*.

Популярность в сфере обработки естественного языка приобрел второй тип *онтологии* (см. *Bateman J.A.-tTheorSoOiNLP-1997art* (с. 8)), так как он, в отличие от третьего подхода, который совсем не формализует лингвистическую информацию, позволяет специфицировать больше информации о естественных языках. Так, одна из самых популярных онтологий, используемых в системах для обработки естественного языка, the *Generalized Upper Model* (см. *Bateman J.A..tGenerUMKBO-2002art*), является онтологией второго типа (см. *Bateman J.A.-tTheorSoOiNLP-1997art*). П. Буителар и другие подчеркивают, что всем *формальным онтологиям* необходима связь с языковой информацией для решения таких задач как выделение информации из текстов *естественного языка*, автоматизированное заполнение *онтологий* и генерации текста на *естественном языке* (см. *Buitelaar P..TowarLGO-2009art*).

Так как использование онтологий в обработке *естественного языка* позволяет задать семантику получаемым в результате обработки *естественного языка* данным и потенциально повысить качество анализа, начинается переход к созданию движимых *онтологиями* систем обработки *естественных языков* (см. *Kostareva T..UsingODMtD-2016art* ,*Nevzorova O..OntolDPoUT-2019art*). *онтологии естественного языка* активно применяются для генерации текстов *естественного языка* на основе некоторой *онтологии* предметной области (см. *Cimiano P..ExploOLfGNL-2013art* , *Bouayad-Agha N..NaturLGitCotSW-2014art*).

Онтологический подход также используется в системах естественно-языковых запросов для баз данных, в которых запрос на *естественном языке* транслируется в *язык* запросов по *онтологиям* конкретных *предметных областей*, конструкции которого затем транслируются в SQL для обеспечения взаимодействия с реляционными базами данных (см. *Saha D..Athen aOSfNL-2016art*).

Кроме того, спецификация лингвистической информации в виде *онтологий* помогает решать задачу автоматизированного создания *онтологий* на основе естественно-языковых текстов (см. *Shamsfard M..LearnOfNLT-2004art*).

Создаются *онтологии* частных областей лингвистики: например, онтология пространственных выражений в естественных языках (см. *Bateman J.A..aLinguOoSfNLP-2010art*), онтология темпоральных сущностей на основе естественного языка (см. *Moens M..TempoOiNL-1987art*), *онтологии* конкретных естественных языков (см. *Dobrov*

A..CompuOoTfMD-2018art). При использовании *онтологий* для обработки естественного языка необходимо "связать" концепты из *онтологии* с лексикой конкретного *естественного языка*. Для этого создаются различные расширения существующих языковых баз данных, таких как *WordNet* (см. *What-2016el*), *VerbNet* (см. *VerbNet-2022el*) и *FrameNet* (см. *Frame-2022el*), направленные на их использование совместно с *онтологиями верхнего уровня* (например, см. *Huang C.ed.FormalOaItSaW-2010bk*). Активные разработки идут в сфере создания онтологий словарного состава *естественных языков*, в результате которых появилось множество формализованных описаний лексики (см. *Matsukawa T..Devel otCDIoLK-1991art*, *Calzolari N.Acqui aRSIiaLK-1991art*, *Buitelaar P..LingiDaAoaMftIoL-2006art*, *Cimiano P..Lexon aMfOLfON-2007art*, *Buitelaar P..aMultiMLMfO-2006art*). Так как распространенные базы данных лексики естественного языка не являются *онтологиями* и не имеют достаточной степени формализации (например, *WordNet*), создаются *онтологии*, являющиеся своего рода "надстройкой" над такими базами данных, самой известной из которых является *lemon* (см. *McCrae J..InterLRotSW-2012art*.

Многие из приведенных выше онтологий созданы с использованием технологии *Semantic Web* (см. *SemanW-2022el*), который является внешней технологией по отношению к существующим решениям для обработки *естественных языков*, поэтому последним приходится обращаться к ней с помощью API и стандартизированных *языков запросов* (в частности, *SPARQL*) (см. *Bouayad-Agha N..NaturLGitCotSW-2014art*).

Стоит отметить, что несмотря на активное развитие в направлении применения *онтологий* для обработки *естественного языка*, многие популярные библиотеки по обработке естественного языка (например, *NLTK* (см. *NLTKNLPL-2022el*) и *spaCy* (см. *SpaCyNL-2022el*) в принципе не поддерживают использование *онтологий*, а большинство инструментов для разметки естественно-языковых текстов используют обычно свой формат, что требует использования специфичных для таких инструментов парсеров и конвертеров, чтобы данные можно было применить при решении каких-либо задач (см. *Erekhinskaya T.N..TenWoLOfR-2020art* (с. 3))).

Таким образом, в настоящее время в данной области можно выделить следующие проблемы:

- Отсутствие унификации (стандартизации) приведенных выше решений приводит к существенным накладным расходам на их интеграцию и значительно усложняет построение различных систем с их использованием в силу большой трудоемкости их интеграции (см. *Голенков В.В..СтандОТОП-2021кн*, *Golenkov V.V..MethoPotCSoW-2021art*).
- Несмотря на то, что *онтологии* потенциально способствуют решению широкого круга задач в сфере обработки естественного языка, большинство движимых *онтологиями* систем по обработке естественного языка сконцентрированы на решении специализированных задач (например, только генерации текста, только заполнения *онтологии* или только обеспечения поиска с помощью естественного языка).
- Создано довольно большое количество частных лингвистических *онтологий*, формализующих, однако, лишь некий подраздел предметной области лингвистики (в особенности лексики), что отчасти вытекает из предыдущего пункта. В то же время, существующие лингвистические *онтологии* верхнего уровня (например, *OLiA*) все равно не до конца решают проблему унификации, так как им требуется вводить промежуточный уровень для интеграции полученных в результате анализа текста естественного языка данных с фрагментами *онтологии*.

Так как используемый в *Технологии OSTIS язык — SC-код* (см. Главу *2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*) — обладает достаточной экспрессивностью для описания знаний любого вида, а сама технология нацелена на создание интероперабельных интеллектуальных систем нового поколения, *естественно-языковые интерфейсы ostis-систем* смогут справляться с широким кругом задач по обработке текстов на естественных языках — будь то синтез естественно-языковых текстов в целом, ведение диалога в диалоговых системах, поиск с использованием *естественного языка*, выделение информации из текстов и тому подобное. При этом в то время как в текущем состоянии сферы обработки естественных языков данные классы задач выполняются зачастую специализированными средствами и требуют дополнительных затрат на обеспечение потенциальной совместимости с конкретными компьютерными системами, в рамках *Технологии OSTIS* для их решения будет использоваться один универсальный *язык смыслового представления знаний*, на котором будут написаны как компоненты решателя задач, так и онтология языков и конкретных предметных областей, что позволит решить проблему интероперабельности.

Более того, *онтология естественных языков*, разработанная в рамках такой технологии, могла бы быть использована не только для решения прикладных задач по обработке *естественного языка*, но и для обеспечения интероперабельности данных, полученных в ходе лингвистических исследований, что было бы ценным вкладом в область теоретической лингвистики.

Наконец, *онтологию естественных языков* можно рассматривать в качестве подмножества *онтологии* языков вообще (как естественных, так и искусственных и формальных), чего не делают рассмотренные выше существующие *онтологии*. Это позволит концептуализировать *естественный язык* в одной системе с языками программирования и более тесно связать используемые в соответствующих предметных областях понятия для более эффективного решения задач по обработке *естественного языка* в интеллектуальных компьютерных системах.

Цель данной работы — предложить базовые средства формального описания *синтаксиса* и *денотационной семантики* различных *языков* в виде фрагмента *онтологии языков* и *информационных конструкций*, который можно будет использовать при проектировании интеллектуальных компьютерных систем нового поколения.

Как уже говорилось выше, для использования достижений лингвистики при проектировании интеллектуальных компьютерных систем требуется представить полученные результаты в формальном виде.

Далее мы предложим формализацию основных лингвистических концептов, выполненную на формальном языке представления знаний — *SC-коде*.

язык
⇒ *разбиение**:
{• *естественный язык*
⇒ *пояснение**:
[*естественный язык* представляет собой *язык*, который не был создан целенаправленно]
• *искусственный язык*
⇒ *пояснение**:
[*искусственный язык* представляет собой *язык*, специально разработанный для достижения определенных целей]
∋ *Эсперанто*
∋ *Python*
⊃ *сконструированный язык*
⇒ *пояснение**:
[*сконструированный язык* представляет собой искусственный *язык*, предназначенный для общения людей]
∋ *Эсперанто*
}
⊃ *международный язык*
⇒ *пояснение**:
[*международный язык* представляет собой *естественный* или *искусственный язык*, использующийся для общения людей из разных стран]
∋ *Английский язык*
∋ *Русский язык*

плановый язык
⇐ *пересечение**:
{• *сконструированный язык*
• *международный язык*
}

язык общения
⇐ *объединение**:
{• *естественный язык*
• *сконструированный язык*
}
∋ *Английский язык*
∋ *Русский язык*
∋ *Эсперанто*
⇐ *объединение**:
{• *корневой язык*
⇒ *пояснение**:
[*корневой язык* представляет собой *язык*, для которого характерно полное отсутствие словоизменения и наличие грамматической значимости порядка слов, состоящих только из корня.]
∋ *Английский язык*
• *агглютинативный язык*
⇒ *пояснение**:
[*агглютинативный язык* характеризуется развитой системой употребления суффиксов, приставок, добавляемых к неизменяемой основе слова, которые используются для выражения категорий числа, падежа, рода и так далее]
∋ *Английский язык*
• *флективный язык*
⇒ *пояснение**:
[Для *флективного языка* характерно развитое употребление окончаний для выражения категорий рода, числа, падежа, сложная система склонения глаголов, чередование гласных в корне, а также строгое различение частей речи.]
∋ *Русский язык*

- *профлективный язык*
  ⇒ *пояснение**:
  [Для *профлективного языка* характерны агглютинация (в случае именного словоизменения), флексия и чередование гласных (аблаут)(в случае глагольного словоизменения).]

}

§ 2.7.1. Формализация синтаксиса естественных языков

Приводимая ниже формализация *синтаксиса естественных языков* является ядром, общим для всех *естественных языков*. Очевидно, что *синтаксис* некоторого конкретного *естественного языка* может отличаться от *синтаксиса* других *языков*. В таком случае, более частные отличия необходимо отдельно специфицировать в формализованном виде. Таким образом, ниже будет предложено описание наиболее общих аспектов *синтаксиса* всех *естественных языков*, которое в дальнейшем может дополняться, если того требует конкретная реализация *естественно-языкового интерфейса* какой-либо *ostis-системы*. При этом любые дополнения, специфичные для некоторого конкретного *естественного языка*, не должны противоречить ядру формализации *синтаксиса естественных языков*, приведенному в данном разделе.

лексема — минимальная единица *языка*, имеющая семантическую интерпретацию и обозначающая концепт, отражающий взгляд на мир некоторого языкового сообщества (см. *SIL:GoLT-2022el*).

грамматическая категория — система противопоставленных друг другу рядов грамматических форм с однородными значениями. В рамках нашей формализации предлагается представить грамматические категории как классы ролевых отношений, каждый из которых соответствует определенному грамматическому значению. Следует отметить, что приводятся основные *грамматические категории*, часто встречающиеся в *естественных языках*, а не всех возможные.

грамматическая категория
∋ *лицо*
  ⇐ *семейство подмножеств**:
    ролевое отношение
  ∋ *первое лицо'*
  ∋ *второе лицо'*
  ∋ *третье лицо'*
∋ *число*
  ⇐ *семейство подмножеств**:
    ролевое отношение
  ∋ *единственное число'*
  ∋ *множественное число'*
  ∋ *двойственное число'*
  ∋ *тройственное число'*
  ∋ *паукальное число'*
∋ *род*
  ⇐ *семейство подмножеств**:
    ролевое отношение
  ∋ *мужской род'*
  ∋ *средний род'*
  ∋ *женский род'*
∋ *падеж*
  ⇐ *семейство подмножеств**:
    ролевое отношение
  ∋ *именительный падеж'*
  ∋ *родительный падеж'*
  ∋ *дательный падеж'*
  ∋ *винительный падеж'*
  ∋ *творительный падеж'*
  ∋ *предложный падеж'*
  ∋ *звательный падеж'*
  ∋ *абсолютивный падеж'*
  ∋ *эргативный падеж'*
∋ *время*

⇐ *семейство подмножеств**:
ролевое отношение
∋ *настоящее время'*
∋ *прошедшее время'*
∋ *будущее время'*

∋ *наклонение*
⇐ *семейство подмножеств**:
ролевое отношение
∋ *изъявительное наклонение'*
∋ *повелительное наклонение'*
∋ *сослагательное наклонение'*
∋ *условное наклонение'*

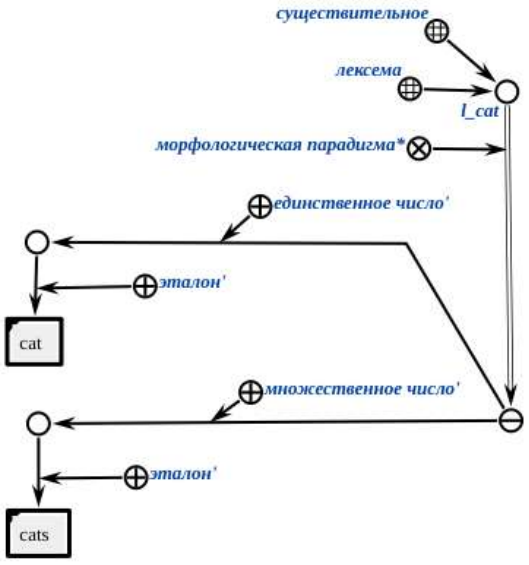
∋ *залог*
⇐ *семейство подмножеств**:
ролевое отношение
∋ *действительный залог'*
∋ *страдательный залог'*
∋ *средний залог'*
∋ *возвратный залог'*
∋ *взаимный залог'*

∋ *вид*
⇐ *семейство подмножеств**:
ролевое отношение
∋ *совершенный вид'*
∋ *несовершенный вид'*
∋ *общий вид'*
∋ *прогрессивный вид'*
∋ *перфектный вид'*

∋ *степень сравнения*
⇐ *семейство подмножеств**:
ролевое отношение
∋ *положительная степень сравнения'*
∋ *сравнительная степень сравнения'*
∋ *превосходная степень сравнения'*

Пример формализации части приведенных выше *отношений* в *SCg-коде* приведен на *SCg-текст. Иллюстрация к спецификации лексемы в базе знаний.*.

SCg-текст. Иллюстрация к спецификации лексемы в базе знаний.
=



часть речи — *категория*, представляющая собой класс синтаксически эквивалентных *знаков естественного языка*.

часть речи
⇐ *семейство подмножеств**:
лексема
∋ *существительное*
∋ *прилагательное*
∋ *глагол*
∋ *наречие*
∋ *предлог*
∋ *комплементатор*
∋ *вспомогательный глагол*
∋ *детерминант*

морфологическая парадигма* — *бинарное ориентированное отношение*, связывающее *лексему* и множество ее *словоформ*.

словоформа — *подмножество лексемы*, которому принадлежат все вхождения *лексемы* с определенными *грамматическими значениями*. В рамках нашей *онтологии* словоформа понимается несколько иначе, чем принято в лингвистике, так как все вхождения лексемы в технологии OSTIS являются *файлами*.

При формализации синтаксиса в основном использовались стандартные положения генеративной грамматики (см. *Adger D.CoreSaMA-2003bk*, *Jackendoff R..XS aSoPS-1977bk*, *Haegeman L.Intro tGaBT-1994bk*, *Carnie A.Syntax aGL-2012bk*).

дистрибуция знака — это подмножество синтаксических правил, в которые входит данный *знак*.

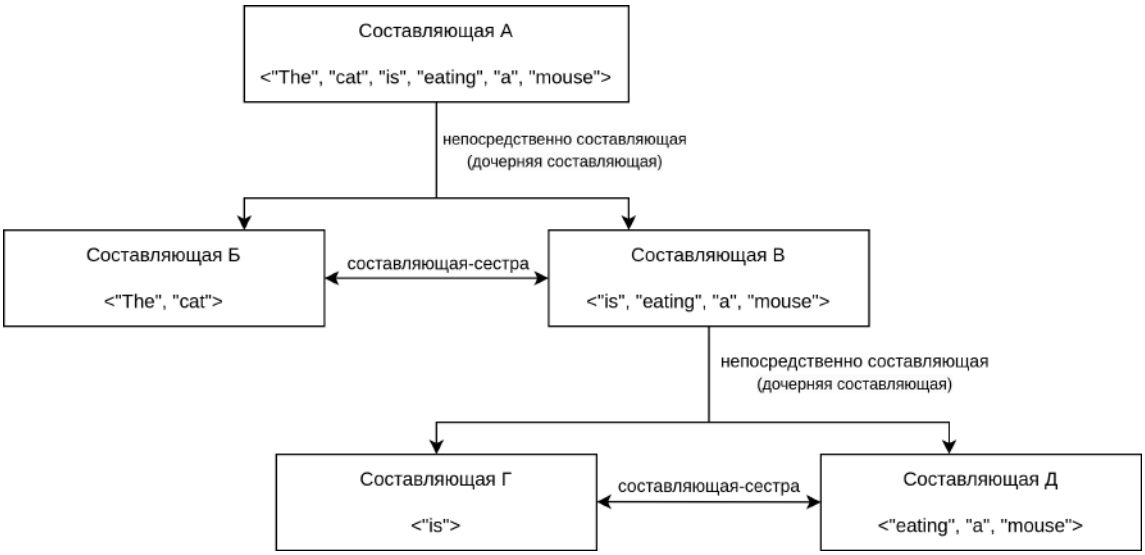
составляющая — элемент множества *C* подмножеств кортежа вхождений лексем *S*, которое содержит в качестве элементов как сам *S*, так и все вхождения лексем в *S*, таким образом, что любые два подмножества, входящие в *C*, либо не пересекаются, либо одно из них включается в другое.

непосредственно составляющая — есть множество *составляющих S*, в которое входят *составляющие A* и *B*. B является *непосредственно составляющей A* если и только если *B* является подмножеством *A* и нет такой *составляющей C*, которая является подмножеством *A* и подмножеством которой является *B*.

составляющими-сестрами* считаются *составляющие*, являющиеся *непосредственно составляющими* одной и той же *составляющей*.

Связи между составляющими представлены на *Рисунок. Иллюстрация связей между составляющими.*.

Рисунок. Иллюстрация связей между составляющими.
=



элементарная составляющая — элемент кортежа вхождений *лексем L*, являющихся *непосредственно составляющими* множества *составляющих C* и не имеющих непосредственно составляющих *составляющих*.

синтаксическая группа — класс *составляющих*, в который входят *составляющие* с вершинами, принадлежащими к одной *части речи*. *синтаксические группы* представляют собой либо *синглетон* (минимально включают в себя вершину), либо упорядоченную пару, состоящую из *вершины* и другой *синтаксической группы*.

вершина — *составляющая*, *дистрибуция* которой совпадает с *дистрибуцией* всей *синтаксической группы*.

составляющая
⇒ *разбиение**:
{• *синтаксическая группа*
• *вершина*
}

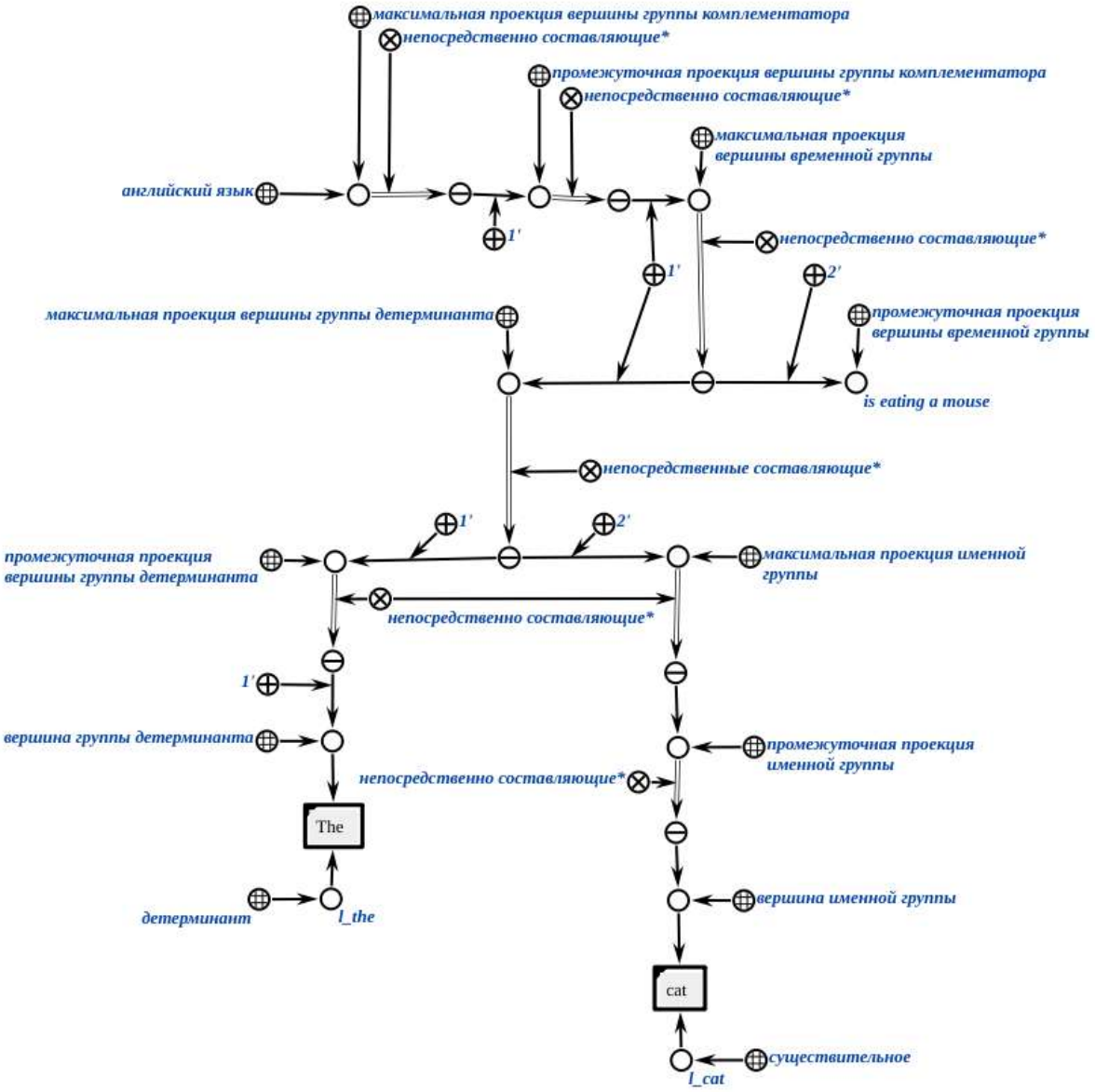
синтаксическая группа
⇒ *разбиение**:
{• *именная группа*
⇒ *пояснение**:
[*Именная группа — синтаксическая группа*, *вершиной* которой является *существительное*.]
• *глагольная группа*
⇒ *пояснение**:
[*Глагольная группа — синтаксическая группа*, *вершиной* которой является *глагол*.]
• *группа прилагательного*
⇒ *пояснение**:
[*Группа прилагательного — синтаксическая группа*, *вершиной* которой является *прилагательное*.]
• *наречная группа*
⇒ *пояснение**:
[*Наречная группа — синтаксическая группа*, *вершиной* которой является *наречие*.]
• *предложная группа*
⇒ *пояснение**:
[*Предложная группа — синтаксическая группа*, *вершиной* которой является *предлог*.]
• *группа комплементатора*
⇒ *пояснение**:
[*Группа комплементатора — синтаксическая группа*, *вершиной* которой является *комплементатор*.]
• *временная группа*
⇒ *пояснение**:
[*Временная группа — синтаксическая группа*, *вершиной* которой является *вспомогательный* либо *модальный глагол*.]
• *группа детерминанта*
⇒ *пояснение**:
[*Группа детерминанта — синтаксическая группа*, *вершиной* которой является *детерминант*.]
}
⇒ *разбиение**:
{• *максимальная проекция вершины*
:= [максимальная проекция вершины синтаксической группы]
• *промежуточная проекция вершины*
:= [промежуточная проекция вершины синтаксической группы]
}

При этом для упрощения могут быть введены более узкие классы, являющиеся пересечением приведенных выше, например *максимальная проекция вершины группы детерминанта*.

максимальная проекция вершины группы детерминанта
⇐ *пересечение**:
{• *группа детерминанта*
• *максимальная проекция вершины*
}

Пример синтаксической структуры предложения, записанный с применением введенных выше понятий представлен на *SCg-текст. Иллюстрация синтаксической структуры предложения. Первая часть* и *SCg-текст. Иллюстрация синтаксической структуры предложения. Вторая часть*.

SCg-текст. Иллюстрация синтаксической структуры предложения. Первая часть
=



Структуры синтаксических групп не являются произвольными — элементы внутри группы могут граничить только с определенными множествами элементов. Ниже приводятся возможные структуры синтаксических групп. В скобках указаны опциональные элементы.

Группа детерминанта:

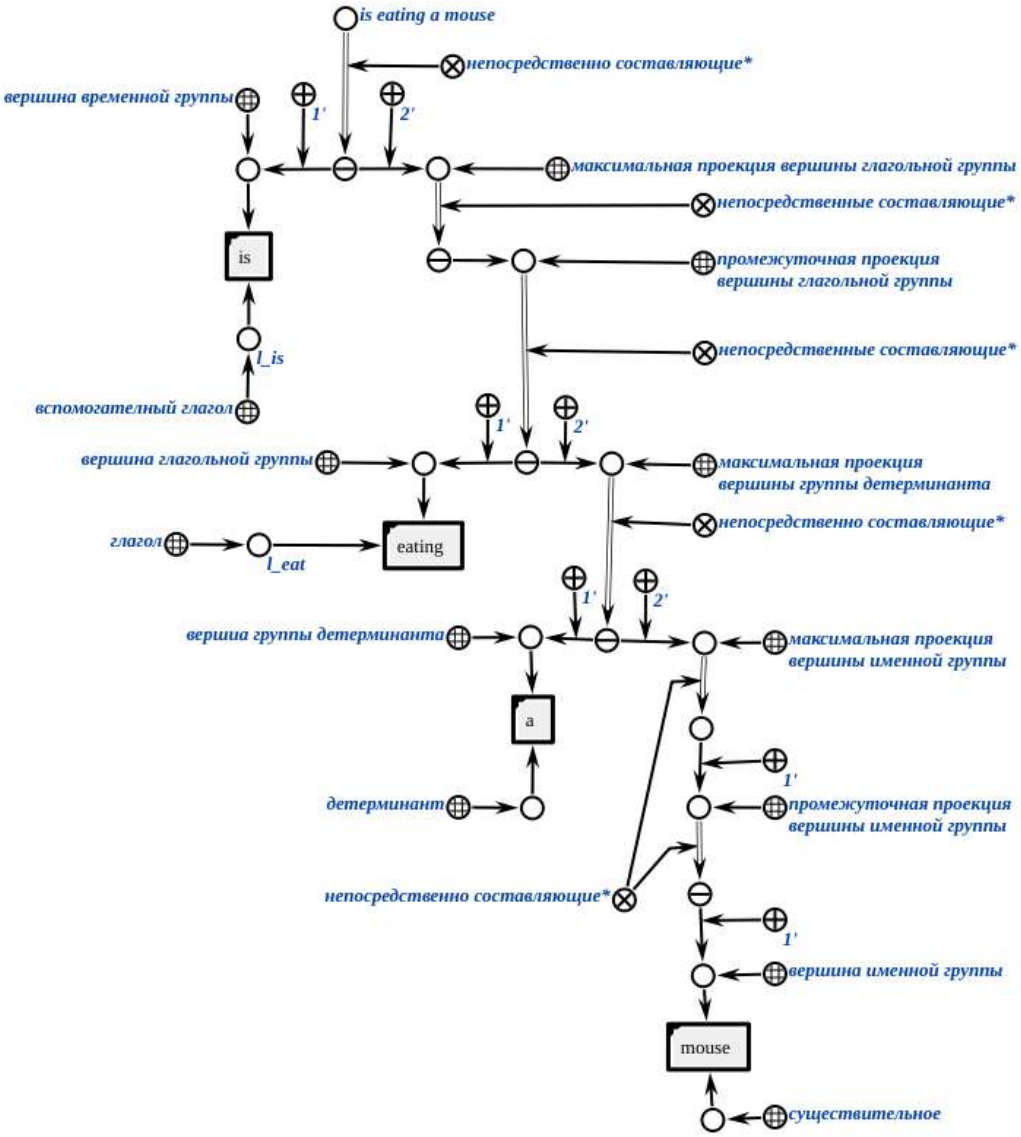
- *максимальная проекция вершины группы детерминанта* состоит из (*максимальной проекции вершины группы детерминанта*) и *промежуточной проекции вершины группы детерминанта*;
- *промежуточная проекция вершины группы детерминанта* состоит из *вершины группы детерминанта* (и *максимальной проекции вершины именной группы*).

Именная группа:

- *максимальная проекция вершины именной группы* состоит из (*максимальной проекции вершины группы детерминанта*) и *промежуточной проекции вершины именной группы*;
- *промежуточная проекция вершины именной группы* состоит из (*максимальной проекции вершины группы прилагательного*) и *промежуточной проекции вершины именной группы* ИЛИ *промежуточной проекции вершины именной группы* (и *максимальной проекции вершины предложной группы*);
- *промежуточная проекция вершины именной группы* состоит из *вершины именной группы* (и *максимальной проекции вершины предложной группы*).

Глагольная группа:

SCg-текст. Иллюстрация синтаксической структуры предложения. Вторая часть
=



- *максимальная проекция вершины глагольной группы* состоит из *промежуточная проекция вершины глагольной группы*;
- *промежуточная проекция вершины глагольной группы* состоит из *промежуточной проекции вершины глагольной группы* (и *максимальной проекции вершины предложной группы*) ИЛИ *промежуточной проекции вершины глагольной группы* (и *максимальной проекция вершины наречной группы*);
- *промежуточная проекция вершины глагольной группы* состоит из *вершины глагольной группы* (и *максимальной проекции вершины именной группы*).

Наречная группа:
- *максимальная проекция вершины наречной группы* состоит из *промежуточной проекции вершины наречной группы*;
- *промежуточная проекция вершины наречной группы* состоит из (*максимальной проекции вершины наречной группы*) и *промежуточной проекции вершины наречной группы*;
- *промежуточная проекция вершины наречной группы* состоит из *вершины наречной группы* (и *максимальной проекции вершины предложной группы*).

Группа прилагательного:
- *максимальная проекция вершины группы прилагательного* состоит из *промежуточной проекции вершины группы прилагательного*;
- *промежуточная проекция вершины группы прилагательного* состоит из (*максимальной проекции вершины наречной группы*) и *промежуточной проекции вершины группы прилагательного*;

- *промежуточная проекция вершины группы прилагательного* состоит из *вершины группы прилагательного* (и *максимальной проекцим вершины предложной группы*).

Предложная группа:
- *максимальная проекция вершины предложной группы* состоит из *промежуточной проекции вершины предложной группы*;
- *промежуточная проекция вершины предложной группы* состоит из *промежуточной проекции вершины предложной группы* (и *максимальной проекции вершины предложной группы*) ИЛИ (*максимальной проекции вершины наречной группы*) и *промежуточной проекции вершины предложной группы*;
- *промежуточная проекция вершины предложной группы* состоит из *вершины предложной группы* (и *максимальной проекции вершины именной группы*).

Временная группа:
- *максимальная проекция вершины временной группы* состоит из (*максимальной проекции вершины группы детерминанта*) и *промежуточной проекции вершины временной группы*;
- *промежуточная проекция вершины временной группы* состоит из *вершины временной группы* (и *максимальной проекции вершины глагольной группы*).

Группа комплементатора:
- *максимальная проекция вершины группы комплементатора* состоит из (*максимальной проекции вершины некоторой синтаксической группы*) и *промежуточной проекции вершины группы комплементатора*;
- *промежуточная проекция вершины группы комплементатора* состоит из *вершины группы комплементатора и максимальной проекции вершины временной группы*.

В формальном виде данные правила можно представить следующим образом (см. *SCg-текст. Иллюстрация правила структуры синтаксической группы*).

комплемент — *синтаксическая группа*, являющаяся сестрой вершины.

адъюнкт — *синтаксическая группа*, являющаяся дочерью (*непосредственно составляющей*) промежуточной проекции и сестрой промежуточной проекции вершины той же *синтаксической группы*.

спецификатор — *синтаксическая группа*, являющейся дочерью максимальной проекции и сестрой промежуточной проекции.

Приведенные выше правила структуры синтаксических групп можно обобщить и свести к трем более абстрактным.

Правило спецификатора: *максимальная проекция вершины синтаксической группы XP* состоит из (*максимальной проекции вершины YP*) *промежуточной проекции вершины синтаксической группы X'*

В формализованном виде данное правило представлено на *SCg-текст. Правило спецификатора*.

Правило адъюнкта: *промежуточная проекция вершины синтаксической группы X'* состоит из *промежуточной проекции вершины синтаксической группы* $\boldsymbol{X_1'}$ (и *максимальной проекции вершины синтаксической группы ZP*) ИЛИ из (*максимальной проекции вершины синтаксической группы ZP*) и *промежуточной проекции вершины синтаксической группы X'*.

В формализованном виде данное правило представлено на *SCg-текст. Правило адъюнкта*.

Правило комплемента: *промежуточная проекция вершины синтаксической группы X'* состоит из *вершины X* (и *максимальной проекции вершины синтаксической группы WP*).

В формализованном виде данное правило представлено на *SCg-текст. Правило комплемента*.

Формальное представление данных правил аналогично приведенному на *SCg-текст. Иллюстрация правила структуры синтаксической группы*.

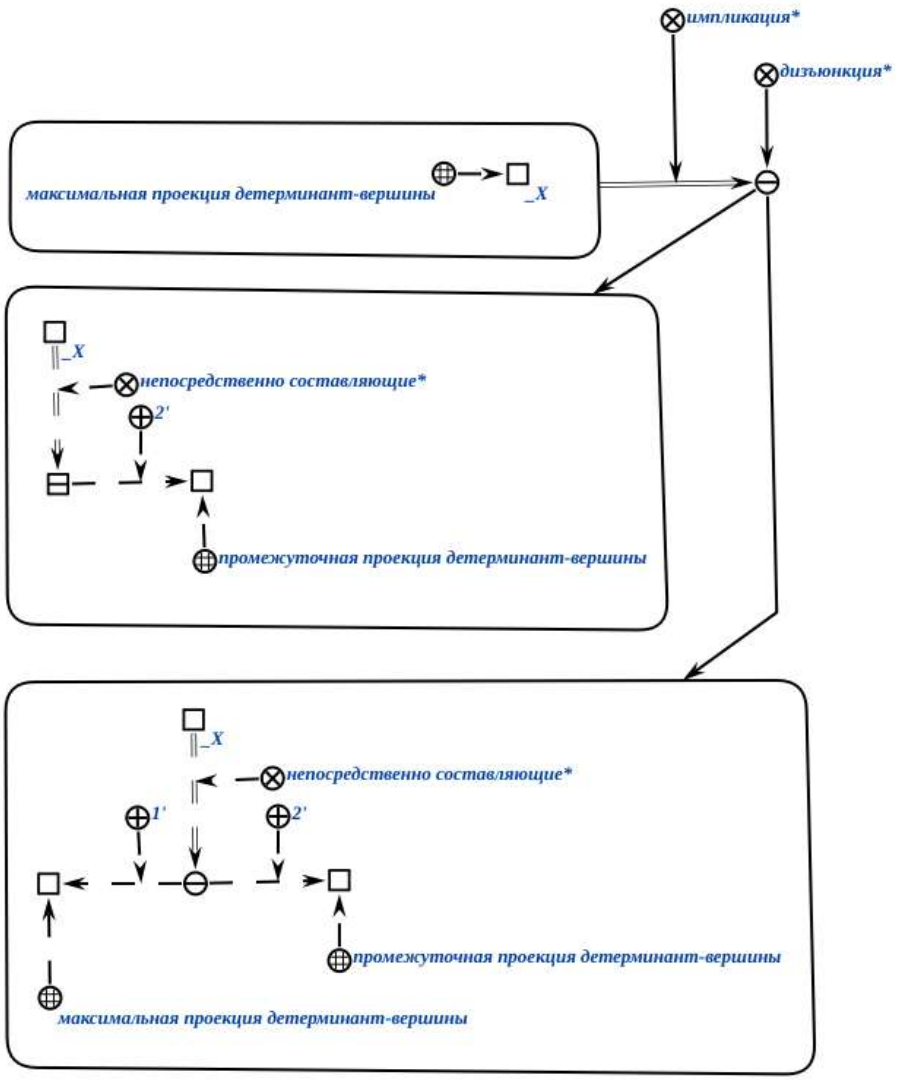
§ 2.7.2. Формализация денотационной семантики естественных языков

Формализацией *денотационной семантики естественного языка* мы будем считать множество общих правил перехода от *синтаксиса информационных конструкций естественных языков* к *смысловому представлению информации*, содержащейся в исходной *информационной конструкции*.

денотационная семантика языка специфицирует интерпретацию элементов *синтаксиса* данного *языка* и представляет собой множество формул, описывающих то, каким образом *информационным конструкциям языка* ставятся в соответствие обозначаемые ими сущности и конфигурации отношений между этими сущностями.

денотационная семантика естественных языков должна обладать свойством композициональности — то есть интерпретация всего высказывания должна выводиться из интерпретации отдельных его частей. Таким образом, необ-

SCg-текст. Иллюстрация правила структуры синтаксической группы
=



ходимо предоставить формальное описание интерпретации элементов *синтаксиса естественного языка*, представленных в предыдущем разделе, а также описание правил совмещения интерпретации отдельных элементов для получения *смысла* всего высказывания.

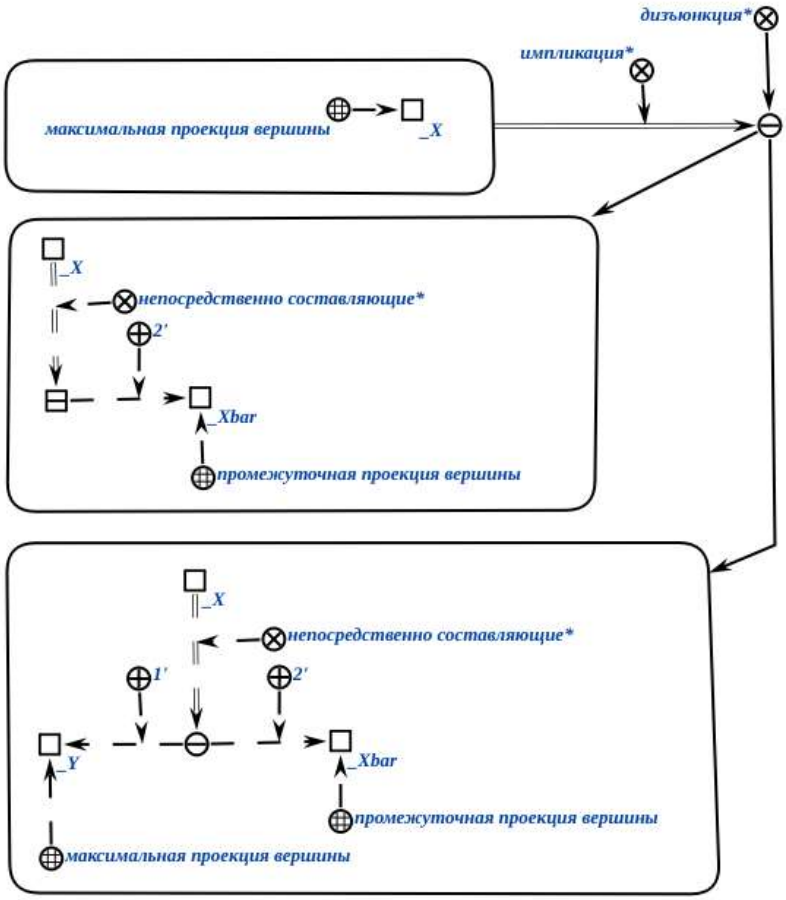
В данной главе мы предлагаем вариант формализации *денотационной семантики естественных языков* в рамках *Технологии OSTIS*, для составления которой использовались стандартные положения формальной семантики (см. *Heim I..Seman iGG-1998bk*, *Winter Y.Eleme oFS-2016bk*, *Portner P.H..FormaStER-2008bk*).

Рассмотрим примеры правил, реализующих *денотационную семантику языка*. Приведенные ниже правила должны применяться последовательно и позволяют получить *смысл* текста *естественного языка* по его синтаксической структуре, "поднимаясь" по дереву *составляющих* от *вершин* к *максимальным проекциям*.

На *SCg-текст. Правило интерпретации вершины группы прилагательного и вершины именной группы* приведено правило, по которому происходит интерпретация вершин *именной группы* и *группы прилагательного*. Смыслом таких вершин является класс, например: *прилагательному* "черный" соответствует множество черных объектов, а *существительному* "кот" — множество котов.

На *SCg-текст. Правило интерпретации максимальной проекции вершины именной группы* приведено правило, по которому происходит интерпретация *именной группы*, максимальная проекция которой включается в себя также группу прилагательного. Как говорилось выше, для применения данного правила необходимо предварительное применение правила, представленного на *SCg-текст. Правило интерпретации вершины группы прилагательного и вершины именной группы*.

SCg-текст. Правило спецификатора
=



Смыслом таких конструкций является класс, являющийся результатом *пересечения* классов, полученных в результате интерпретации *вершин групп прилагательного* и *именной группы* по отдельности. Например: *черный кот* — множество черных котов, пересечение множества котов и черных объектов.

На *SCg-текст. Правило интерпретации максимальной проекции вершины глагольной группы, содержащей непереходный глагол* приведено правило, по которому происходит интерпретация *глагольной группы*. Необходимость включения в посылку правила всей ветки глагольной группы объясняется ее необходимостью для определения типа *глагола* — данное правило предназначено для интерпретации непереходных *глаголов*. *смыслом* такой конструкции является класс *действий*.

На *SCg-текст. Правило интерпретации максимальной проекции вершины группы детерминанта* приведено правило, по которому происходит интерпретация *группы детерминанта* с неопределенным артиклем. Смыслом такой конструкции является существование элемента класса, являющегося *смыслом* входящей в состав данной *группы детерминанта именной группы*.

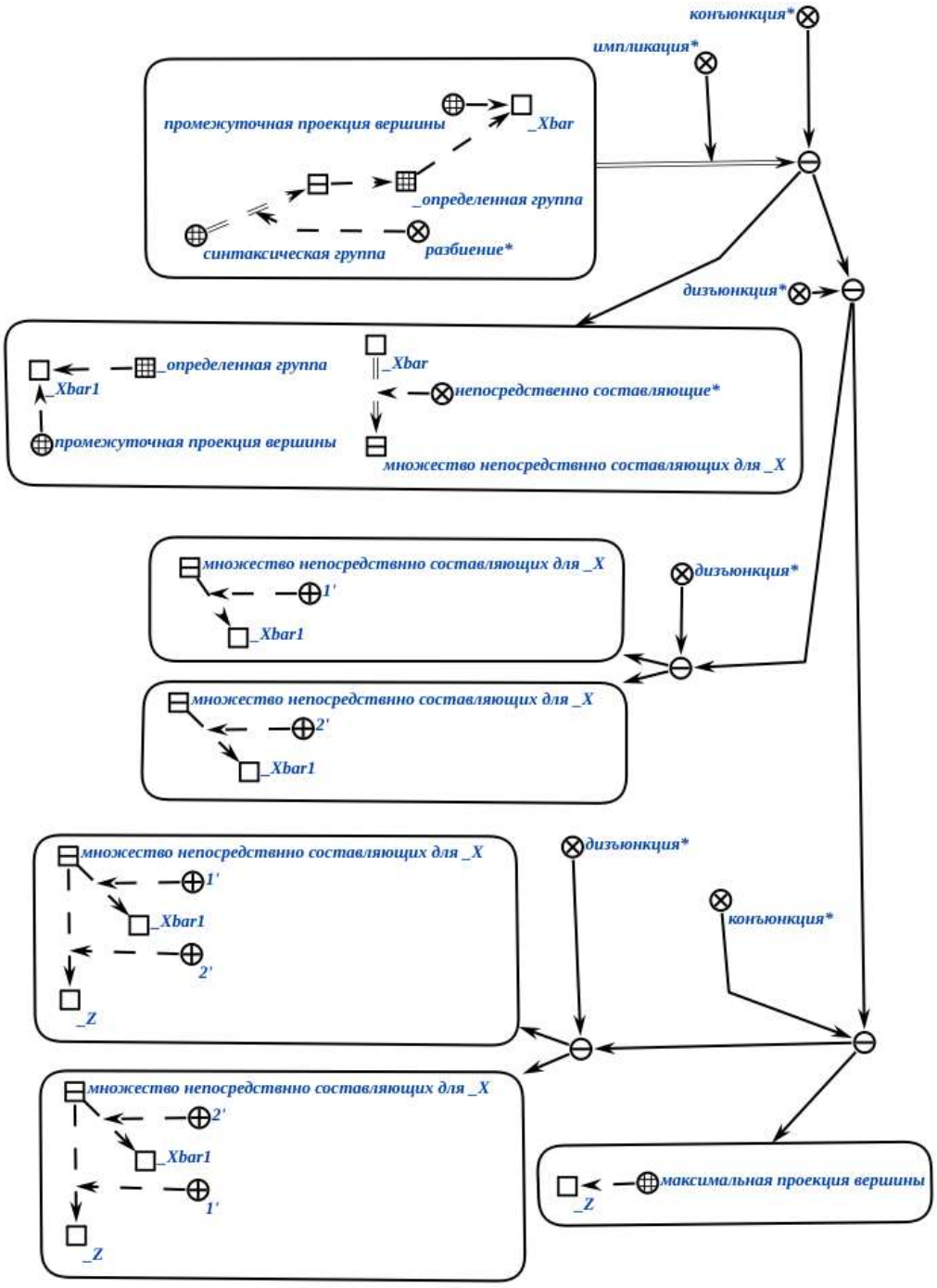
На *SCg-текст. Правило интерпретации промежуточной проекции вершины временной группы* приведено правило, по которому происходит интерпретация *промежуточной проекции вершины временной группы*, состоящей из вспомогательного *глагола* и полнозначного глагола. Вспомогательный *глагол* в данном случае задет класс действий по времени (является ли оно запланированным, выполняемым, уже выполненным и так далее).

На *SCg-текст. Правило интерпретации максимальной проекции вершины временной группы* приведено правило, по которому происходит интерпретация *максимальной проекции вершины временной группы* на основе полученного на предыдущем шаге *смысла промежуточной проекции вершины временной группы* и *смысла максимальной проекции группы детерминанта*.

На *SCg-текст. Правило интерпретации предложения с переходным глаголом* приведено правило, по которому происходит интерпретация *максимальной проекции вершины группы комплементатора* на основе полученных на предыдущих шагах *смыслов* более частных конструкций. Данным правилом задается интерпретация предложения с переходным *глаголом*.

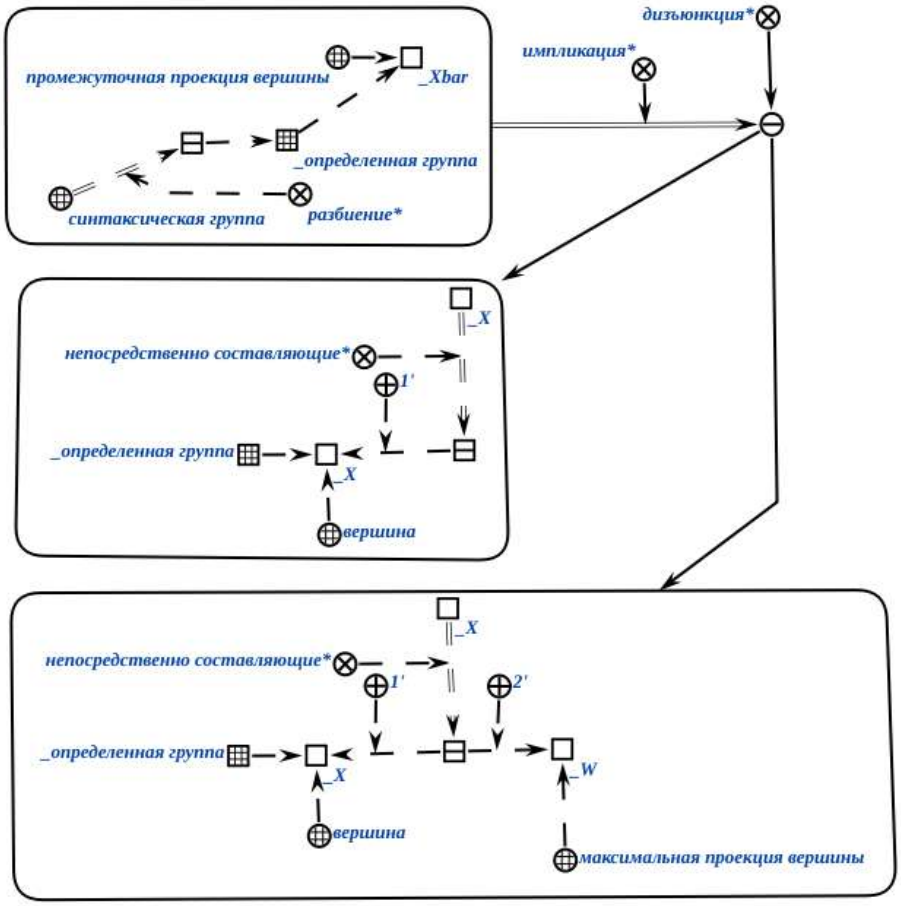
SCg-текст. Правило адъюнкта

=



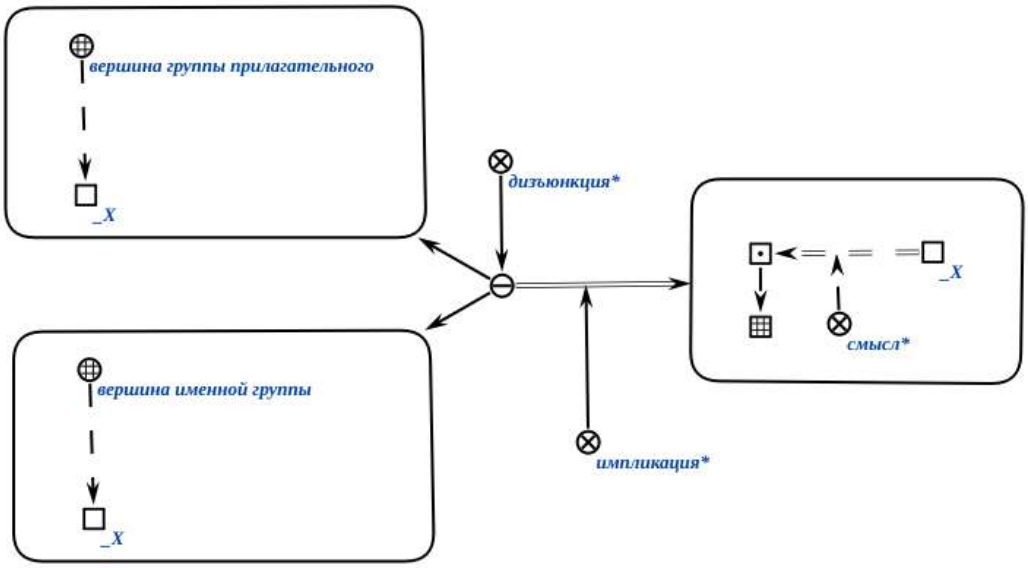
SCg-текст. Правило комплемента

=

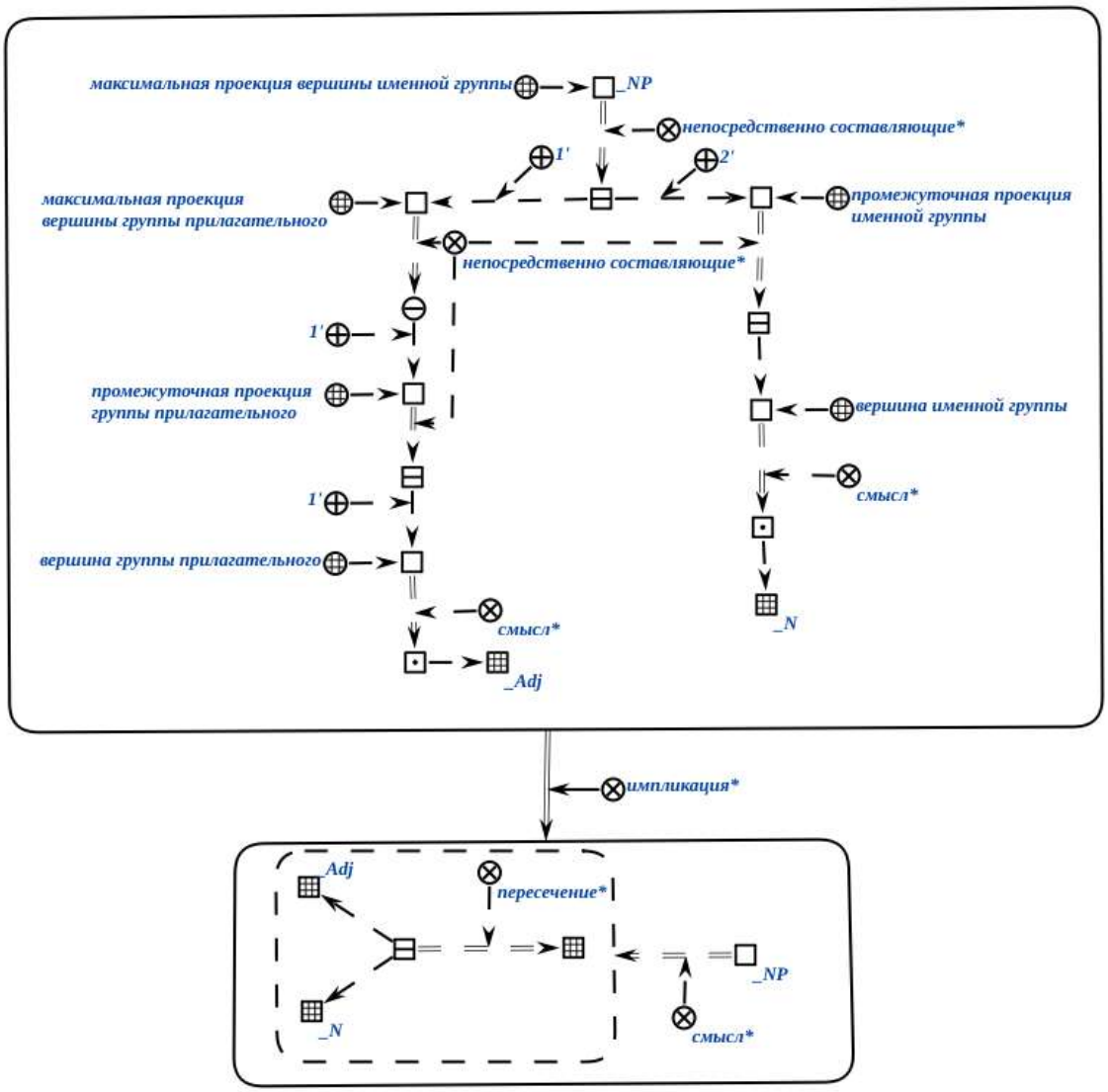


SCg-текст. Правило интерпретации вершины группы прилагательного и вершины именной группы

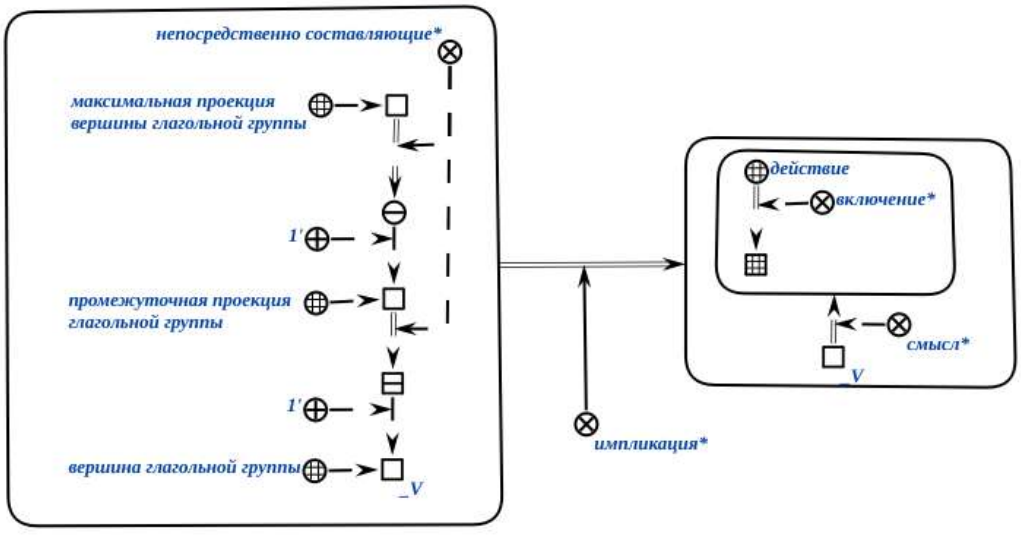
=



SCg-текст. Правило интерпретации максимальной проекции вершины именной группы
=

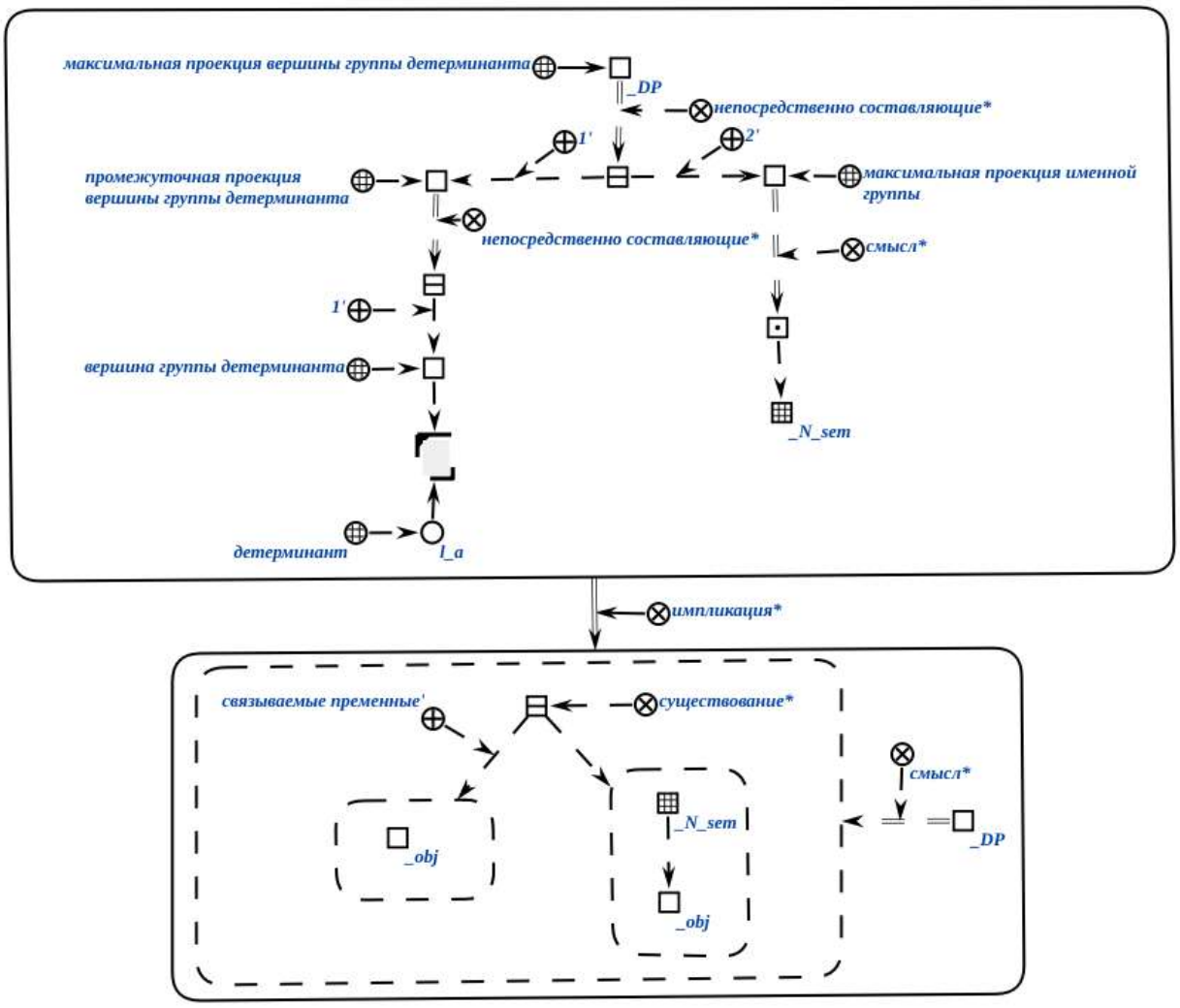


SCg-текст. Правило интерпретации максимальной проекции вершины глагольной группы, содержащей непереходный глагол
=

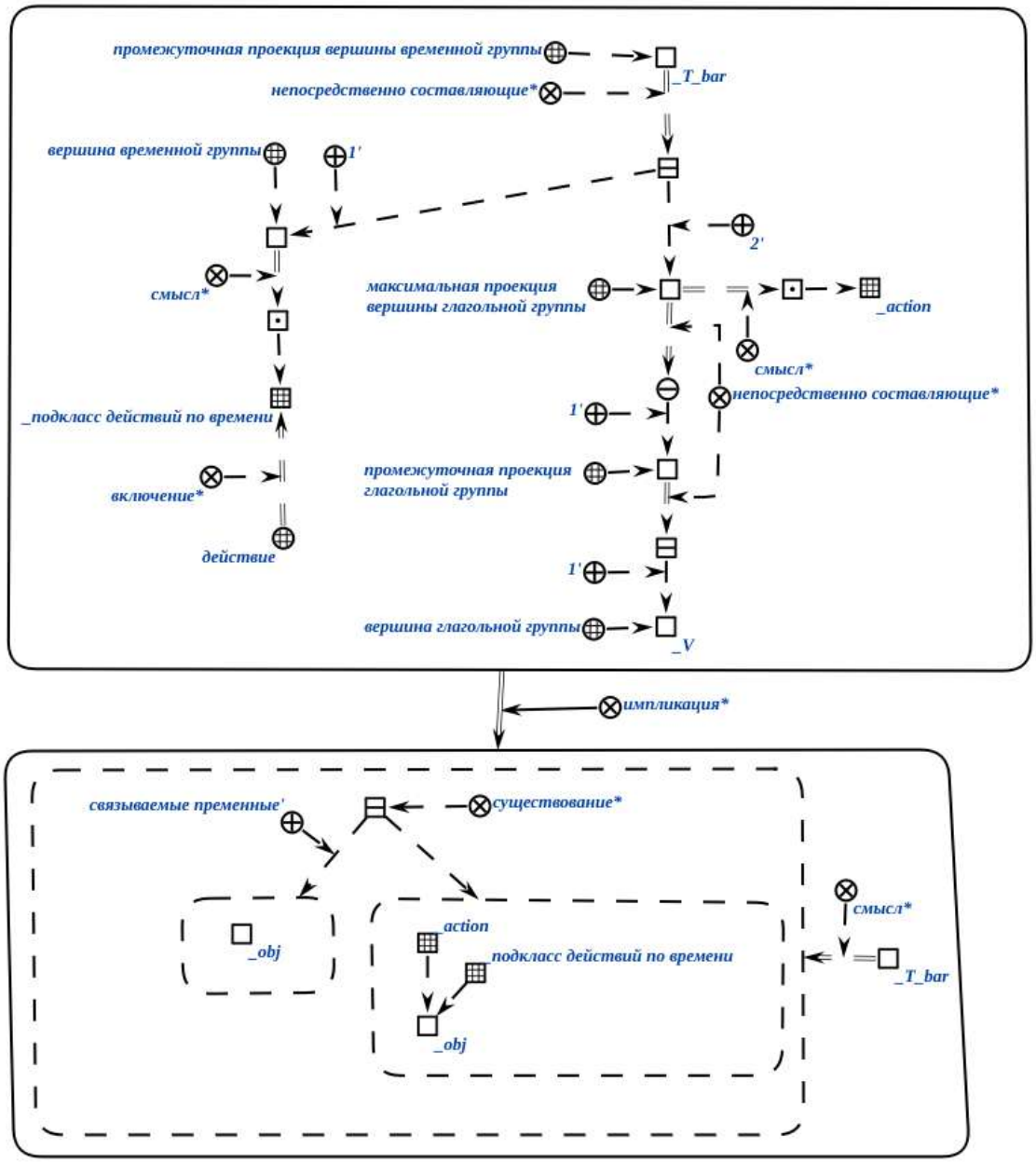


SCg-текст. Правило интерпретации максимальной проекции вершины группы детерминанта

=

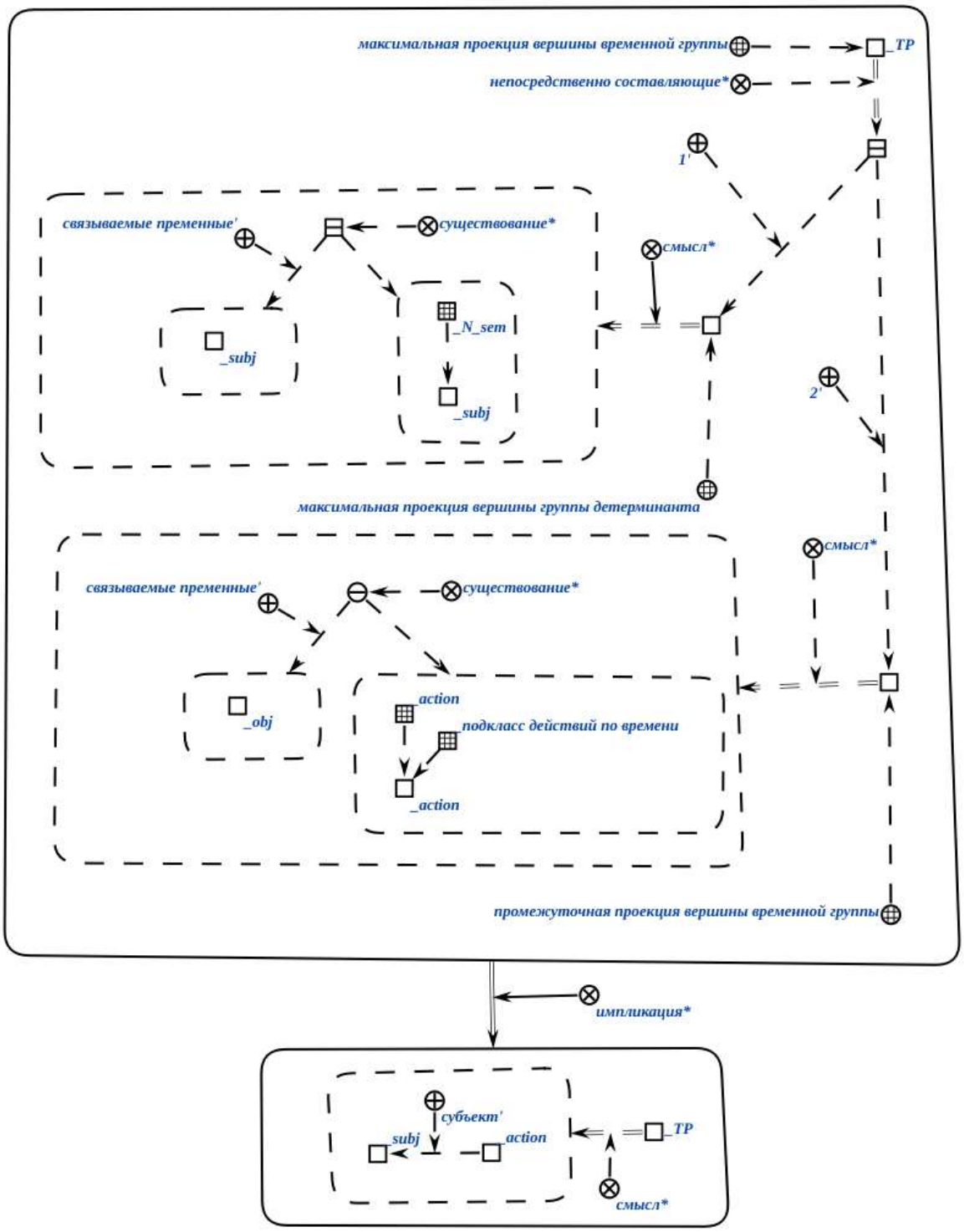


SCg-текст. Правило интерпретации промежуточной проекции вершины временной группы
=

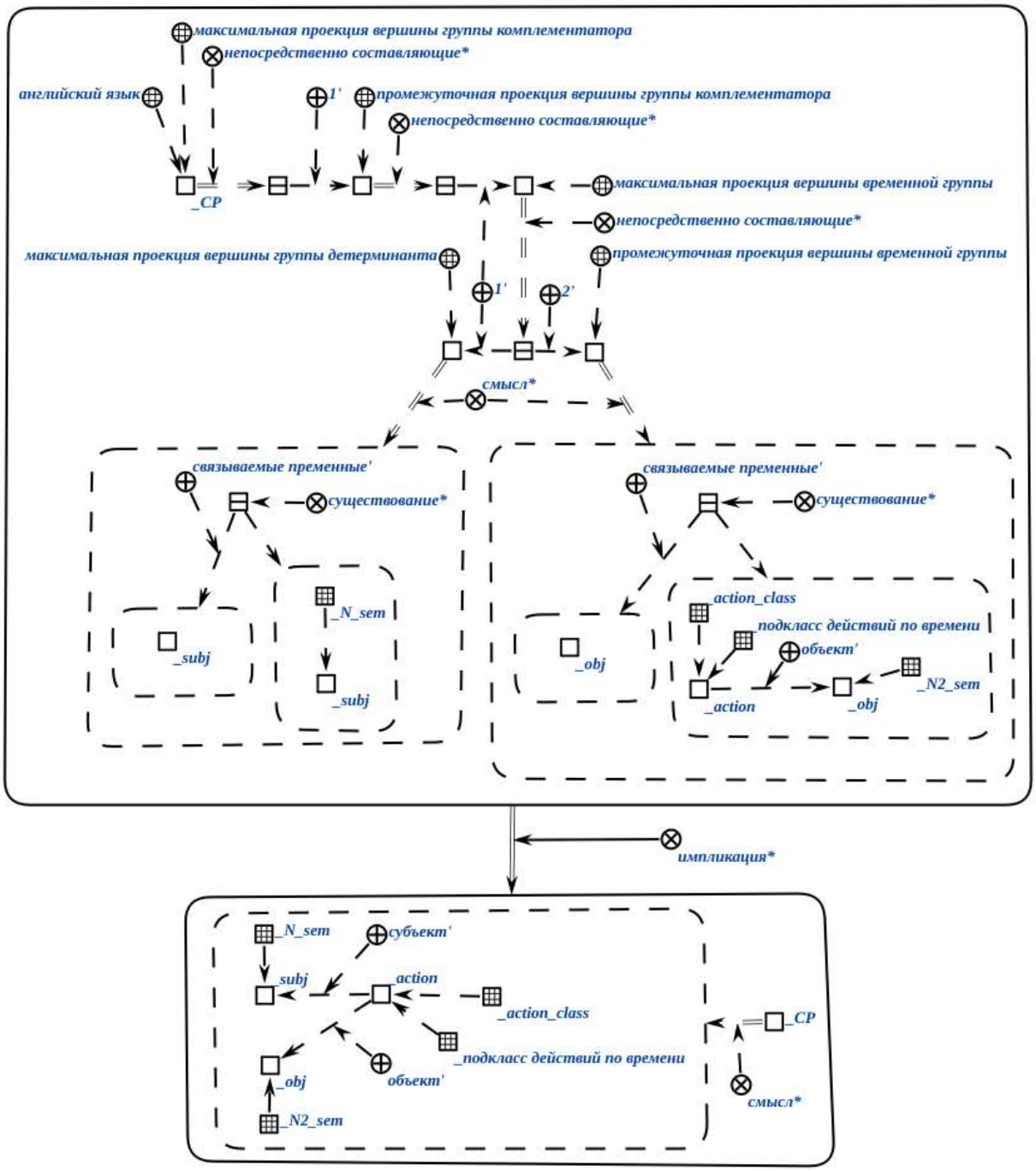


SCg-текст. Правило интерпретации максимальной проекции вершины временной группы

=



SCg-текст. Правило интерпретации предложения с переходным глаголом
=



Часть 3.

Многоагентные модели решателей задач интеллектуальных компьютерных систем нового поколения

:= [Часть 3. Многоагентные модели обработки различных видов знаний в интеллектуальных компьютерных системах нового поколения]

:= [Часть 3. Принципы конвергенции и интеграции различных методов и моделей решения задач в многоагентных интеллектуальных компьютерных системах нового поколения]

:= [Часть 3. Представление и обработка различных методов и моделей решения задач в многоагентных ostis-системах]

⇒ *аннотация**:
[Уточнение понятий действия, задачи, метода, средства, навыка и технологии. Принципы программирования в ostis-системах. Принципы разработки различных видов решателей задач в многоагентных ostis-системах. Ситуационное управление в многоагентных ostis-системах. Базовый язык программирования для ostis-систем. Принципы конвергенции и интеграции различных моделей решения задач в ostis-системах.]

⇒ *подраздел**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Глава 3.4. Язык вопросов для ostis-систем*
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Глава 3.1.

Формализация понятий действия, задачи, метода, средства, навыка и технологии

⇒ *автор**:
- *Шункевич Д.В.*
- *Ковалев М.В.*

⇒ *аннотация**:
[В главе уточнена формальная трактовка таких понятий, как *действие*, *задача*, *класс действий*, *класс задач*, *метод*, *навык*, что в совокупности позволило определить на их основе понятие *модели решения задач*. Также уточнены понятия *деятельности*, *вида деятельности* и *технологии*.]

⇒ *подраздел**:
- *§ 3.1.1. Понятие действия*
- *§ 3.1.2. Понятие задачи*
- *§ 3.1.3. Понятие класса действий и класса задач*
- *§ 3.1.4. Понятие метода*
- *§ 3.1.5. Спецификация методов и понятие навыка*
- *§ 3.1.6. Понятие класса методов и языка представления методов*
- *§ 3.1.8. Понятие модели решения задач*
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

⇒ *ключевое понятие**:
- *воздействие*
- *действие*
- *задача*
- *класс действий*
- *класс задач*
- *метод*
- *язык представления методов*
- *модель решения задач*
- *навык*
- *деятельность*
- *вид деятельности*
- *технология*

⇒ *библиографическая ссылка**:
- *Shunkevich D.V.OntolDoHPS-2022art*
- *Голенков В.В..СтандОТОП-2021кн*
- *Fayans A.M..atOntoloTTaMotS-2020art*
- *Поспелов Д.А.СитуаУТиП-1986кн*
- *Shunkevich.D.V.AgentMMaToC-2018art*

Введение в Главу 3.1.

Возможности решателя задач интеллектуальной системы в значительной степени определяются качеством ее базы знаний. Другими словами, при разработке решателей задач необходимо описывать не только *операционную семантику* решателя, то есть семейство интерпретаторов соответствующих моделей решения задач, но и *декларативную семантику* модели решения задач, то есть собственно тексты программ (не программ низкоуровневых агентов, а программ более высокого уровня, интерпретируемых соответствующим набором агентов), логические утверждения, конкретные конфигурации искусственных нейронных сетей и так далее.

В рамках данной главы формально уточняются в рамках соответствующего набора онтологий такие понятия, как *действие*, *задача*, *модель решения задач*, *метод*, *навык* и другие, на основе которых в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем* уточняется собственно модель гибридного *решателя задач ostis-системы*.

Разработка указанного семейства онтологий позволяет:
- явно связать *класс задач* и способ (метод) решения задач данного класса;
- это, в свою очередь, позволит накапливать более сложные компоненты решателей задач и еще больше упростить их интеграцию, поскольку вместе с коллективом sc-агентов в соответствующий компонент также будут входить необходимые фрагменты базы знаний, априори согласованные с указанным коллективом sc-агентов;
- это, в свою очередь, позволит сделать средства автоматизации разработки решателей задач более интеллектуальными, в частности, позволит автоматизировать процесс подбора компонентов решателя на основе спецификации классов задач, которые должна уметь решать проектируемая интеллектуальная система;
- в дальнейшем это позволит интеллектуальной системе самостоятельно обращаться в библиотеку компонентов решателей задач (см. *§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем*) и подбирать компоненты, исходя из новых классов задач, с которыми столкнулась система, то есть позволит интеллектуальной системе самостоятельно изучать новые *навыки*;
- с другой стороны, такой подход позволит интеллектуальной системе самостоятельно подобрать комбинацию моделей решения задач для решения задач определенного класса (точнее говоря, поскольку в основу решателя положен многоагентный подход, то коллектив sc-агентов, интерпретирующих различные модели решения задач, сможет лучше определить, какие именно из sc-агентов и в каком порядке должны работать при решении конкретной комплексной задачи).

Результаты, ранее полученные авторами в данной области исследований опубликованы в таких работах, как *Shunkevich D.V.OntolDoHPS-2022art*, *Голенков В.В..СтандОТОП-2021кн*.

§ 3.1.1. Понятие действия

⇒ *ключевое понятие**:
- *воздействие*
- *действие*
- *субъект*
- *информационное действие*
- *поведенческое действие*
- *эффекторное действие*
- *рецепторное действие*
- *атомарное действие*
- *неатомарное действие*

⇒ *ключевое отношение**:
- *субъект′*
- *объект′*
- *цель**

Прежде чем говорить о моделях решения задач и решателя задач, необходимо формально уточнить понятие задачи и понятие действия, направленного на решение той или иной задачи или ее подзадач.

В рамках *Технологии OSTIS* задачу будем трактовать как формальную спецификацию некоторого действия, поэтому целесообразно вначале уточнить понятие ***действия***, которое определяется через понятие ***воздействия***. Рассмотрим спецификацию понятия *воздействие* в SCn-коде.

воздействие
- ≔ [*процесс* воздействия одной сущности (или некоторого множества *сущностей*) на другую *сущность* (или на некоторое множество других *сущностей*)]
- ≔ [*процесс*, в котором могут быть явно выделены хотя бы одна воздействующая сущность (*субъект′*) и хотя бы одна *сущность*, на которую осуществляется воздействие (*объект′*)]
- ≔ [акция]
- ⊂ *процесс*
  - ≔ [динамическая структура]

субъект
- ≔ [активная сущность]
- ⊂ *сущность*
- ≔ *часто используемый sc-идентификатор**:
  [индивид]

субъект′
:= [воздействующая сущность′]
:= [сущность, создающая *причину* изменений другой сущности (объекта)′]

объект′
:= [воздействуемая сущность′]
:= [сущность, являющаяся в рамках заданного воздействия исходным условием (аргументом), необходимым для выполнения этого воздействия′]

Каждому *воздействию* может быть поставлен в соответствие (1) некоторый *субъект′*, то есть сущность, осуществляющая *воздействие* (в частности, это может быть некоторое физическое поле), и (2) некоторый *объект′*, то есть сущность, на которую воздействие направлено. Если *воздействие* связано с *материальной сущностью*, то его объектом является либо сама эта *материальная сущность*, либо некоторая ее пространственная часть.

Поскольку *воздействия* являются частным видом *процессов*, воздействиями наследуются все свойства *процессов*. В частности, используются все *параметры*, заданные на множестве *процессов*, например, *длительность**, *момент начала процесса**, *момент завершения процесса*^. Подробнее эти отношения описываются в *Главе 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*.

Также, так как воздействие является процессом и, соответственно, представляет собой *динамическую структуру*, то и знак *субъекта воздействия′*, и знак *объекта воздействия′* являются элементами данной структуры. В связи с этим можно рассматривать отношения *субъект воздействия′* и *объект воздействия′* как *ролевые отношения*. Данный факт не запрещает вводить аналогичные *неролевые отношения*, однако это нецелесообразно.

Рассмотрим спецификацию понятия *действие* в SCn-коде.

действие
:= [*воздействие*, в котором *субъект′* осуществляет *воздействие* целенаправленно, то есть в соответствии с некоторой *целью**]
:= [целенаправленное воздействие, выполняемое одним или несколькими субъектами (кибернетическими системами) с возможным применением некоторых инструментов]
:= [акт]
:= [операция]
:= [осознанное воздействие]
:= [активное воздействие]
⊂ *воздействие*
  := [процесс, в котором могут быть явно выделены хотя бы одна воздействующая сущность (субъект воздействия′) и хотя бы одна сущность, на которую осуществляется воздействие (объект воздействия′)]
  ⊂ *процесс*
:= [целенаправленный ("осознанный") процесс, выполняемый (управляемый, реализуемый) неким субъектом]
:= [работа]
:= [процесс решения некоторой задачи]
:= [процесс достижения некоторой цели]
:= [целостный фрагмент некоторой деятельности]
:= [целенаправленный процесс, управляемый некоторым субъектом]
:= [процесс выполнения некоторого действия некоторым субъектом (исполнителем) над некоторыми объектами]

цель*
:= [целевая ситуация*]
⊂ *спецификация**
:= [описание того, что требуется получить (какая ситуация должна быть достигнута) в результате выполнения заданного (специфицируемого) действия*]

Каждое *действие*, выполняемое тем или иным *субъектом*, трактуется как процесс решения некоторой задачи, то есть процесс достижения заданной *цели** в заданных условиях, и, следовательно, выполняется целеноправленно. При этом явное указание *действия* и его связи с конкретной задачей может не всегда присутствовать в памяти. Некоторые *задачи* могут решаться определенными субъектами перманентно, например, оптимизация *базы знаний*, поиск некорректностей и так далее и для подобных задач не всегда есть необходимость явно вводить *структуру*, являющуюся формулировкой *задачи*.

Каждое *действие* может обозначать сколь угодно малое преобразование, осуществляемое во внешней среде либо в памяти некоторой *кибернетической системы*, однако в памяти явно вводятся знаки только тех *действий*, для которых есть небходимость явно хранить в памяти их спецификацию в течение некоторого времени.

При выполнении *действия* можно выделить этапы:
- построение *плана действия*, декомпозиция (детализация) действия в виде системы его *поддействий*;
- выполнение построенного плана *действия*.

Класс действий имеет разбиение по следующим признакам:
- место выполнения действия;
- функциональная сложность действия;
- многоагентность выполнения действия;
- текущее состояние действия.

Далее рассмотрим разбиение по каждому признаку.

действие
⇒ *разбиение**:
Типология классов действий по признаку места выполнения действия
= {• *информационное действие*
:= [действие, выполняемое в памяти субъекта действия]
:= [действие, выполняемое в памяти]
:= [действие кибернетической системы, направленное на обработку информации, хранимой в ее памяти]
• *поведенческое действие*
:= [действие, выполняемое во внешней среде субъекта действия]
• *рецепторное действие*
:= [сенсорное действие]
• *эффекторное действие*
}

Результатом выполнения ***информационного действия*** в общем случае является некоторое новое состояние памяти информационной системы (не обязательно *sc-памяти*), достигнутое исключительно путем преобразования информации, хранящейся в памяти системы, то есть либо посредством генерации новых знаний на основе уже имеющихся, либо посредством удаления знаний, по каким-либо причинам ставших ненужными. Следует отметить, что если речь идет об изменении состояния *sc-памяти*, то любое преобразование информации можно свести к ряду атомарных действий по генерации, удалению или изменению инцидентности *sc-элементов* относительно друг друга.

В случае ***поведенческого действия*** результатом его выполнения будет новое состояние внешней среды. Очень важно отметить, что под внешней средой в данном случае понимаются также и компоненты системы, внешние с точки зрения памяти, то есть не являющиеся хранимыми в ней информационными конструкциями. К таким компонентам можно отнести, например, различные манипуляторы и прочие средства воздействия системы на внешний мир, то есть к поведенческим задачам можно отнести изменение состояния механической конечности робота или непосредственно вывод некоторой информации на экран для восприятия пользователем.

Каждое ***рецепторное действие*** обозначает класс *действий*, которые осуществляют преобразования в памяти субъекта действия под воздействием внешней среды.

Соответственно, каждое ***эффекторное действие*** обозначает класс *действий*, которые осуществляют преобразования внешней среды под воздействием памяти субъекта воздействия.

Также действия могут разбиваться по признаку их функциональной сложности.

действие
⇒ *разбиение**:
Типология классов действий по признаку функциональной сложности действия
= {• *атомарное действие*
:= [элементарное действие]
• *неатомарное действие*
:= [неэлементарное действие]
:= [неатомарное действие]
⇒ *покрытие**:
{• *легко выполнимое неатомарное действие*
• *трудно выполнимое неатомарное действие*
:= [интеллектуальное действие]
}
}

Разберем подробнее каждый элемент приведенной иерархии. ***атомарное действие*** выполняется одним индивидуальным субъектом и не включает в себя выполнение каких-либо дочерних действий.

Соответственно, ***неатомарное действие*** является действием, выполнение которого требует декомпозиции этого действия на множество его поддействий, то есть частных действий более низкого уровня. Более частные действия могут выполняться как последовательно, так и параллельно.

Декомпозиция неатомарного действия на поддействия может иметь весьма сложный иерархический вид с большим числом уровней иерархии, то есть поддействиями *неатомарного действия* могут быть также *неатомарные действия*. Уровень сложности действия можно определять (1) общим числом его поддействий и (2) числом уровней иерархии этих поддействий. Примером может служить запись одной и той же процедурной программы на языке программирования более высокого уровня и на языке программирования более низкого уровня. В данном случае атомарность действий строго определяется на уровне языка.

Темпоральные соотношения между *поддействиями* неатомарного *действия* могут быть самые различные, но в пройстейшем случае *неатомарное действие* представляет собой строгую последовательность *действий* более низкого уровня иерархии.

В состав *неатомарного действия* могут входить не только *собственно поддействия* этого *неатомарного действия*, но также и специальные *поддействия*, осуществляющие управление процессом выполнения *неатомарного действия*, и, в частности, *поддействия*, осуществляющие инициирование поддействий, передачу управления *поддействиям*.

Действие можно назвать ***легко выполнимым неатомарным действием*** в случае, когда для выполнения этого действия известен соответствующий *метод* и соответствующие этому методу исходные данные, а также (для действий, выполняемых во внешней среде) имеются в наличии все необходимые исходные объекты (расходные материалы и комплектация), а также средства (инструменты).

В свою очередь, ***трудно выполнимым неатомарным действием*** действие является тогда, когда для его выполнения в текущий момент либо неизвестен соответствующий *метод*, либо возможные *методы* известны, но отсутствуют условия их применения. Это действие декомпозируется на несколько самостоятельных поддействий, каждое из которых выявляет (локализует) противоречия (ошибки) конкретного формализуемого вида, для которого в базе знаний существует точное определение.

Признак многоагентности выполнения действия характеризует количество субъектов, выполнеяющих действие.

действие
⇒ *разбиение**:
Типология классов действий по признаку многоагентности выполнения действия
= {• *индивидуальное действие*
:= [действие, выполняемое одним субъектом (агентом)]
:= [действие, выполняемое индивидуальной кибернетической системой]
• *коллективное действие*
:= [действие, выполняемое коллективом субъектов (многоагентной системой)]
:= [действие, выполняемое коллективом кибернетических систем (коллективом субъектов)]
⊃ *действие, выполняемое коллективом людей*
⊃ *действие, выполняемое коллективом индивидуальных компьютерных систем*
⊃ *действие, выполняемое коллективом людей и индивидуальных компьютерных систем*
⊃ *действие, выполняемое Экосистемой OSTIS*
⊃ *действие, выполняемое одним человеком во взаимодействии с одной индивидуальной компьютерной системой*
}

Последним признаком разбиения классов действия является признак текущего состояния действия.

действие
⇒ *разбиение**:
Типология классов действий по признаку текущего состояния действия
= {• *планируемое действие*
:= [запланированное, но не инициированное действие]
:= [будущее действие]
• *инициированное действие*
:= [действие, ожидающее начала своего выполнения]
:= [действие, подлежащее выполнению]
• *выполняемое действие*
:= [активное действие]

:= [действие, выполняемое в текущий момент]
:= [настоящее действие]
- *прерванное действие*
  := [действие, ожидающее продолжения своего выполнения]
  := [отложенное действие]
- *выполненное действие*
  := [завершенное действие]
  := [прошлое действие]
  ⇒ *разбиение**:
    {• *успешно выполненное действие*
     • *безуспешно выполненное действие*
       ⊃ *действие, выполненное с ошибкой*
    }
- *отмененное действие*

}

Во множество ***планируемых действий*** входят *действия*, начало выполнение которых запланировано на какой-либо момент в будущем(*начало**).

Во множество ***инициированных действий*** входят *действия*, выполнение которых инициировано в результате какого-либо события. В общем случае, *действия* могут быть инициированы по следующим причинам:
- явно путем проведения соответствующей *sc-дуги принадлежности* каким-либо *субъектом (заказчиком*)*. В случае действия в *sc-памяти*, оно может быть инициировано как внутренним *sc-агентом* системы, так и пользователем при помощи соответствующего пользовательского интерфейса. При этом, спецификация действия может быть сформирована одним *sc-агентом*, а собственно добавление во множество *инициированных действий* может быть осуществлено позже другим *sc-агентом*;
- в результате того, что одно или несколько *действий*, предшествовавших данному в рамках некоторой декомпозиции, стали *прошлыми сущностями* (процедурный подход);
- в результате того, что в *памяти* системы появилась конструкция, соответствующая некоторому условию инициирования *sc-агента*, который должен выполнить данное *действие* (декларативный подход).

Следует отметить, что декларативный и процедурный подходы можно рассматривать как две крайности, использование только одной из которых не является удобным и целесообразным. При этом, например, принципы инициирования по процедурному подходу могут быть полностью сведены к набору декларативных условий инициирования, но как было сказано, это не всегда удобно и наиболее рациональным будет комбинировать оба подхода в зависимости от ситуации.

Принадлежность некоторого *действия* множеству *инициированных действий* говорит о том, что, по мнению некоторого *субъекта* (заказчика, инициатора), оно готово к выполнению и должно быть выполнено, то есть спецификация данного *действия* по мнению данного *субъекта* сформирована в степени, достаточной для решения поставленной *задачи* и существует некоторый другой *субъект* (исполнитель), который может приступать к выполнению *действия*. Однако стоит отметить, что с точки зрения *исполнителя* такая спецификация *действия* в общем случае может оказаться недостаточной или некорректной.

Во множество ***выполняемых действий*** входят *действия*, к выполнению которых приступил какой-либо из соответствующих *субъектов*. Попадание *действия* в данное множество говорит о следующем:
- рассматриваемое *действие* уже попало во множество *инициированных действий*;
- существует как минимум один *субъект*, условие инициирования которого соответствует спецификации данного *действия*.

После того, как собственно процесс выполнения завершился, *действие* должно быть удалено из множества *выполняемых действий* и добавлено во множество *выполненных действий* или какое-либо из его подмножеств.

Понятие *выполняемое действие* является неосновным, и вместо того, чтобы относить конкретные действия к данному классу, их относят к классу *настоящих сущностей*.

Во множество ***прерванных действий*** входят *действия*, которые уже были инициированы, однако их выполнение невозможно по каким-либо причинам, например в случае, когда у исполнителя в данный момент есть более приоритетные задачи.

Во множество ***выполненных действий*** попадают *действия*, выполнение которых завершено с точки зрения *субъекта*, осуществлявшего их выполнение. Таким образом, понятие *выполненного действия* является относительным, поскольку с точки зрения разных субъектов одно и то же действие может считаться выполненным или все еще выполняющимся.

В зависимости от результатов конкретного процесса выполнения, рассматриваемое *действие* может стать элементом одного из подмножеств множества *выполненных действий*.

Понятие *выполненное действие* является неосновным, и вместо того, чтобы относить конкретные *действия* к данному классу, их относят к классу *прошлых сущностей*.

Во множество ***успешно выполненных действий*** попадают *действия*, выполнение которых успешно завершено с точки зрения *субъекта*, осуществлявшего их выполнение, то есть достигнута поставленная цель, например, получены решение и ответ какой-либо задачи, успешно преобразована какая-либо конструкция и так далее. Очень важно отметить, что в общем случае выделить критерии успешности или безуспешности выполнения действий того или иного класса невозможно, поскольку эти критерии, во-первых, зависят от контекста, во-вторых, могут быть разными с точки зрения разных субъектов. Однозначно критерии успешности выполнения действий могут быть сформулированы для некоторых частных классов действий, например, классов операторов некоторого процедурного языка программирования (например, *scp-операторов*). Таким образом, понятие *успешно выполненное действие* является относительным.

Если действие было выполнено успешно, то, в случае действия по генерации каких-либо знаний, к *действию* при помощи связки отношения *результат** приписывается *sc-конструкция*, описывающая результат выполнения указанного действия. В случае, когда действие направлено на какие-либо изменения базы знаний, *sc-конструкция*, описывающая результат действия, формируется в соответствии с правилами описания истории изменений базы знаний.

В случае, когда успешное выполнение *действия* приводит к изменению какой-либо конструкции в *sc-памяти*, которое необходимо занести в историю изменений базы знаний или использовать для демонстрации протокола решения задачи, то генерируется соответствующая связка отношения *результат**, связывающая задачу и *sc-конструкцию*, описывающую данное изменение.

Во множество ***безуспешно выполненных действий*** попадают *действия*, выполнение которых не было успешно завершено с точки зрения *субъекта*, осуществлявшего их выполнение, по каким-либо причинам.

Можно выделить две основные причины, по которым может сложиться указанная ситуация:
- соответствующая *задача* сформулирована некорректно;
- формулировка соответствующей *задачи* корректна и понятна системе, однако решение данной задачи в текущий момент не может быть получено за удовлетворительные с точки зрения заказчика или исполнителя сроки.

Для конкретизации факта некорректности формулировки задачи можно выделить ряд более частных классов *безуспешно выполненных действий*, например:
- *действие, спецификация которого противоречит другим знаниям системы* (например, не выполняется неравенство треугольника);
- *действие, при спецификации которого использованы понятия, неизвестные системе*;
- *действие, выполнение которого невозможно из-за недостаточности данных* (например, найти площадь треугольника по двум сторонам);
- и другие.

Для конкретизации факта безуспешности выполнения некоторого действия в системе могут также использоваться дополнительные подмножества данного множества, при необходимости снабженные естественно-языковыми или формальными комментариями.

Во множество ***действий, выполненных с ошибкой***, попадают *действия*, выполнение которых не было успешно завершено с точки зрения *субъекта*, осуществлявшего их выполнение, по причине возникновения какой-либо ошибки, например, некорректности спецификации данного *действия* или нарушения ее целостности каким-либо *субъектом* (в случае *действия в sc-памяти*).

§ 3.1.2. Понятие задачи

⇒ *ключевое понятие**:
- *задача*
- *информационная задача*
- *поведенческая задача*
- *декларативная формулировка задачи*
- *процедурная формулировка задачи*
- *декларативно-процедурная формулировка задачи*
- *вопрос*
- *команда*

⇒ *ключевое отношение**:
- *спецификация**
- *декларативная формулировка задачи**

- *процедурная формулировка задачи**

Понятие ***задача*** будем трактовать как спецификацию некоторого действия, в рамках которой, в зависимости от ситуации, при помощи перечисленных выше отношений может быть заранее указан контекст выполнения действия, способ его выполнения, исполнитель, заказчик, планируемый результат и так далее

Рассмотрим спецификацию понятия *задача* в SCn-коде.

задача
:= [описание некоторого желаемого состояния или события либо в базе знаний, либо во внешней среде]
:= [формулировка задачи]
:= [задание на выполнение некоторого действия]
:= [постановка задачи]
:= [задачная ситуация]
:= [спецификация некоторого действия, обладающая достаточной полнотой для выполнения этого действия]

Формальное описание условия некоторой *задачи есть*, по сути, формальная *спецификация* некоторого *действия*, направленного на решение данной *задачи*, достаточная для выполнения данного *действия* каким-либо *субъектом*.

Каждая ***задача*** представляет собой спецификацию действия, которое либо уже выполнено, либо выполняется в текущий момент (в настоящее время), либо планируется (должно) быть выполненным, либо может быть выполнено (но не обязательно). В зависимости от конкретного класса задач, описываться может как внутреннее состояние самой интеллектуальной системы, так и требуемое состояние внешней среды.

Классификация задач может осуществляться по дидактическому признаку в рамках каждой предметной области, например, задачи на треугольники, задачи на системы уравнений и тому подобное.

Каждая *задача* может включать:
- факт принадлежности *действия* какому-либо частному классу *действий* (например, *действие. сформировать полную семантическую окрестность указываемой сущности*), в том числе состояние *действия* с точки зрения жизненного цикла (инициированное, выполняемое и так далее);
- описание *цели** (*результата**) *действия*, если она точно известна;
- указание *заказчика** действия;
- указание *исполнителя** *действия* (в том числе коллективного);
- указание *аргумента(-ов) действия'*;
- указание инструмента или посредника *действия*;
- описание *декомпозиции действия**;
- указание *последовательности действий** в рамках *декомпозиции действия**, то есть построение процедурного плана решения задачи. Другими словами, построение плана решения представляет собой декомпозицию соответствующего *действия* на систему взаимосвязанных между собой поддействий;
- указание области *действия*;
- указание условия инициирования *действия*;
- момент начала и завершения *действия*, в том числе планируемый и фактический, предполагаемая и/или фактическая длительность выполнения.

Некоторые задачи могут быть дополнительно уточнены контекстом — дополнительной информацией о сущностях, рассматриваемых в формулировке *задачи*, то есть описанием того, что дано, что известно об указанных сущностях.

Кроме этого, *задача* может включать любую дополнительную информацию о действии, например:
- перечень ресурсов и средств, которые предполагается использовать при решении задачи, например, список доступных исполнителей, временные сроки, объем имеющихся финансов и так далее;
- ограничение области, в которой выполняется *действие*, например, необходимо заменить одну *sc-конструкцию* на другую по некоторому правилу, но только в пределах некоторого *раздела базы знаний*;
- ограничение знаний, которые можно использовать для решения той или иной задачи, например, необходимо решить задачу по алгебре, используя только те утверждения, которые входят в курс школьной программы до седьмого класса включительно, и не используя утверждения, изучаемые в старших классах;
- и прочее.

Как и в случае с действиями, решаемыми системой, можно классифицировать *информационные задачи* и *поведенческие задачи*.

С другой стороны, с точки зрения формулировки поставленной задачи, можно выделить *декларативные формулировки задачи* и *процедурные формулировки задачи*. Следует отметить, что данные классы задач не противопоставляются друг другу, и могут существовать формулировки задач, использующие оба подхода.

задача
⊃ *вопрос*
⊃ *команда*
⊂ *знание*
⊃ *инициированная задача*
:= [формулировка задачи, которая подлежит выполнению]
⊃ *декларативная формулировка задачи*
⊃ *процедурная формулировка задачи*
⊃ *декларативно-процедурная формулировка задачи*
:= [задача, в формулировке которой присутствуют как декларативные (целевые), так и процедурные аспекты]
⊃ *задача, решаемая в памяти кибернетической системы*
⊃ *задача, решаемая в памяти индивидуальной кибернетической системы*
⊃ *задача, решаемая в общей памяти многоагентной системы*
:= [информационная задача]
:= [задача, направленная либо на генерацию или поиск информации, удовлетворяющей заданным требованиям, либо на некоторое преобразование заданной информации]
⊃ *математическая задача*

Формулировка *задачи* может не содержать указания контекста (области решения) *задачи* (в этом случае областью решения *задачи* считается либо вся *база знаний*, либо ее согласованная часть), а также может не содержать либо описания исходной ситуации, либо описания целевой ситуации. Так, например, описание целевой ситуации для явно специфицированного противоречия, обнаруженного в *базе знаний*, не требуется.

декларативная формулировка задачи представляет собой описание исходной (начальной) ситуации, являющейся условием выполнения соответствующего действия, и целевой (конечной) ситуации, являющейся результатом выполнения этого действия, то есть описание ситуации (состояния), которое должно быть достигнуто в результате выполнения планируемого действия. Другими словами, такая формулировка задачи включает явное или неявное описание:
- того, что дано, — исходные данные, условия выполнения специфицируемого действия;
- того, что требуется, — формулировка цели, результата выполнения указанного действия.

В случае ***процедурной формулировки задачи*** явно указывается характеристика действия, специфицируемого этой задачей, а именно, например, указывается:
- субъект или субъекты, выполняющие это действие;
- объекты, над которыми действие выполняется, — аргументы действия;
- инструменты, с помощью которых выполняется действие;
- момент и, возможно, дополнительные условия начала и завершения выполнения действия;
- явно указывается класс или классы, которым принадлежит каждое *действие* (включая поддействия).

При этом явно не уточняется, что должно быть результатом выполнения соответствующего действия.

Заметим, что, при необходимости, *процедурная формулировка задачи* может быть сведена к *декларативной формулировке задачи* путем трансляции на основе некоторого правила, например, определения класса действия через более общий класс.

Частными видами задач являются *вопрос* и *команда*.

вопрос
:= [запрос]
⊂ *задача, решаемая в памяти кибернетической системы*
:= [непроцедурная формулировка задачи на поиск (в текущем состоянии базы знаний) или на генерацию знания, удовлетворяющего заданным требованиям]
⊃ *вопрос — что это такое*
⊃ *вопрос — почему*
⊃ *вопрос — зачем*
⊃ *вопрос — как*
:= [каким способом]
:= [запрос метода (способа) решения заданного (указываемого) вида задач или класса задач либо плана решения конкретной указываемой задачи]
:= [задача, направленная на удовлетворение информационной потребности некоторого субъекта-заказчика]

команда
:= [инициированная задача]
:= [спецификация инициированного действия]

Следует отметить, что, наряду с приведенной предельно общей классификацией задач, по сути отражающей классы задач с точки зрения их формулировки, должна существовать классификация задач с точки зрения их семантики, то есть с точки зрения сути специфицируемого действия. За основу такой классификация можно взять классификацию, представленную в работе *Fayans A.M..atOntoloTTaMotS-2020art*.

В рамках же данной работы, как уже было сказано, наибольший интерес представляют задачи, решаемые в sc-памяти.

Сужение бинарного ориентированного отношения *спецификация** (быть спецификацией*), связывающее *действия* с *задачами*, которые решаются в результате выполнения этих *действий*, не является взаимно однозначным. Каждому *действию* может соответствовать несколько формулировок *задач*, которые с разной степенью детализации или с разных аспектов специфицируют указанное *действие*. Кроме того, интерпретация разных формулировок семантически одной и той же *задачи* в общем случае приводит к разным *действиям*, решающим эту *задачу*. Подчеркнем, что разные, но семантически эквивалентные формулировки *задач* считаются формулировками формально разных *задач*.

Примером сужения отношения *спецификация** являются отношения *декларативная формулировка задачи** и *процедурная формулировка задачи**.

Декларативная формулировка задачи включает в себя:
- связку отношения *цель**, связывающую специфицируемое действие с описанием целевой ситуации;
- само описание целевой ситуации;
- связку отношения *исходная ситуация**, связывающую специфицируемое действие с описанием исходной ситуации;
- непосредственно описание исходной ситуации;
- указание контекста (области решения) задачи.

При этом указание и описание исходной ситуации может отсутствовать.

Процедурная формулировка задачи включает в себя указание:
- *класса действий*, которому принадлежит специфицируемое *действие*, а также
- *субъекта* или субъектов, выполняющих это действие (с дополнительным указанием роли каждого участвующего субъекта);
- *объекта* или объектов, над которыми осуществляется действие (с указанием "роли" каждого такого объекта);
- используемых материалов;
- используемых инструментов (инструментальных средств);
- дополнительных темпоральных характеристик специфицируемого действия (сроки, длительность);
- приоритета (важности) специфицируемого действия;
- если описываемое действие не является атомарным в текущем контексте, то декомпозиции описываемого действия на поддействия и этих поддействий на еще более простые поддействия, вплоть до атомарных действий.

Для некоторых *отношений*, заданных на множестве *действий*, вводятся аналогичные *отношения*, заданные на множестве *задач*, соответствующих этим *действиям*. От *отношений*, связывающих некоторые сущности, несложно перейти к *отношениям*, связывающим *спецификации** этих сущностей.

§ 3.1.3. Понятие класса действий и класса задач

⇒ *ключевое понятие**:
- *класс действий*
- *класс атомарных действий*
- *класс легковыполнимых неатомарных действий*
- *класс задач*

С точки зрения организации процесса решения задач более важными являются не столько понятия *действия* и *задачи*, сколько понятия *класса действий* и *класса задач*, поскольку именно для них разрабатываются соответствующие алгоритмы выполнения и способы решения.

Класс действий определим как максимальное множество аналогичных (похожих в определенном смысле) действий, для которого существует (но не обязательно известный в текущий момент) по крайней мере один **метод** (или средство), обеспечивающий выполнение любого действия из указанного множества действий.

класс действий
⇐ *семейство подклассов**:
действие
:= [множество однотипных действий]

⊃ *класс атомарных действий*
⊃ *класс легковыполнимых неатомарных действий*

Каждому выделяемому *классу действий* соответствует по крайней мере один общий для них *метод* выполнения этих *действий*. Это означает то, что речь идет о семантической "кластеризации" множества *действий*, то есть о выделении *классов действий* по признаку семантической близости (сходства) *действий*, входящих в состав выделяемого *класса действий*. При этом прежде всего учитывается аналогичность (сходство) *исходных ситуаций* и *целевых ситуаций* рассматриваемых *действий*, то есть аналогичность *задач*, решаемых в результате выполнения соответствующих *действий*. Поскольку одна и та же *задача* может быть решена в результате выполнения нескольких разных *действий*, принадлежащих разным *классам действий*, следует говорить не только о *классах действий* (множествах аналогичных действий), но и о ***классах задач*** (о множествах аналогичных задач), решаемых этими *действиями*. Так, например, на множестве *классов действий* заданы следующие *отношения*:

- *отношение*, каждая связка которого связывает два разных (непересекающихся) *класса действий*, осуществляющих решение одного и того же *класса задач*;
- *отношение*, каждая связка которого связывает два разных *класса действий*, осуществляющих решение разных *классов задач*, один из которых является *надмножеством* другого.

Кроме класса действий также выделяется понятие ***класса атомарных действий***, то есть множества атомарных действий, указание принадлежности которому является необходимым и достаточным условием для выполнения этого действия. Множество всевозможных атомарных действий, выполняемых каждым субъектом, должно быть разбито на классы атомарных действий.

Принадлежность некоторого *класса действий* множеству ***класс атомарных действий*** фиксирует факт того, что при указании всех необходимых аргументов принадлежности *действия* данному классу достаточно для того, чтобы некоторый субъект мог приступить к выполнению этого действия.

При этом, даже если *класс действий* принадлежит множеству ***класс атомарных действий***, не запрещается вводить более частные *классы действий*, для которых, например, заранее фиксируется один из аргументов.

Если конкретный ***класс атомарных действий*** является более частным по отношению к *действиям в sc-памяти*, то это говорит о наличии в текущей версии системы как минимум одного *sc-агента*, ориентированного на выполнение действий данного класса.

Кроме того, целесообразно также ввести понятие *класса легковыполнимых неатомарных действий*, то есть множества *сложных действий*, для которых известен и доступен по крайней мере один ***метод***, интерпретация которого позволяет осуществить полную (окончательную, завершающуюся атомарными действиями) декомпозицию на поддействия каждого сложного действия из указанного выше множества. Кроме того, целесообразно также ввести понятие *класса легковыполнимых неатомарных действий*, то есть множества *сложных действий*, для которых известен и доступен по крайней мере один ***метод***, интерпретация которого позволяет осуществить полную (окончательную, завершающуюся атомарными действиями) декомпозицию на поддействия каждого сложного действия из указанного выше множества.

Принадлежность некоторого *класса действий* множеству ***класс легковыполнимых неатомарных действий*** фиксирует факт того, что даже при указании всех необходимых аргументов принадлежности *действия* данному классу недостаточно для того, чтобы некоторый *субъект* приступил к выполнению этого действия, и требуются дополнительные уточнения.

В свою очередь, под ***классом задач*** будем понимать множество задач, для которого можно построить обобщенную формулировку задач, соответствующую всему этому множеству задач. Каждая *обобщенная формулировка задач соответствующего класса*, по сути, есть не что иное, как строгое логическое определение указанного класса задач.

класс задач
⇐ *семейство подмножеств**:
задача

Конкретный класс действий может быть определен как минимум двумя способами.

класс действий
⇒ *разбиение**:
{• ***класс действий, однозначно задаваемый решаемым классом задач***
:= [*класс действий*, обеспечивающих решение соответствующего *класса задач* и использующих при этом любые, самые разные *методы* решения задач этого класса]
• ***класс действий, однозначно задаваемый используемым методом решения задач***
}

§ 3.1.4. Понятие метода

⇒ *ключевое понятие**:
- *метод*
- *стратегия решения задач*
- *стратегия решения информационных задач*

⇒ *ключевое отношение**:
- *эквивалентность задач**
- *подметод**

Под методом будем понимать описание того, как может быть выполнено любое или почти любое (с явным указанием исключений) действие, принадлежащее соответствующему классу действий.

Формально, *метод* — это спецификация решения задачи какого-то класса *Голенков В.В..СтандОТОП-2021кн*. В состав спецификации каждого класса задач входит описание способа "привязки" метода к исходным данным конкретной задачи, решаемой с помощью этого метода.

метод
:= [метод решения соответствующего класса задач, обеспечивающий решение любой или большинства задач указанного класса]
:= [обобщенная спецификация выполнения действий соответствующего класса]
:= [обобщенная спецификация решения задач соответствующего класса]
:= [программа решения задач соответствующего класса, которая может быть как процедурной, так и декларативной (непроцедурной)]
:= [знание о том, как можно решать задачи соответствующего класса]
⊂ *знание*
∈ *вид знаний*
:= [способ]
:= [знание о том, как надо решать задачи соответствующего класса задач (множества эквивалентных (однотипных, похожих) задач)]
:= [метод (способ) решения некоторого (соответствующего) класса задач]
:= [информация (знание), достаточная для того, чтобы решить любую *задачу*, принадлежащую соответствующему *классу задач*, с помощью соответствующей *модели решения задач*]

В состав спецификации каждого *класса задач* входит описание способа "привязки" *метода* к исходным данным конкретной *задачи*, решаемой с помощью этого *метода*. Описание такого способа "привязки" включает в себя:
- набор переменных, которые входят как в состав *метода*, так и в состав *обобщенной формулировки задач соответствующего класса*, и значениями которых являются соответствующие элементы исходных данных каждой конкретной решаемой задачи;
- часть *обобщенной формулировки задач* того класса, которому соответствует рассматриваемый *метод*, являющихся описанием условия применения этого *метода*.

Сама рассматриваемая "привязка" *метода* к конкретной *задаче*, решаемой с помощью этого *метода*, осуществляется путем поиска в *базе знаний* такого фрагмента, который удовлетворяет условиям применения указанного *метода*. Одним из результатов такого поиска и является установление соответствия между указанными выше переменными используемого *метода* и значениями этих переменных в рамках конкретной решаемой *задачи*.

Другим вариантом установления рассматриваемого соответствия является явное обращение (вызов, call) соответствующего *метода* (программы) с явной передачей соответствующих параметров. Но такое не всегда возможно, так как при выполнении процесса решения конкретной *задачи* на основе декларативной спецификации выполнения этого действия нет возможности установить:
- когда необходимо инициировать вызов (использование) требуемого *метода*;
- какой конкретно *метод* необходимо использовать;
- какие параметры, соответствующие конкретной инициируемой *задаче*, необходимо передать для "привязки" используемого *метода* к этой *задаче*.

Процесс "привязки" *метода* решения *задач* к конкретной *задаче*, решаемой с помощью этого *метода*, можно также представить как процесс, состоящий из следующих этапов:
- построение копии используемого *метода*;
- склеивание основных (ключевых) переменных используемого *метода* с основными параметрами конкретной решаемой *задачи*.

В результате этого, на основе рассматриваемого *метода*, используемого в качестве образца (шаблона), строится спецификация процесса решения конкретной задачи — процедурная спецификация (*план*) или декларативная.

Заметим, что *методы* могут использоваться даже при построении *планов* решения конкретных *задач* в случае, когда возникает необходимость многократного повторения неких цепочек *действий* при априори неизвестном количестве таких повторений. Речь идет о различного вида **циклах**, которые являются простейшим видом процедурных *методов* решения задач, многократно используемых (повторяемых) при реализации *планов* решения некоторых *задач*.

Очевидно также, что одному *классу действий* может соответствовать несколько *методов*.

Термин “метод”, таким образом, будем считать синонимичным термину “программа” в обобщенном понимании этого термина.

метод
- ⇒ *часто используемый идентификатор**:
  [программа]
- := [программа выполнения действий некоторого класса]
- ⊃ *непроцедурная программа*
- ⊃ *процедурная программа*
  - := [обобщенный план]
  - := [обобщенный план выполнения некоторого класса действий]
  - := [обобщенный план решения некоторого класса задач]
  - := [обобщенная спецификация декомпозиции любого действия, принадлежащего заданному классу действий]
  - ⊂ *алгоритм*

Рассмотрим более подробно понятие процедурной программы (процедурного метода). Каждая ***процедурная программа*** представляет собой обобщенный план выполнения *действий*, принадлежащих некоторому классу, то есть *семантическую окрестность; ключевым sc-элементом′* является *класс действий*, для элементов которого дополнительно детализируется процесс их выполнения.

Входным параметрам *процедурной программы* в традиционном понимании соответствуют аргументы, соответствующие каждому *действию* из *класса действий*, описываемого данной *процедурной программой*. При генерации на основе данной программы *плана* выполнения конкретного *действия* из данного класса эти аргументы принимают конкретные значения.

Каждая *процедурная программа* представляет собой систему описанных действий с дополнительным указанием для действия:
- либо *последовательности выполнения действий** (передачи инициирования), когда условием выполнения (инициирования) действий является завершение выполнения одного из указанных или всех указанных действий;
- либо события в базе знаний или внешней среде, являющегося условием его инициирования;
- либо ситуации в базе знаний или внешней среде, являющейся условием его инициирования.

Понятие метода позволяет определить отношение *эквивалентность задач** на множестве задач. Задачи являются эквивалентными в том и только в том случае, если они могут быть решены путем интерпретации одного и того же *метода* (способа), хранимого в памяти кибернетической системы. Некоторые *задачи* могут быть решены разными *методами*, один из которых, например, является обобщением другого. Таким образом, на множестве методов можно также задать ряд отношений.

Отметим, что понятие *метода* фактически позволяет локализовать область решения задач соответствующего класса, то есть ограничить множество знаний, которых достаточно для решения задач данного класса определенным способом. Это, в свою очередь, позволяет повысить эффективность работы системы в целом, исключая число лишних действий.

отношение, заданное на множестве методов
- ∋ *подметод**
  - := [подпрограмма*]
  - := [быть методом, использование которого (обращение к которому) предполагается при реализации заданного метода*]
  - ⇔ *следует отличать**:
    *частный метод**
    - := [быть методом, обеспечивающим решение класса задач, который является подклассом задач, решаемых с помощью заданного метода*]

В литературе, посвященной построению решателей задач, встречается понятие ***стратегии решения задач***. Определим его как метаметод решения задач, обеспечивающий либо поиск одного релевантного известного метода, либо синтез целенаправленной последовательности акций применения в общем случае различных известных методов.

стратегия решения задач
$\subset$ *метод*
$\supset$ *стратегия решения информационных задач*

Можно говорить об универсальном метаметоде (универсальной стратегии) решения задач, объясняющем всевозможные частные стратегии. В частности, можно говорить о нескольких глобальных ***стратегиях решения информационных задач*** в базах знаний. Пусть в базе знаний появился знак инициированного действия с формулировкой, соответствующей информационной цели, то есть цели, направленной только на изменение состояния базы знаний. И пусть текущее состояние базы знаний не содержит контекст (исходные данные), достаточный для достижения указанной выше цели, то есть такой контекст, для которого в доступном пакете (наборе) методов (программ) имеется метод (программа), использование которого позволяет достичь указанной выше цели. Для достижения такой цели, контекст (исходные данные) которой недостаточен, существует три подхода (три стратегии):

- декомпозиция (сведение изначальной цели к иерархической системе и/или подцелей (и/или подзадач) на основе анализа текущего состояния базы знаний и анализа того, чего не хватает в базе знаний для использования того или иного метода).

  При этом наибольшее внимание уделяется методам, для создания условий использования которых требуется меньше усилий. В конечном счете мы должны дойти (на самом нижнем уровне иерархии) до подцелей, контекст которых достаточен для применения одного из имеющихся методов (программ) решения задач;
- генерация новых знаний в семантической окрестности формулировки изначальной цели с помощью любых доступных методов в надежде получить такое состояние базы знаний, которое будет содержать нужный контекст (достаточные исходные данные) для достижения изначальной цели с помощью какого-либо имеющегося метода решения задач;
- комбинация первого и второго подходов.

Аналогичные стратегии существуют и для построения плана решения задач, решаемых во внешней среде.

§ 3.1.5. Спецификация методов и понятие навыка

$\Rightarrow$ *ключевое понятие**:
- *навык*
- *активный навык*
- *пассивный навык*

$\Rightarrow$ *ключевое отношение**:
- *операционная семантика метода**
- *денотационная семантика метода**

Каждый конкретный *метод* рассматривается нами не только как важный вид спецификации соответствующего класса задач, но также и как объект, который и сам нуждается в спецификации, обеспечивающей непосредственное применение этого метода. Другими словами, метод является не только спецификацией (спецификацией соответствующего класса задач), но и объектом спецификации. Важнейшим видом такой спецификации является указание *операционной семантики метода*.

операционная семантика метода*
$\subset$ *спецификация**
$:=$ [семейство методов, обеспечивающих интерпретацию заданного метода*]
$:=$ [формальное описание интерпретатора заданного метода*]
$\Rightarrow$ *второй домен**:
операционная семантика метода
$\supset$ ***полное представление операционной семантики метода***
$:=$ [представление *операционной семантики метода*, доведенное (детализированное) до уровня всех *спецификаций атомарных действий*, выполняемых в процессе интерпретации соответствующего *метода*]

денотационная семантика метода*
$\subset$ *спецификация**
$:=$ [описание системы понятий, которые используются в рамках данного метода*]

Отношение *денотационная семантика метода** связывает *метод* и формальное описание системы понятий (фрагмент *логической онтологии* соответствующей *предметной области*), которые используются (упоминаются) в рамках данного метода. Это необходимо для того, чтобы гарантировать однозначность трактовки одного и того же понятия в рамках метода и остальной части базы знаний, что особенно актуально при заимствовании метода из *библиотеки многократно используемых компонентов решателей задач*. Важно отметить, что тот факт, что какие-либо понятия используются в рамках метода, не означает, что формальная запись их определений является частью данного метода. Например, в состав метода, позволяющего решать задачи на вычисление площади треугольника, будут входить различные формулы расчета площади треугольника, но не будут входить сами определения понятий "площадь", "треугольник" и так далее, поскольку при наличии априори верных формул эти определения не будут непосредственно использоваться в процессе решения задачи. В то же время формальные определения указанных понятий будут входить в состав декларативной семантики данного метода.

Объединение *метода* и его операционной семантики, то есть информации о том, каким образом должен интерпретироваться данный *метод*, будем называть ***навыком***.

навык
:= [умение]
:= [объединение *метода* с его исчерпывающей спецификацией — *полным представлением операционной семантики метода*]
:= [метод, интерпретация (выполнение, использование) которого полностью может быть осуществлено данной кибернетической системой, в памяти которой хранится указанный метод]
:= [метод, который данная кибернетическая система умеет (может) применять]
:= [метод + метод его интерпретации]
:= [умение решать соответствующий класс эквивалентных задач]
:= [метод плюс его операционная семантика, описывающая то, как интерпретируется (выполняется, реализуется) этот метод, и являющаяся одновременно операционной семантикой соответствующей модели решения задач]
⇒ *разбиение**:
{• *активный навык*
:= [самоинициирующийся навык]
• *пассивный навык*
}

Таким образом, понятие *навыка* является важнейшим понятием с точки зрения построения решателей задач, поскольку объединяет в себе не только денотационную часть описания способа решения класса задач, но и операционную.

Навыки могут быть *пассивными навыками*, то есть такими *навыками*, применение которых должно явно инициироваться каким-либо агентом, либо *активными навыками*, которые инициируются самостоятельно при возникновении соответствующей ситуации в базе знаний. Для этого в состав *активного навыка*, помимо *метода* и его операционной семантики, включается также *sc-агент*, который реагирует на появление соответствующей ситуации в базе знаний и инициирует интерпретацию *метода* данного *навыка*. Такое разделение позволяет реализовывать и комбинировать различные подходы к решению задач, в частности, *пассивные навыки* можно рассматривать в качестве способа реализации концепции интеллектуального пакета программ.

§ 3.1.6. Понятие класса методов и языка представления методов

⇒ *ключевое понятие**:
• *класс методов*
• *язык представления методов*
⇒ *ключевое отношение**:
• *метод заданного языка представления методов**

Как действия и задачи, методы могут быть классифицированы по различным классам. Будем называть ***классом методов*** множество методов, для которых можно унифицировать представление (спецификацию) этих методов.

класс методов
⇐ *семейство подклассов**:
метод
:= [множество методов, для которых можно унифицировать представление (спецификацию) этих методов]
:= [множество всевозможных методов решения задач, имеющих общий язык представления этих методов]
:= [множество методов, для которых задан язык представления этих методов]

∋ *процедурный метод решения задач*
  ⊃ *алгоритмический метод решения задач*
∋ *непроцедурный метод решения задач*
  ⊃ *логический метод решения задач*
  ⊃ *продукционный метод решения задач*
  ⊃ *функциональный метод решения задач*
    ⊃ *искусственная нейронная сеть*
      ≔ [класс методов решения задач на основе искусственных нейронных сетей]
    ⊃ *генетический "алгоритм"*
≔ [множество методов, основанных на общей онтологии]
≔ [множество методов, представленных на одинаковом языке]
≔ [множество методов решений задач, которому соответствует специальный язык (например, sc-язык), обеспечивающий представление методов из этого множества]
≔ [множество методов, которому ставится в соответствие отдельная модель решения задач]

Каждому конкретному *классу методов* взаимно однозначно соответствует *язык представления методов*, принадлежащих этому (специфицируемому) *классу методов*. Таким образом, спецификация каждого *класса методов* сводится к спецификации соответствующего *языка представления методов*, то есть к описанию его синтаксической, денотационной и операционной семантики. Примерами *языков представления методов* являются все *языки программирования*, которые, в основном, относятся к подклассу *языков представления методов* — к *языкам представления методов обработки информации*. Но сейчас все большую актуальность приобретает необходимость создания эффективных формальных языков представления методов выполнения действий во внешней среде кибернетических систем. Без этого комплексная автоматизация *Поспелов Д.А.СитуаУТиП-1986кн*, в частности, в промышленной сфере, невозможна.

Таких специализированных языков может быть выделено целое множество, каждому из которых будет соответствовать своя модель решения задач (то есть свой интерпретатор).

Под *языком представления методов* будем подразумевать формальный язык, (1) знаковыми конструкциями которого являются соответствующие методы, для которых существуют общие правила построения и (2) общие правила соотнесения с теми сущностями и связями между ними, которые описываются этими методами.

язык представления методов
⇒ *часто используемый идентификатор**:
  [я.п.м.]
⇒ *часто используемый идентификатор**:
  [язык программирования]
≔ [язык методов]
≔ [язык представления методов, соответствующих определенному классу методов]
≔ [язык (например, sc-язык) представления методов соответствующего класса методов]
⊂ *язык*
⊂ *формальный язык*
≔ [формальный язык, (1) знаковыми конструкциями которого являются соответствующие методы, для которых существуют общие правила построения и (2) общие правила соотнесения с теми сущностями и связями между ними, которые описываются этими методами]
⊃ *язык представления методов обработки информации*
  ≔ [язык программирования внутренних действий кибернетической системы, выполняемых в их памяти]
  ≔ [язык представления методов решения задач в памяти кибернетических систем]
⊃ *язык представления методов решения задач во внешней среде кибернетических систем*
  ≔ [язык программирования внешних действий кибернетических систем]

Метод принадлежит языку представления методов, если он является синтаксически корректным, синтаксически целостным, семантически корректным и семантически целостным методом заданного языка представления методов.

отношение, заданное на множестве языков представления методов^
≔ [отношение, область определения которого включает в себя множество всевозможных языков представления методов]
∋ *метод заданного языка представления методов**
∋ *синтаксически корректный метод для заданного языка представления методов**
  ≔ [метод, не содержащий синтаксических ошибок для заданного языка представления методов*]
  ⊂ *синтаксически корректная знаковая конструкция для заданного языка**
∋ *синтаксически целостный метод для заданного языка представления методов**

$\subset$ *синтаксически целостная знаковая конструкция для заданного языка**
$\ni$ *семантически корректный метод для заданного языка представления методов**
$:=$ [метод, не содержащий семантических ошибок для заданного языка представления методов*]
$\subset$ *семантически корректная знаковая конструкция для заданного языка**
$\ni$ *семантически целостный метод для заданного языка представления методов**
$\subset$ *семантически целостная знаковая конструкция для заданного языка**
$:=$ [метод заданного языка представления методов, содержащий достаточную информацию для установления его истинности*]

метод заданного языка представления методов*
$:=$ [метод, принадлежащий заданному языку программирования*]
$\subset$ *текст заданного языка**
$\Rightarrow$ *второй домен**:
метод
$\Leftarrow$ *объединение**:
{• {}
$\Leftarrow$ *объединение**:
{• *синтаксически корректный метод для заданного языка представления методов**
• *синтаксически целостный метод для заданного языка представления методов**
}
• {}
$\Leftarrow$ *объединение**:
{• *семантически корректный метод для заданного языка представления методов**
• *синтаксически целостный метод для заданного языка представления методов**
}
}

§ 3.1.7. Общая классификация языков представления методов

$\Rightarrow$ *ключевое понятие**:
• *процедурный язык представления методов*
• *непроцедурный язык представления методов*

Языки представления методов в современном информационном обществе различают по их парадигмам: *процедурные*, *функциональные*, *логические*, *объектно-ориентированные* и так далее. Так, например, в методах процедурного я.п.м. решение задачи компьютером формируется в виде последовательности операторов, в методах функционального я.п.м. — указанием других методов. В логическом я.п.м. применяются высказывания, а в объектно-ориентированном — объекты.

язык представления методов
$\supset$ *язык представления методов общего назначения*
$:=$ [язык программирования общего назначения]
$\supset$ *предметно-ориентированный язык представления методов*
$:=$ [предметно-ориентированный язык программирования]
$\Rightarrow$ *разбиение**:
парадигма языка представления методов^
$=$ {• *процедурный язык представления методов*
• *непроцедурный язык представления методов*
}

процедурные языки представления методов задают вычисления как последовательность операторов (команд). Они ориентированы на компьютеры с архитектурой фон Неймана. Основные понятия процедурных я.п.м. тесно связаны с компонентами компьютера:

- переменными различных типов, которые моделируют ячейки памяти компьютера;
- операторами присваивания, которые моделируют пересылки данных между участками памяти;
- повторений действий в форме итерации, которые моделируют хранение информации в смежных ячейках памяти;
- и другое.

процедурный язык представления методов
:= [императивный язык представления методов]
⊃ *структурный язык представления методов*
  ∋ *пример′*:
    • *Fortran*
    • *Cи*
    • *Pascal*
⊃ *объектно-ориентированный язык представления методов*
  ∋ *пример′*:
    • *Java*
    • *Smalltalk*
    • *HTML*
  ⊃ *аспектно-ориентированный язык представления методов*
⊃ *скриптовый язык представления методов*
  := [склеивающий язык представления методов]

непроцедурные языки представления методов, в отличие от процедурных, задают вычисления как последовательность связанных между собой объектов. Основные понятия непроцедурных я.п.м. обычно не связаны с компонентами компьютера.

непроцедурный язык представления методов
:= [декларативный язык представления методов]
⊃ *логический язык представления методов*
  ∋ *пример′*:
    • *Prolog*
⊃ *продукционный язык представления методов*
⊃ *функциональный язык представления методов*
  := [аппликативный язык представления методов]
  ∋ *пример′*:
    • *LISP*

§ 3.1.8. Понятие модели решения задач

⇒ *ключевое понятие**:
  • *модель решения задач*
  • *денотационная семантика языка представления методов соответствующего класса*
  • *операционная семантика языка представления методов соответствующего класса*
⇒ *ключевое отношение**:
  • *модель решения задач**

По аналогии с понятием стратегии решения задач введем понятие ***модели решения задач***, которое будем трактовать как метаметод интерпретации соответствующего класса методов.

модель решения задач
⊂ *метод*
:= [метаметод]
:= [абстрактная машина интерпретации соответствующего класса методов]
:= [иерархическая система "микропрограмм", обеспечивающих интерпретацию соответствующего класса методов]
⊃ *алгоритмическая модель решения задач*
⊃ *процедурная параллельная синхронная модель решения задач*
⊃ *процедурная параллельная асинхронная модель решения задач*
⊃ *продукционная модель решения задач*
⊃ *функциональная модель решения задач*
⊃ *логическая модель решения задач*
  ⊃ *четкая логическая модель решения задач*
  ⊃ *нечеткая логическая модель решения задач*
⊃ *"нейросетевая" модель решения задач*
⊃ *"генетическая" модель решения задач*

Каждая *модель решения задач* задается:
- соответствующим классом методов решения задач, то есть языком представления методов этого класса;
- предметной областью этого класса методов;
- онтологией этого класса методов (то есть денотационной семантикой языка представления этих методов);
- операционной семантикой указанного класса методов.

Важно отметить, что для интерпретации всех моделей решения задач может быть использован агентно-ориентированный подход, рассмотренный в работе *Shunkevich.D.V.AgentMMaToC-2018art*.

спецификация*
⊃ ***модель решения задач****
  = *сужение отношения по первому домену (спецификация*; класс методов)**
  := [спецификация *класса методов**]
  := [спецификация *языка представления методов**]
  ⇒ *разбиение**:
    {• ***синтаксис языка представления методов соответствующего класса****
    • ***денотационная семантика языка представления методов соответствующего класса****
    • ***операционная семантика языка представления методов соответствующего класса****
    }

Модель решения задач ставит в соответствие некоторому классу методов синтаксис, денотационную и операционную семантику языка представления методов соответствующего класса.

денотационная семантика языка представления методов соответствующего класса
:= [онтология соответствующего класса методов]
:= [денотационная семантика соответствующего класса методов]
:= [денотационная семантика языка (sc-языка), обеспечивающего представление методов соответствующего класса]
:= [денотационная семантика соответствующей модели решения задач]
⇒ *примечание**:
  [Если речь идет о языке, обеспечивающем внутреннее представление методов соответствующего класса в ostis-системе, то синтаксис этого языка совпадает с Синтаксисом SC-кода]
⊂ *онтология*

операционная семантика языка представления методов соответствующего класса
:= [метаметод интерпретации соответствующего класса методов]
:= [семейство агентов, обеспечивающих интерпретацию (использование) любого метода, принадлежащего соответствующему классу методов]
:= [операционная семантика соответствующей модели решения задач]

Поскольку каждому *методу* соответствует *обобщенная формулировка задач*, решаемых с помощью этого *метода*, то каждому *классу методов* должен соответствовать не только определенный *язык представления методов*, принадлежащих указанному *классу методов*, но и определенный *язык представления обобщенных формулировок задач для различных классов задач*, решаемых с помощью *методов*, принадлежащих указанному *классу методов*.

§ 3.1.9. Понятие деятельности, вида деятельности и технологии

⇒ *ключевое понятие**:
  - *деятельность*
  - *вид деятельности*
  - *анализ*
  - *проектирование*
  - *разработка плана производства*
  - *производство*
  - *реинжиниринг*
  - *технология*
⇒ *ключевое отношение**:
  - *контекст**
  - *технология**

Понятие ***деятельности*** трактуется как сложный процесс, состоящий из действий, направленных на достижение нескольких разных целей (то есть целей, не связанных отношением "цель-подцель"). При этом некоторые из указанных "максимальных" целей могут достигаться с помощью одного и того же метода или одного и того же (фиксированного) семейства методов.

деятельность — это целостный процесс поведения (функционирования) одного субъекта или сообщества субъектов, осуществляемый на основе хорошо или не очень хорошо продуманной и согласованной *технологии* в последнем случае качество деятельности определяется уровнем интеллекта единоличного или коллективного субъекта, осуществляющего этот процесс.

деятельность
:= [система действий, являющаяся некоторым кластером семантически близких действий, обладающих семантической близостью, семантической связностью и семантической целостностью]
:= [трудно выполнимая семантически целостная система действий]
:= [кластер множества действий, определяемый семантической близостью этих действий]
$\subset$ *процесс*
$\supset$ *физическая деятельность*
:= [деятельность по преобразованию материальных сущностей (физических объектов)]
$\supset$ *информационная деятельность*
:= [деятельность, направленная на обработку информации]
:= [умственная деятельность]

В состав каждой конкретной *деятельности* входят также *действия*, не являющиеся *поддействиями** других *действий*, входящих в состав этой же *деятельности*. Такие "первичные" ("независимые", "самостоятельные", "автономные") *действия* для заданной *деятельности* могут инициироваться извне этой *деятельности* с помощью соответствующих инициирующих эти *действия ситуаций* или *событий*. Примерами таких инициирующих ситуаций, "порождающих" соответствующие действия, являются:

- появление в *базе знаний* каких-либо противоречий, информационных дыр, информационного мусора;
- появление в *базе знаний* описаний (информационных моделей) каких-либо нештатных ситуаций в сложном объекте управления, на которые необходимо реагировать;
- появление в *базе знаний* формулировок различного рода задач с явным указанием инициирования соответствующих действий, направленных на решение этих задач.

К числу указанных "первичных" *действий*, входящих в состав *объединенной деятельности кибернетической системы*, также относятся:

- неатомарное действие, целью которого является перманентное обеспечение комплексной *безопасности кибернетической системы*;
- неатомарное действие, целью которого является перманентное повышение качества информации (базы знаний), хранимой в памяти *кибернетической системы*;
- неатомарное действие, целью которого является перманентное повышение *качества решателя задач кибернетической системы*;
- неатомарное действие, целью которого является перманентная поддержка высокого уровня *семантической совместимости* кибернетической системы со своими партнерами.

Локализация (минимизация) *контекста* заданного действия или деятельности является важнейшим "подготовленным" этапом, обеспечивающим существенное снижение "накладных расходов" при непосредственном выполнении этого *действия* или *деятельности*.

Чаще всего *контекстом* заданного *действия* или *деятельности* является некоторая *предметная область* вместе с соответствующей ей интегрированной (объединенной) *онтологией*. Поэтому хорошо продуманная декомпозиция *базы знаний* интеллектуальной компьютерной системы на иерархическую систему *предметных областей* и соответствующих им *онтологий* имеет важное "практическое" значение, существенно повышающее качество (в частности, быстродействие) *решателя задач* интеллектуальной компьютерной системы благодаря априорному разбиению множества выполняемых *действий* (решаемых задач) по соответствующим им *контекстам*.

контекст*
:= [область исполнения действия или деятельности*]
$\Rightarrow$ *первый домен**:
(действие $\cup$ *деятельность)*

Подчеркнем, что *действие* является частью (фрагментом) *деятельности* всех участвующих в этом субъектов. Система действий выполняемых соответствующим субъектом (кибернетической системой) "скрепленное" общим контекстом и определенным набором используемых навыков и инструментов.

Имеет смысл выделять целые классы семантически целостных систем действия, для которых можно унифицировать используемые методы, информационные ресурсы и инструменты. Для обозначения таких классов вводится понятие ***вида деятельности***.

вид деятельности
⇐ *семейство подклассов**:
деятельность
:= [класс трудно выполнимых и семантически целостных систем сложных действий]
:= [класс кластеров систем действий]
:= [множество деятельностей, которые могут быть реализованы с помощью общей технологии]
∋ *анализ*
∋ *проектирование*
:= [проектная деятельность]
∋ *разработка плана производства*
∋ *производство*
:= [воспроизводство]
:= [реализация]
:= [материализация]
∋ *реинжиниринг*
:= [модификация]
:= [реконфигурация]
:= [повышение качества]
⊃ *совершенствование*
∋ *интеграция*

анализ обозначает вид дейстельности, направленной на построение (разработка, создание) спецификации (описания) основных связей и/или структуры, свойств, закономерностей, соответствующих (описываемой) сущности. Объектом анализа может быть не только материальная сущность, но и процесс, ситуация, статическая структура, внешняя информационная конструкция, знание, понятие и другие абстрактные сущности

проектирование направлено на построение (разработку) такой информационной модели (проекта) некоторой материальной сущности, которой достаточно, чтобы соответствующий индивидуальный или коллективный субъект по соответствующей технологии (то есть с помощью соответствующих методов и средств (инструментов)) смог воспроизвести (изготовить) указанную материальную сущность либо в одном экземпляре, либо в достаточно большом количестве таких экземпляров (копий), то есть воспроизвести в промышленном масштабе.

Примерами проектирования являются:
- проектирование здания;
- проектирование машиностроительной конструкции;
- проектирование микросхемы;
- проектирование ostis-системы;
- разработка системы шунтирования сердца;
- разработка такого описания сложной геометрической фигуры, которого было бы достаточно для построения изображения (рисунка) этой фигуры с помощью, например, циркуля и линейки.

разработка плана производства является видом деятельности, направленным на разработку плана производства материального объекта по заданному проекту этого объекта. Выделяют *разработку плана единичной реализации материального объекта по заданному проекту этого объекта* и *разработку плана массовой реализации материального объекта по заданному проекту этого объекта*. Примерами последнего можно назвать:
- разработка плана-графика строительства конкретного здания;
- разработка типового плана строительства зданий по заданному их типовому проекту;
- разработка типового плана операций шунтирования сердца;
- разработка алгоритма построения изображения заданной геометрической фигуры с помощью циркуля и линейки.

производство обозначает вид деятельности, направленный на воспроизводство материального объекта по заданному его проекту и плану реализации. Примерами данного вида действий являются:
- непосредственно строительство конкретного здания;
- проведение конкретной хирургической операции;
- процесс построения изображения (рисунка) геометрической фигуры с помощью циркуля и линейки.

Если классу легко выполнимых сложных действий ставится в соответствие чаще всего один *метод* и, возможно, некоторый набор инструментальных средств, используемых в этом методе, то каждому виду деятельности ставится в соответствие своя ***технология***, включающая в себя некоторый набор используемых *методов*, а также набор *инструментальных средств*, используемых в этих *методах*. Сложность здесь заключается:

- в нетривиальности организации использования всего арсенала имеющейся *технологии* для реализации (выполнения) каждой соответствующей *деятельности*;
- в трудности, а часто и в принципиальной невозможности полностью автоматизировать реализацию соответствующей *деятельности*.

Рассмотрим разницу между понятиями *действие*, *класс действий*, *деятельность* и *вид деятельности* на примере геометрии:

- действие доказательства Теоремы Пифагора;
- класс действий, направленных на построение доказательств (логических обоснований) всевозможных теорем в различных формальных теориях;
- деятельность как процесс эволюции формальной теории, являющейся формальным представлением Геометрии Евклида(в данный процесс входит и генерация гипотез в рамках Геометрии Евклида, и доказательство теорем, и выявление противоречий между высказываниями, и разрешение этих противоречий, и минимизация числа используемых определяемых понятий, и многое другое);
- вид деятельности как класс процессов, направленных на эволюцию всевозможных формальных теорий (логических онтологий), которая также включает в себя возможность коррекции этих теорий.

У каждого вида деятельности есть своя спецификация — ***технология***, которая указывается с помощью отношения *технология**. Каждая *технология* представляет собой комплекс *методов* и средств, обеспечивающих выполнение некоторого множества *действий*, входящих в состав соответствующего *вида деятельности*. Каждая *технология* задается:

- множеством методов, которое разбивается на классы методов, эквивалентных по своей операционной семантике (по набору агентов, осуществляющих интерпретацию соответствующего класса методов);
- множеством агентов, являющихся средством интерпретации методов из указанного выше множества.

С формальной точки зрения каждая технология задается ориентированной связкой, компонентами которой являются:

- знак множества используемых методов;
- знак множества используемых инструментов;
- знак множества дополнительных используемых ресурсов.

Поскольку разработка каждой конкретной *технологии* требует больших затрат, очень важно, чтобы *технологии* создавались не под конкретные *деятельности*, а для целых классов деятельностей (*видов деятельности*). При этом важно, чтобы разрабатываемые *технологии* охватывали как можно большее количество деятельностей, входящих в состав указанных *видов деятельности*. Из этого следует целесообразность конвергенции и унификации различных сфер *деятельности* для того, чтобы повысить мощность применения (использования) каждой разрабатываемой *технологии*.

Кроме того важна *совместимость технологий*, позволяющая решать *задачи*, требующие одновременного использования нескольких *технологий*, причем, в непредсказуемых сочетаниях. Очень важно также, кроме *видов деятельности*, которым соответствуют конкретные *технологии*, ввести *обобщенные виды деятельности* и построить их иерархии, явно фиксировать стандарты, которым должны соответствовать все виды соответствующего обобщенного *вида деятельности*. Это необходимо для обеспечения совместимости *технологий*. Все используемые технологии должны "пронизывать" друг друга и составлять стройную иерархическую систему совместимых технологий (сумму технологий).

Заключение к Главе 3.1.

Дальнейшее развитие представленных в данной главе онтологий предполагает формализацию классификации задач, решаемых интеллектуальными системами, унификацию описания задач и классов задач, описания целей, хода и результата решения задачи, методов решения задач, связей между классами задач и методами решения задач данного класса. Это позволит обеспечить возможность глубокой интеграции всевозможных моделей решения задач различных классов и возможность облегчить процесс интеграции новых моделей решения задач в интеллектуальную систему, а также положит основу для решения ряда проблем в области разработки гибридных решателей задач, рассмотренных в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*.

Глава 3.2.
Семантическая теория программ для ostis-систем

⇒ *автор**:
- *Зотов Н.В.*
- *Шункевич Д.В.*

⇒ *аннотация**:
[Несмотря на активное развитие и использование современных технологий и языков программирования, общей семантической теории программ, на основе которой можно было бы проектировать и разрабатывать прикладные системы, на данный момент не существует. В данной главе предлагается семантическая теория программ для ostis-систем. Глава показывает особенности представления и ключевые моменты процесса интерпретации программ в ostis-системах.]

⇒ *подраздел**:
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*
- *§ 3.2.2. Существующие онтологии языков программирования*
- *§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем*
- *§ 3.2.4. Синтаксис и семантика программ в ostis-системах*
- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*
- *§ 3.2.6. Комплекс свойств, определяющих эффективность программ в ostis-системах*

⇒ *ключевой знак**:
- *Семантическая теория программ для ostis-систем*
  ≔ [Предлагаемый нами вариант теории для проектирования языков программирования и программ для интеллектуальных компьютерных систем нового поколения]

⇒ *ключевое понятие**:
- *метод*
  ≔ [программа]
- *язык представления методов*
  ≔ [язык программирования]
- *эффективность метода*

⇒ *ключевое отношение**:
- *спецификация метода**
- *синтаксис метода**
- *денотационная семантика метода**
- *операционная семантика метода**
- *спецификация языка представления методов**
- *синтаксис языка представления методов**
- *денотационная семантика языка представления методов**
- *операционная семантика языка представления методов**

⇒ *библиографическая ссылка**:
- *Sebesta R.W.Conce oPL-2012bk*
- *Zapata J..Ontol iSEATGKA-2010art*
- *Golenkov V.V..Princ oOaAotSC-2019art*
- *Penta M..otRelatBRAaBaD-2020art*
- *Scalabrino S..ImproCRMwTF-2016art*
- *Голенков В.В..ГрафоМПОЗ-2012ст*
- *Брукс Ф.МифичЧМиКСП-2020кн*
- *Sellitto G.tUnder tIoRoPC-2020art*
- *Turner ..Towar aPLO-2007bk*
- *Chaparro O..otImpac oROoCQM-2014art*
- *Golenkov V..otCurreSaCoAI-2022art*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Eden A..Probl itOoCP-2007art*
- *Lando P..Towar aGOoCP-2007art*
- *Lando P..aOntolIitFoCP-2007art*

- *Turner ..ProgrLaTA-2014art*
- *Dijkstra E.ProgrD-1978bk*
- *Голенков В.В..СтандОТОП-2021кн*
- *Касьянов В.Н..Графы вПОВиП-2003кн*
- *Петров С.В.ГрафоГиА-1978ст*
- *Scott M.L.ProgrLP-2006bk*
- *Scott D.LattiTDTaF-1972art*
- *Орлов С.А.Теори иПЯПУ-2021кн*
- *Lu K..RethiMCfSCtSC-2022art*
- *Гулякина Н.А..Языки иТПОнО-2012ст*
- *Пивоварчик О.В.СеманМПИС-2016ст*
- *Ford B..CompoD-2019art*
- *МетасOSTIS-2022эл*
- *Пивоварчик.О.В.СемантЯОТРПООЗ-2013ст*
- *Tin ..TowarSOPL-1995art*
- *Schiitze H.tPrositL-1991art*
- *Black A.aAppro tCSS-1993th*
- *Zotov N.SemanToPiN-2022art*

Введение в Главу 3.2.

За долгий период развития компьютерных систем практически сняты аппаратные ограничения на решение различных задач. Оставшиеся ограничения отводятся на долю программного обеспечения. Прежде всего эти ограничения связаны с текущими проблемами развития программного обеспечения:

- аппаратная сложность опережает умение человечества строить программные компьютерные системы, использующее потенциальные возможности аппаратуры;
- навыки и технологии разработки программ отстают от требований, предъявляемых к разработке программ нового поколения;
- возможностям эксплуатировать существующие программы угрожает низкое качество их разработки.

Ключом к решению этих проблем является глубокое понимание и грамотное использование существующих *языков программирования* как основного инструмента для массового создания *программных компьютерных систем нового поколения*.

В данной главе акцент делается на достижение следующих результатов:

- (1) изложение классических основ, отражающих накопленный мировой опыт в области разработки и применения современных *языков программирования*;
- и (2) систематизация основных результатов в этой области в виде единой унифицированной *Семантической теории программ для интеллектуальных компьютерных систем нового поколения*, построенных по принципам *Технологии OSTIS*.

В данной главе подробно описываются проблемы текущего состояния в области *технологий* и *языков программирования*. Она посвящена базовым понятиям *теории языков программирования*, дается обзорная характеристика областей применения *языков программирования*, достаточно востребованных современным человеческим обществом, рассматриваются способы представления и интерпретации *программ* различных *языков программирования*, подробно описываются формы и содержание критериев для оценки *эффективности языков*.

§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования

В современную эру развития информационных технологий существует огромное количество *языков программирования*, каждый из которых имеет свое важное назначение в области проектирования *программных компьютерных систем*. Многообразие *языков программирования* (см. *Sebesta R.W.Conce oPL-2012bk*) и решений, созданных на них, настолько велико, что очень легко потеряться в море информации о всех аспектах применения и проектирования *языков программирования*. Кроме этого, основная проблема заключается не в количестве существующих решений в области разработки и применения современных *языков программирования*, а количестве форм (!), на которых представляются конкретные *языки программирования*. Так, *декларативные знания*, то есть знания, являющиеся, например, спецификацией какой-то программы, и *процедурные знания*, то есть знания, которые являются

программами, принадлежащими какому-то *языку программирования*, представляются совершенно различными способами, методами и средствами.

В связи со сказанным можно выделить следующие ключевые проблемы в области разработки и применения современных *языков программирования*:

- Поскольку количество *языков программирования* растет с увеличением потребности в них, то растут и потребности в описании этих *языков программирования* для дальнейшего использования и проектирования прикладных систем. Это в свою очередь требует высокого уровня качества спецификации конкретного языка: и описания *синтаксиса* и семантики конструкций этого языка, и описания средств и методов реновации инструментальных средств, обеспечивающих интерпретацию или трансляцию этого языка. То есть, с увеличением количества *языков программирования* растет не только многообразие форм представления знаний (*языков программирования*), но и количество *программных компьютерных систем* на различных формах представления знаний (см. *Zapata J..Ontol iSEATGKA-2010art*).
- Большое многообразие форм представления знаний, как говорилось выше, предоставляет большой спектр возможностей проектирования *программных компьютерных систем* на каждой из них. Получается, чтобы произвести интеграцию нескольких *программных компьютерных систем*, реализованных на разных *языках программирования*, необходимо сделать так, чтобы системы могли коммуницировать между собой на каждом из тех языков, на котором они реализованы (см. *Golenkov V.V..Princ oOaAotSC-2019art*). Так, стремление к использованию существующих программных компонентов затрудняется реализацией самих компонентов, поскольку чтобы объединить эти компоненты необходимо изменить их программный код (см. *Penta M..otRelatBRAaBaD-2020art*, *Scalabrino S..ImproCRMwTF-2016art*). Наличие многообразия форм затрудняет реализацию *совместимых интероперабельных программных компьютерных систем* (см. *Голенков В.В..ГрафоМПОЗ-2012ст*).
- С ростом сложности программного кода, уменьшается количество способных понять его смысл. Современные разработчики создают *программные компьютерные системы*, не учитывая полный ее жизненный цикл (см. *Брукс Ф.МифичЧМиКСП-2020кн*). Системы должны постоянно обновляться и совершенствоваться с развитием технологий, на которых она основана (см. *Sellitto G.tUnder tIoRoPC-2020art*). Это должно обеспечиваться хорошей документацией реализации компонентов этих систем — это снижает не только потребности в привлечении новых ресурсов и кадров, но и способствует снижению реинжиниринга *программных компьютерных систем* (см. *Penta M..otRelatBRAaBaD-2020art*, *Scalabrino S..ImproCRMwTF-2016art*).
- Полная автоматизация проектирования *программных компьютерных систем* невозможна, поскольку современные языки, на которых они проектируются не имеют свойства рефлексивности — системы не могут познавать и понимать себя и развиваться в полной мере самостоятельно. Таким образом, существующие *программные компьютерные системы* не являются как таковыми интеллектуальными, потому что не имеют необходимых им свойств (см. *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*).
- Ключом к легкому и глубокому освоению конкретного языка как основного профессионального инструмента программиста является понимание общих принципов построения и применения языков программирования (см. *Turner ..Towar aPLO-2007bk*), описываемых их общей теорией. До сегодняшнего дня, общей *Семантической теории языков программирования* до сих пор не существует, что затрудняет разработку, верификацию и использование новых и существующих *языков программирования*. Без общей теории каждый может разрабатывать принципиально общие методы и средства так, как хочется, а не так, как требуется (см. *Голенков В.В..ГрафоМПОЗ-2012ст*).
- Достижение максимума услуг и средств при минимуме затрат возможно только путем глубокого понимания принципов построения *языков программирования* за счет простоты средств и методов представления знаний. Сложное нужно сводить к простому и изъяснять простыми понятиями, не создавая дополнительной иллюзии важности (см. *Sellitto G.tUnder tIoRoPC-2020art*, *Chaparro O..otImpac oROoCQM-2014art*).

Все эти проблемы связаны и являются проблемами текущего состояния направлений развития в области *Искусственного интеллекта* (см. *Golenkov V..otCurreSaCoAI-2022art*).

Итак, для решения перечисленных проблем необходимо создавать комфортные условия для реализации *программных компьютерных систем*, семантически совместимых и интероперабельных между собой. В контексте *языков программирования* необходима общая *Семантическая теория программ для интеллектуальных компьютерных систем нового поколения*, которая:

- позволит без больших усилий и затрат интегрировать имеющиеся решения в области проектирования программ компьютерных систем (см. *Golenkov V.V..Metho aTfECoC-2019art*);
- объединит формы представления знаний декларативного и процедурного вида;
- будет иметь широкий спектр средств не только для описания синтаксиса и семантики существующих языков программирования, но и для проектирования новых аналогов;
- будет понятна не только человеку, но и машине (см. *Zapata J..Ontol iSEATGKA-2010art*);
- обозначит принципы, по которым необходимо проектировать *языки программирования нового поколения*.

К проектированию таких общих теорий, строго говоря, нужно подходить с высокой степенью важности. Проектируемые *компьютерные системы* должны всегда иметь возможности использовать те свойства, которые им начертаны. Для того, чтобы и эта теория могла быть использована как некоторая система знаний о том, как надо проектировать и использовать *языки программирования* и программы в *программных компьютерных системах*, и том, как интерпретировать их *программы*, необходимо, чтобы эта теория была описана средствами и методами, которыми проектируются эти *программные компьютерные системы*. Речь идет о том, что принципиально важным подходом к проектированию общей теории программ является *онтологический подход* (см. *Zapata J..Ontol iSEATGKA-2010art*, *Golenkov V.V..Metho aTfECoC-2019art*).

Для воплощения данных идей необходимо изучить и интегрировать опыт, накопленный в области разработки и применения *современных языков программирования*. Поэтому далее будут рассмотрены результаты других исследований в области проектирования общей теории языков программирования и программ.

§ 3.2.2. Существующие онтологии языков программирования

⇒ *ключевой знак**:
- *Недостатки современных технологий программирования и языков программирования*

В большинстве, идеи, предлагаемые в научных работах по исследованию языков программирования, безусловно являются востребованными и полезными для проектирования *программных компьютерных систем*. Так, идея о том, что языки программирования и программы, реализуемые на них, должны быть организованы в общую таксономию понятий, является основополагающей, поскольку обеспечивает наиболее качественную среду для проектирования и реализации *программных компьютерных систем*. Общая теория программ нужна не только для того чтобы описывать термины и понятия как некоторую спецификацию, используемую для проектирования *программных компьютерных систем* (что тоже немаловажно), но и для того, чтобы определять качество языков программирования и программ по таким вопросам, как: "Является ли данный язык языком программирования", "Является ли данное знание программой", "Насколько эффективна данная программа", "Какова степень интеллекта данной программной системы" и так далее. Данные идеи предложены и рассмотрены в работах Raymond Turner *Eden A..Probl itOoCP-2007art*, *Turner ..Towar aPLO-2007bk*.

До сегодняшнего дня существует большое количество аналогов онтологий языков программирования и программ. Примеры можно найти в работах (см. *Lando P..Towar aGOoCP-2007art*, *Lando P..aOntolIitFoCP-2007art*). Также стоит отметить разработанные онтологии программ в работах *Turner ..ProgrLaTA-2014art*, *Turner ..Towar aPLO-2007bk*, система понятий в которых определяется строго и однозначно на формальных языках: языках логики и языках описания грамматик формальных языков. Однако ни одна из них не является таким результатом, который можно было бы использовать при проектировании *программных компьютерных систем* без существенных проблем. Разработанные онтологии сосредотачивают в себе лишь краткое описание связанных между собой понятий, но общей картины того, как данные онтологии можно использовать в конкретных задачах, почти не видно.

Сегодня встречаются и вовсе протовоположные суждения о назначении программ и языков программирования, противоречащие формальным основам Искусственного интеллекта. *Программные компьютерные системы* должны быть не только понятны человеку, но и сами должны понимать себя, свои возможности, намерения, действия и цели, и понимать себе подобные кибернетические системы. Только таким образом человечество и результаты его деятельности в виде каких-то конкретных систем смогут работать сообща, дополняя друг друга и преумножая свои результаты *Голенков В.В..ГрафоМПОЗ-2012ст*.

В результате анализа приведенных работ можно сделать вывод о том, что:
- *общей теории программ и языков программирования*, которая могла быть задействована при решении любой прикладной задачи и представлении и реализации средств проектирования компьютерных систем, до сих пор не существует;
- унификация представления средств описания и реализации по этим описании как главный аргумент к оперированию смысловому представлению знаний, к полному взаимопониманию между *программными компьютерными системами* вовсе не рассматривается;
- *программы* и совокупности этих *программ* в виде *программных компьютерных систем* реализуются в большинстве случаев в индивидуальном порядке и плохо документируются, что усложняет их использование, интеграцию с другими программами и *программными компьютерными системами*, тестирование и совершенствование.

§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем

⇒ *ключевой знак**:
- *Принципы программирования в интеллектуальных компьютерных системах нового поколения*

Поскольку количество *языков программирования* растет с увеличением потребности в них, то растут и потребности в описании этих *языков программирования* для проектирования и разработки *программных компьютерных систем* на этих языках. То есть, с увеличением количества *языков программирования* растет не только многообразие форм представления знаний, но и количество *программных компьютерных систем* на различных формах представления знаний. Это в свою очередь требует не только качественной спецификации конкретного *языка программирования* для разработки *программ* на этом языке, но и новых требований к существующим разработчикам. В итоге, это влечет за собой появление барьеров и для создания семантически совместимых и интероперабельных *программных компьютерных систем*, и для обеспечения благоприятной среды для взаимодействия их разработчиков.

Для преодоления данных проблем нет необходимости пересматривать уже существующие решения в области разработки программного обеспечения. Необходимо создавать принципиально новые *языки программирования*, а также реализовывать *программные компьютерные системы* на них, в которых будут учтены и решены существующие проблемы. Для этого следует учитывать следующие *принципы программирования* этих систем:
- Расширение многообразия форм представления знаний происходит за счет появления новых синтаксических конструкций в *языках программирования*. Поэтому разработка *языков программирования* должна сводиться к уточнению *синтаксиса* и *семантики* уже существующих *языков программирования*. При этом все *языки программирования* должны являться подъязыками некоторого базового *языка программирования*.
- Нет необходимости в создании дополнительных языков, с помощью которых можно описывать семантику программ на *языках программирования*. Наоборот, *язык программирования*, на котором разрабатываются программы, должен позволять своими же средствами описывать *семантику программ* на этом же языке.
- Документирование *программ*, в том числе *программных компьютерных систем*, должно минимизироваться за счет этапов их качественного проектирования и разработки. Смысл конструкций *программ языков программирования* должен быть настолько ясным и понятным, чтобы использование *программ* на этом *языке программирования* не требовало дополнительных ресурсов и инструментов как и у разработчиков этих программ и систем, таких и у новых разработчиков.
- Появление новых программ должно влечь за собой к расширению *Библиотеки многократно используемых программ* и к уменьшению количества семантически эквивалетных программ. Таким образом, программы должны быть не только максимальным образом совместимыми между собой, но и открытыми для переиспользования в других *программных компьютерных системах нового поколения*.
- Полный жизненный цикл разработки новых программ должен обеспечиваться теми же средствами и *языками программирования*, на которых разрабатываются эти программы.
- Сложность программ и *программных компьютерных систем* должна сводиться к минимуму. То, что выглядит сложно, должно и может быть сделано максимально просто.
- Построение качественного коллектива *программных компьютерных систем* может быть обеспечено только совместимостью и интероперабельностью самих систем, и коллективов тех разработчиков, которые их создают.

Ключом к решению всех этих проблем является общая *Технология проектирования компьютерных систем нового поколения*, на базе которой можно построить общую *Семантическую теорию программ* (дисциплину программирования) для *интеллектуальных компьютерных систем нового поколения*, построенных по принципам *Технологии OSTIS* (см. *Dijkstra E.ProgrD-1978bk*).

Почему *Технология OSTIS* является ключом к решению описанных проблем в области проектирования и применения *языков программирования*?
- Стандарт Технологии OSTIS *Голенков В.В..СтандОТОП-2021кн* уже реализует базовые средства, необходимые для проектирования и разработки интероперабельных *программных компьютерных систем*, в основе которых лежит смысловое представление знаний. Это устраняет не только необходимость создания *онтологий верхнего уровня*, которые должны быть использованы в общей теории программ как базовые для описания понятий этой теории, но и помогает проектировать решения согласованно с другими онтологиями. В результате формируется общая слаженная картина мира, которая (1) непротиворечива, то есть согласована, (2) однозначно трактуема, (3) универсальна и, (4) самое главное, понятна для каждого.
- *Технология OSTIS* проектируется одним языком унифицированного представления знаний, называемым *SC-кодом*. Смысл *программ* и *языков программирования* понятен и однозначен тогда и только тогда, когда этот смысл описывается на одном общем языке, понятному любой *кибернетической системе*.
- *SC-код* синтаксически минимален. Для описания объектов и связей между ними используется минимальное количество знаков. В то же время многообразие этих связей сводится к многообразию знаковых конструкций. Все это обеспечивается за счет представления информации в виде графовых структур (см. *Касьянов В.Н..Графы вПОВиП-2003кн*, *Петров С.В.ГрафоГиА-1978ст*).

- SC-код не просто удобен для описания и проектирования каких-то сложных объектов — с его помощью можно проектировать и реализовывать любые *языки представления знаний*, в том числе программ, компьютерные системы и, вообще, описывать реальный мир.
- Онтологический и компонентный подходы к проектированию любых сложных объектов обеспечивают выполнение главных принципов, по которым должны проектироваться современные системы. То, что реализовано и можно использовать, нужно переиспользовать везде.

Проектирование и реализация *программы* на каком-либо *языке программирования* должна сводиться к описанию ее *синтаксиса* и *денотационной семантики* в базе знаний ostis-системы с помощью некоторой библиотеки предметных областей и онтологий программ, описываемой в рамках этой базы знаний. Для этого нужна онтология программ, которые позволили бы в достаточном объеме описывать программы на любых языках программирования в ostis-системах. Такой подход позволяет не только описывать сложноструктурированные объекты простым и понятным языком, но и позволяет унифицировать представление различных видов знаний. Тем самым, информация о программах и сами программы представляются на одном и том же языке (имеют один синтаксис), но содержательно описываются при помощи разных онтологий. Таким образом, решением всех проблем будет являться общая теория программ, однозначно соответствующей некоторой онтологии программ, с помощью которых можно было бы описывать синтаксис и денотационную семантику любых программ в ostis-системах.

Таким образом, результатом данной главы является *Предметная область и онтология программ* (далее — Предметная область и онтология методов), с помощью которой можно описывать синтаксис, денотационную и операционную семантику различных методов в ostis-системах. *Предметная область и онтология методов* является дочерней предметной областью по отношению к *Предметной области и онтологии информационных конструкций и языков*. Это означает, что она наследует все свойства исследуемых в ней понятий и отношений.

Предметная область и онтология информационных конструкций и языков
⇒ *дочерняя предметная область**:
- *Предметная область и онтология языков*
  ⇒ *дочерняя предметная область**:
  - *Предметная область и онтология естественных языков*
  - *Предметная область и онтология формальных языков*

Предметная область и онтология формальных языков
⇒ *дочерняя предметная область**:
- *Предметная область и онтология языков представления знаний*
  ⇒ *дочерняя предметная область**:
  - ***Предметная область и онтология методов***

Предметная область и онтология методов
⇒ *дочерняя предметная область**:
- *Предметная область и онтология методов ostis-систем*
  ⇒ *дочерняя предметная область**:
  - *Предметная область и онтология процедурных методов ostis-систем*

Каждая теория должна быть согласована понятийно. Несмотря на то, что в литературе сложилась разное трактование понятия *языка программирования*, должно быть одно универсальное. Для этого вместо языков программирования далее будем говорить о языках представления методов, а вместо программ этих языков программирования — о методах как знаковых конструкциях языков представления методов. Такое решение обосновывается тем, что обычно язык выступает в роли инструмента какого-то знания определенного вида, а термин *языка программирования* является вырожденным, поскольку стоит говорить не о языках, на которых что-то можно программировать, а о языках, на которых можно представлять знания определенного вида, в данном случае — знания процедурного типа. Сами термины “языка программирования” и “программы” будем считать неосновными идентификаторами понятий “языка представления методов” и “метода”, соответственно. Также это правило применяется на все понятия, используемые в данной главе и содержащие термин "метод".

Следует отметить, что общая *Семантическая теория программ в ostis-системах* не отрицает весь накопленный опыт в сфере разработки современных *технологий программирования*. Наоборот, предлагаемая в данной главе идея позволяет переиспользовать те проверенные инструменты и методы для наиболее быстрой и качественной реализации программ в сложных *программных компьютерных системах*.

§ 3.2.4. Синтаксис и семантика программ в ostis-системах

⇒ *подраздел**:
- *Пункт 3.2.4.1. Синтаксис программ в ostis-системах*
- *Пункт 3.2.4.2. Денотационная семантика программ в ostis-системах*
- *Пункт 3.2.4.3. Операционная семантика программ в ostis-системах*
- *Пункт 3.2.4.4. Синтаксис и семантика языков программирования в ostis-системах*

⇒ *ключевое понятие**:
- *спецификация метода**
- *синтаксис метода**
- *денотационная семантика метода**
- *операционная семантика метода**
- *спецификация языка представления методов**
- *синтаксис языка представления методов**
- *денотационная семантика языка представления методов**
- *операционная семантика языка представления методов**

⇒ *ключевой знак**:
- *Принципы описания синтаксиса и семантики программ в ostis-системах*
- *Язык SCP*

синтаксис и *семантика метода* составляют *спецификацию** этого *метода*. *Семантику метода* можно рассматривать в двух аспектах: как множество знаний, связанных между собой (то есть *денотационную семантику* данного *метода*), и как знание, которое может быть интерпретировано другим методом (то есть *операционную семантику* данного *метода*).

Пункт 3.2.4.1. Синтаксис программ в ostis-системах

Любой *метод* состоит из *информационных конструкций*, которые задают порядок действий в базе знаний, с помощью которых нужно перейти от исходного состояния к целевому, решив таким образом какую-то конкретную задачу. Так, например, в процедурном методе любой такой оператор представляет собой некоторую математическую функцию. Для композиции этих функций в более крупные фрагменты используются выражения и операторы. В свою очередь, линейные последовательности операторов и условные ветвления также могут быть представлены функциями, составленными из функций отдельных компонентов этих конструкций. Цикл легко описывается рекурсивной функцией, составленной из компонентов, входящих в его тело.

Синтаксис языков представления методов в ostis-системах может быть формально описан различными способами. Так, например, можно использовать метаязык Бэкуса-Наура для описания синтаксиса любых *языков представления методов*. Другими не менее известными формами представления методов являются контекстно-свободные грамматики, расширенная форма Бэкуса-Наура, синтаксические графы (см. *Sebesta R.W.Conce oPL-2012bk*, *Scott M.L.ProgrLP-2006bk*, *Scott D.LattiTDTaF-1972art*).

Однако значительно более логично и целесообразно описывать *синтаксис* других языков на универсальном *языке представления знаний — SC-коде*. Такой подход позволит ostis-системам самостоятельно понимать, анализировать и генерировать тексты указанных языков на основе принципов, общих для любых форм внешнего представления информации, в том числе нелинейных (см. *Петров С.В.ГрафоГиА-1978ст*). Таким образом, языки, написанные на *SC-коде*, имеют такой же синтаксис как и сам *SC-код*.

Пункт 3.2.4.2. Денотационная семантика программ в ostis-системах

Семантика метода разъясняет смысл *синтаксических конструкций метода*. Наиболее распространенными методами описания семантики *языков программирования* являются: денотационной, операционный, аксиоматический, алгебраический (см. *Орлов С.А.Теори иПЯПУ-2021кн*). На базе принципов Технологии OSTIS, под семантикой метода будем подразумевать объединение *денотационной* и *операционной семантики метода*.

С помощью *SC-кода* можно представлять и те языки, которые не написаны на нем. Проблема будет в том, что форма и смысл языка и его методов будут разделены, то есть будут представлены по-разному. В данном случае *SC-код* выступает мощным инструментом для интеграции спецификаций различных языков внешнего представления знаний. Однако стоит отметить, что в представлении различных форм методов, принадлежащих разным *языкам представления методов*, в рамках *Технологии OSTIS* нет необходимости. Это объясняется тем, что:

- *SC-код* является достаточно универсальным языком для представления любых видов знаний. Это означает, что различные формы алгоритма решения одной и той же задачи можно свести к минимуму. В *SC-коде* фундаментом является формальная теория, что обеспечивает универсальное представление различных видов декларативных и процедурных знаний. Так, *логические программы* можно представлять в виде *процедурных программ*, в которых в качестве операндов операторов будут не только *логические формулы* и *правила вывода*, но и другие методы, обеспечивающее интерпретацию этих *логических формул* при помощи правил вывода. Таким образом, *SC-код* можно называть не только языком унифицированного представления знания, но и языком, на котором можно решать различные классы задачи одним и тем же способом.
- Различные виды знаний в *ostis-системах*, проектируемые по принципам *Технологии OSTIS*, глубоко интегрированы между собой. Это дает не только простоту для создания этих систем на базе имеющихся языков, которые могут быть описаны на *SC-коде*, но большие возможности для создания базовых *языков программирования* для *программных компьютерных систем нового поколения* таких, как, например, *базового языка представления процедурных методов SCP*, *базового языка представления продукционных методов* и других. Современные *языки представления методов* создаются с целью упрощения описания какого-то алгоритма для быстрого и качественного решения определенного класса задач. В свою очередь, предлагаемые методики и модели позволяют проектировать *языки представления методов* для *компьютерных систем нового поколения* с помощью базовых *языков представления знаний* таким образом, чтобы сама форма представления знаний не менялась. Методы разных *языков представления методов* должны иметь одну универсальную форму представления, то есть один и тот же синтаксис, но могут давать возможности описывать и представлять разными способами *денотационную* и *операционную семантику* своих *методов* с помощью одного и того же синтаксиса.
- Проектирование новых *языков представления методов* должно сводится к их полному описанию на минимальном семействе *языков SC-кода*: *SCP*, *SCL* и других. Речь идет о том, что чтобы спроектировать новый *язык представления методов* достаточно разработать (неатомарный) метаметод на языках *SCP* и *SCL*, который будет интерпретировать методы проектируемых языков, а также описать *денотационную семантику* этих методов. *Метаметод интерпретации методов языков представления методов* можно называть интерпретатором этих языков, то есть некоторой абстрактной sc-машиной, на которой возможно выполнение методов определенного *языка представления* этих *методов*.

Пункт 3.2.4.3. Операционная семантика программ в ostis-системах

Полная *спецификация метода** кроме *денотационной семантики этого метода** должна включать *операционную семантику этого метода**, то есть формальное описание интерпретатора заданного метода. *Операционная семантика языка представления методов* описывает выполнение *метода*, составленного на данном языке, средствами виртуального компьютера. Виртуальный компьютер определяется как абстрактный автомат. Внутренние состояния этого автомата моделируют состояния вычислительного процесса при выполнении метода. Автомат транслирует исходный текст метода в набор формально определенных операций. Этот набор задает переходы автомата из исходного состояния в последовательность промежуточных состояний, изменяя значения переменных метода. Автомат завершает свою работу, переходя в некоторое конечное состояние. Таким образом, здесь идет речь о достаточно прямой абстракции возможного использования языка представления методов. *операционная семантика языка* описывает смысл метода путем выполнения его операторов на простой машине-автомате. Изменения, происходящие в состоянии машины при выполнении данного оператора, определяют смысл этого оператора.

Операционная семантика конкретного *метода* сводится к описанию *метаметода*, который его интерпретирует, верифицирует и так далее.

метаметод
⊂ *метод*
:= [метод, значениями параметров которого являются другие методы]

операционная семантика метода
∋ *метаметод интерпретации**
∋ *метаметод верификации и оценки качества**

Отношение *метаметод интерпретации** представляет собой *класс sc-связок* между *sc-связкой*, обозначающей множество *методов*, и sc-узлом, обозначающим *метод*, который способен произвести интерпретацию заданного множества *методов*. Отношение *метаметод верификации и оценки качества** представляет собой класс sc-связок между *sc-связкой*, обозначающей множество *методов*, и sc-узлом, обозначающим метод, который способен произвести верификацию и оценку качества заданного множества *методов*.

В рамках *Технологии OSTIS* таких метаметодов может быть большое разнообразие. Каждый из них может состоять из множества атомарных и неатомарных подметодов. Это могут быть как метаметоды, интерпретирующие методы определенных *языков представления методов*, так и метаметоды, верифицирующие и анализирующие качество этих методов. В том числе метаметоды могут производить операции над другими метаметодами.

метаметод интерпретации методов базовых языков представления методов
⇒ *класс подметодов**:
- *метаметод интерпретации методов Языка SCP*
- *метаметод интерпретации методов Языка SCL*
- *метаметод интерпретации методов языка представления продукционных методов*
- *метаметод интерпретации методов языка представления функциональных методов*
- *метаметод интерпретации методов языка представления нейросетей*
- *метаметод интерпретации методов языка представления генетических алгоритмов*

метаметод верификации и оценки качества методов базовых языков представления методов
⇒ *класс подметодов**:
- *метаметод верификации и оценки качества методов Языка SCP*
- *метаметод верификации и оценки качества методов Языка SCL*
- *метаметод верификации и оценки качества методов языка представления продукционных методов*
- *метаметод верификации и оценки качества методов языка представления функциональных методов*
- *метаметод верификации и оценки качества методов языка представления нейросетей*
- *метаметод верификации и оценки качества методов языка представления генетических алгоритмов*

Так, например, при реализации методов в ostis-системах метаметодами будут являться *интепретатор Языка SCP*, а также интепретаторы, реализованные непосредственно на *Языке SCP*.

Понятие *синтакиса*, *денотационной* и операционной *семантики языков представления методов* сводятся к понятию синтаксиса, денотационной и операционной семантики вообще любого языка.

Пункт 3.2.4.4. Синтаксис и семантика языков программирования в ostis-системах

Понятно, что для использования *языка представления методов* следует описать каждую конструкцию языка в отдельности, а также ее применение в совокупности с другими конструкциями. В языке существует множество различных конструкций, точное определение которых необходимо как программисту, применяющему язык, так и разработчику компилятора для этого языка. Программисту эти знания позволяют прогнозировать вычисления, производимые операторами метода. Разработчику описания конструкций необходимы для создания правильной реализации компилятора.

Описание формальной модели *языка представления методов* можно задать его *спецификацией*. *спецификация языка представления методов** содержит описание *синтаксиса*, *денотационной*, *операционной семантики языка представления методов*.

спецификация языка представления методов*
⊃ *отношение, заданное на множестве (язык представления методов)**
⇒ *разбиение**:
{• *синтаксис языка представления методов**
⊂ *синтаксис языка**
≔ [теория правильно построенных информационных конструкций, принадлежащих заданному языку представления методов]
• *денотационная семантика языка представления методов**
⊂ *денотационная семантика языка**
≔ [обобщенная формулировка классов задач, решаемых с помощью данного языка представления методов*]
• *операционная семантика языка представления методов**
⊂ *операционная семантика языка**
≔ [перечень обобщенных агентов, обеспечивающих интерпретацию методов заданного языка представления методов*]
≔ [семейство методов интерпретации текстов данного языка представления методов*]
≔ [формальное описание интерпретатора заданного языка представления методов*]
}

§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах

Текущее состояние в области проектирования и разработки программного обеспечения говорит о том, что разработчики больше стремятся автоматизировать разработку методов на конкретных языках представления методов, чем обеспечить себя инструментальными обучающими средствами их проектирования, в том числе проектирования новых *языков представления методов*. Это приводит к следующим проблемам:

- В то время, как количество разработчиков, понимающих код какой-то сложной программной системы, уменьшается, требования к этой системе растут все быстрее и быстрее. Зачастую, разработчики сложных программных систем сами не в состоянии объяснить логику работы этих систем. По этой причине необходимо создавать инструментальные средства, которые будут позволять автоматизировать документирование программных систем (см. *Lu K..RethiMCfSCtSC-2022art*).
- Для обучения новых разработчиков навыкам работы с программными системы и их разработки необходимо привлекать ресурсы экспертов, понимающих принципы работы этих программных систем. Проблема решается разработкой справочной системы, которая будет позволять не только обучать пользователя тому, как проектировать методы решения задачи и программные системы на основе этих методов, но и указывать на пробелы в смежных дисциплинах, необходимых для достижения качественных результатов всей своей деятельности.
- В инженерии часто разработчики проектируют и разрабатывают решения, которые уже когда-то были созданы другими специалистами. Таким образом, получаются функционально эквивалентные методы решения задач, а то, и вовсе, программные системы, решающие схожие проблемы. Ключом к решению данной проблемы является проектирование семантически мощной *библиотеки многократно используемых методов решения различных задач*.

Таким образом, одной *Семантической теории программ* недостаточно. Кроме нее, для перманетного и беспрепятственного проектирования и разработки *методов* различного класса необходимо разрабатывать:

- интеллектуальную систему поддержки проектирования и разработки методов, упомянутую в работах (см. *Гулякина Н.А..Языки иТПОнО-2012ст* , *Пивоварчик О.В.СеманМПИС-2016ст*), которая будет не только помогать разработчику верифицировать разрабатываемый метод, но и подсказывать способы его разработки;
- семантически мощную библиотеку многократно используемых компонентов (см. *Ford B..CompoD-2019art*) для быстрого поиска существующих методов решения задач и их применения для решения других более комплексных задач.

Потенциальная система должна быть частью общего инструментального средства разработки интеллектуальны компьютерных систем нового поколения — *Метасистемы OSTIS* (см. *МетасOSTIS-2022эл*) — и может состоять из следующих компонентов:

- интеллектуальной help-системы по семантической теории программ;
- интеллектуальной help-системы по библиотеке многократно используемых методов решения задач,
- интеллектуальной help-системы по комплексу инструментальных средств проектирования методов решения задач,
- интеллектуальной help-системы по методике обучения проектированию различных методов решения задач.

Каждый компонент должен содержать:

- справочную подсистему,
- подсистему мониторинга и анализа деятельности разработчика методов решения задач,
- подсистему управления обучением.

Каждая из подсистем взаимодействует с другими подсистемами, а также может функционировать автономно.

Справочная подсистема является консультантом-экспертом в области *Семантической теории программ*, который может ответить на любой вопрос новичка или опытного пользователя. Каждая из таких систем может становиться индивидуальным помощников в обучении новых специалистов — персональным ostis-ассистентом.

§ 3.2.6. Комплекс свойств, определяющих эффективность программ в ostis-системах

язык представления методов можно определить множеством показателей, характеризующих отдельные его свойства. Возникает задача введения меры для оценки степени приспособленности языка представления методов к выполнению возложенных на него функций — *критериев эффективности* (см. *Орлов С.А.Теори иПЯПУ-2021кн*). Критерии эффективности методов приводятся на основе частных показателей эффективности этих методов (показателей качества). Способ связи между частными показателями определяет вид критерия эффективности.

эффективность метода

⇒ *свойство-предпосылка**:

- *легкость чтения и понимания метода*
- *легкость представления метода*
- *стоимость метода*
- *общий объем задач, решаемых при помощи данного класса методов*
- *многообразие видов задач, решаемых при помощи данного класса методов*
- *надежность метода*

легкость чтения метода должна способствовать легкому выделению основных понятий каждой части метода без обращения к его спецификации.

легкость чтения и понимания метода
⇒ *свойство-предпосылка**:
- *простота синтаксиса языка представления методов*
- *ортогональность информационных конструкций языка представления методов*
- *структурированность потока управления в методе*

язык представления методов должен предоставить *простой* набор *информационных конструкций*, которые могут быть использованы в качестве базисных элементов при создании методов. Сильное воздействие на простоту оказывает *синтаксис языка*: он должен прозрачно отражать семантику конструкций, исключать двусмысленность и неоднозначность толкования.

ортогональность информационных конструкций языка представления методов означает, что любые возможные комбинации различных *информационных конструкций* будут осмысленными, без неожиданного поведения, возникающего в результате взаимодействия конструкций или контекста использования.

Порядок передач управления между операторами метода, то есть *поток управления*, должен быть удобен для чтения и понимания человеком.

легкость создания метода отражает удобство языка для представления этого метода в конкретной предметной области.

легкость представления метода
⇒ *свойство-предпосылка**:
- *простота синтаксиса языка представления методов*
- *естественность синтаксиса языка представления методов*
- *ортогональность информационных конструкций языка представления методов*
- *полнота и точность спецификации языка представления методов*
- *согласованность и целостность спецификации языка представления методов*

синтаксис метода должен способствовать легкому и прозрачному отображению в нем алгоритмических структур предметной области. *синтаксис языка представления методов* должен быть не только *простым*, но и *естественным*, и поддерживать *ортогональность* информационных конструкций языка.

легкость представления нового метода обеспечивается *полной и точной, согласованной и целостной спецификацией* соответствующего языка. То есть необходимо достаточное количество *информационных конструкций* в этом языке для того чтобы представить конкретный *метод*. При этом *спецификация языка* должна быть согласованной и целостной чтобы представлять на ней непротиворечивые *методы*.

стоимость метода языка представления методов складывается из нескольких составляющих:

общая стоимость метода
⇒ *свойство-предпосылка**:
- *стоимость применения метода*
- *стоимость интерпретации метода*
- *стоимость создания, тестирования и использования метода*
- *стоимость сопровождения метода*
- *согласованность и целостность спецификации языка представления методов*

стоимость применения метода во многом зависит от структуры *языка представления методов*. Язык, требующий многочисленных проверок синтаксических типов во время применения метода, будет препятствовать быстрой работе программы.

размер стоимости интерпретации метода зависит от возможностей используемого метаметода интерпретации. Чем совершеннее методы оптимизации, тем дороже стоит интерпретация. Размер стоимости создания,

тестирования и использования метода зависит от используемого метаметода верификации и оценки качества этого метода.

Многочисленные исследования показывают, что значительную часть стоимости используемого метода составляет не стоимость его разработки, а *стоимость его сопровождения* (см. *Брукс Ф.МифичЧМиКСП-2020кн*). Связывая сопровождение методов с другими их характеристиками, следует выделить, прежде всего, зависимость от читабельности, поскольку сопровождение обычно происходит следующим поколением разработчиков.

общий объем задач и многообразие видов задач, решаемых при помощи данного класса методов, являются не менее важными характеристиками и показывают степень универсальности соответствующего языка представления методов. Чем больше задач можно решить на *я.п.м.*, тем он универсальнее.

надежность методов языка представления методов должна обеспечиваться минимумом ошибок при работе конкретного метода.

Все эти критерии можно применить и касательно самих *языков представления методов*.

Заключение к Главе 3.2.

Данная глава является началом *Семантической теории программ для компьютерных систем нового поколения*. Логичным развитием данной главы будут:

- уточнение и дополнение понятий *Предметной области и онтологии методов* для достижения полноты теории;
- описание дочерних предметных областей *Предметной области и онтологии методов* для конкретных видов методов, а также уточнение денотационной и операционной семантики спецификации этих методов;
- описание возможных путей реализации метаметодов интерпретации методов различных я.п.м.;
- формализация математических моделей для подсчета оценок эффективности методов.

Глава 3.3.

Агентно-ориентированные модели гибридных решателей задач ostis-систем

⇒ *автор**:
- *Шункевич Д.В.*

⇒ *аннотация**:
[В главе сформулированы актуальные проблемы текущего состояния технологий разработки гибридных решателей задач, предложен подход к их решению на основе Технологии OSTIS. Сформулированы принципы построения решателя задач как иерархической системы навыков, основанной на многоагентном подходе, приведены онтологии агентов и выполняемых ими действий. Сформулированы принципы синхронизации деятельности агентов, а также разработана онтология базового языка программирования для реализации программ агентов и модель интерпретатора такого языка.]

⇒ *подраздел**:
- *§ 3.3.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению*
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*
- *§ 3.3.5. Базовый язык программирования ostis-систем*
- *§ 3.3.6. Решатели задач ostis-систем*
- *§ 3.3.7. Принципы решения задач распределенными коллективами ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

⇒ *ключевой знак**:
- *Язык SCP*
- *Абстрактная scp-машина*

⇒ *ключевое понятие**:
- *действие в sc-памяти*
- *действие в sc-памяти, инициируемое вопросом*
- *действие редактирования базы знаний*
- *задача, решаемая в sc-памяти*
- *класс логически атомарных действий*
- *sc-агент*
- *абстрактный sc-агент*
- *атомарный абстрактный sc-агент*
- *неатомарный абстрактный sc-агент*
- *абстрактный sc-агент, реализуемый на Языке SCP*
- *абстрактный sc-агент, не реализуемый на Языке SCP*
- *тип блокировки*
- *транзакция в sc-памяти*
- *scp-оператор*
- *решатель задач ostis-системы*
- *машина обработки знаний*

⇒ *ключевое отношение**:
- *блокировка**
- *планируемая блокировка**
- *приоритет блокировки**
- *удаляемые sc-элементы**
- *параметр scp-программы'*
- *scp-операнд'*

⇒ *библиографическая ссылка**:
- *Колесников А.В.ГибриИСТиТ-2001кн*
- *Пратт Т..ЯзыкиПРиР-2002кн*
- *Гладков Л.А..ГенетАУП-2006кн*

- *Емельянов В.В..Теори иПЭМ-2003кн*
- *Беркинблит М.Б.НейроСЭУП-1993кн*
- *Головко В.А.НейроСООиП-2001кн*
- *Горбань А.Н..НейроСнПК-1996кн*
- *Вагин В.Н..Досто иПВвИС-2008кн*
- *Кулик Б.А.ЛогикЕР-2001кн*
- *Пойа Д.Матем иПР-1975кн*
- *Батыршин И.З.ОсновОНЛииО-2001кн*
- *Деменков Н.П.НечетУвТСУП-2005кн*
- *Поспелов Д.А.МоделРОАМА-1989кн*
- *Reiter R.aLogic fDR-1980ст*
- *Еремеев А.П.ПострРФнБТ-1997ст*
- *Кахро М.В..ИнстрСПЕСЭВМ-1988кн*
- *Ефимов Е.И.РешатИЗ-1982кн*
- *Раговский А.П.ИнтелМСДВ-2011ст*
- *Подколзин А.С.КомпьМЛПА-2008кн*
- *Курбатов С.С..ПрогрОдАРЗ-2016ст*
- *Владимиров А.Н..ПрогрКУПР-2010ст*
- *Попов Э.В..ИскусИССОиЭС-1990кн*
- *Jackson P.Intro tES-1998bk*
- *World-2023el*
- *RDFCAS-2023el*
- *OWL2WOLDO-2023el*
- *SPARQLO-2023el*
- *Neo4jGDPGDMS-2023el*
- *OWLI-2023el*
- *Грибова В.В..БазовТРИС-2015ст*
- *Грибова В.В..ПроекIACP-2011ст*
- *Филипов А.А..ЕдинаОПИАД-2016ст*
- *Борисов А.Н.ПострИСОнЗсП-2014ст*
- *Dutta S.KnowlPaAAI-1993bk*
- *Pau L.F.KnowlBP-1990art*
- *Wooldridge M.aIntro tMS-2009bk*
- *Weyns D..Envir aaFCAiM-2007art*
- *FIPAACL-2023el*
- *Finin T..KQML aaACL-1994art*
- *KnowlIFS-2023el*
- *Harting R.L..UsingMAtRwMO-2008art*
- *Sims M..AutomODfMS-2008art*
- *Excelente-Toledo C.B..tDynamSoCM-2004art*
- *Nagendra Prasad M.V..LearnSSCiCMAS-1999art*
- *Vasconcelos W.W..NormaCRiMA-2009art*
- *Rumbell T..Emoti iAACA-2012art*
- *Городецкий В.И..БазовОКПА-2015ст*
- *Bordini R.H..ProgrMASiAS-2007bk*
- *Castillo O..RecenAoHIS-2014bk*
- *Eve aWBAP-2023el*
- *GAMAP-2023el*
- *GOAL-2023el*
- *Evertsz R..ImpleIMAS-2004art*
- *JAVAADFE-el*
- *Boissier O..MultiAOPwJC-2013art*
- *Omicini A..Coord fIAD-1999art*
- *Jagannathan V..BlackAaA-1989bk*
- *Поспелов Д.А.СитуаУТиП-1986кн*
- *Dijkstra E.W.CoopeSP-2002bk*
- *Hoare C.A.CommuSP-1983art*
- *Chatterjee B..nBlockDUGwWCAB-2022art*
- *Нариньяни А.С.нФактоКВ-2004ст*
- *Cao O.IndepBUaUtBI-2010art*
- *Cao O..BehavIaNP-2014art*
- *Pavel M..BehavIaCMiSoP-2015art*

- *Альтшуллер Г.С..НайтиИВвТРИ-2010кн*
- *Щедровицкий Г.П.СхемаМССС-1995кн*
- *Сапатый П.С.ЯзыкВкОНС-1986ст*
- *Moldovan D.I..SNAP aVLSIAfAIP-1985art*
- *Летичевский А.А..ИнсерП-2003ст*
- *Летичевский А.А..ИнсерМ-2012ст*

Введение в Главу 3.3.

Одним из ключевых компонентов *интеллектуальной системы*, обеспечивающим возможность решать широкий круг *задач*, является *решатель задач*. Их особенностью по сравнению с другими современными *программными системами* является необходимость решать *задачи* в условиях, когда необходимые сведения не локализованы явно в *базе знаний интеллектуальной системы* и должны быть найдены в процессе решения *задачи* на основании каких-либо критериев.

Говоря другими словами, если в традиционных системах при решении задачи всегда подразумевается, что есть некоторые локализованные исходные данные ("дано") и некоторое описание желаемого результата ("что требуется"), то в *интеллектуальной системе* в качестве исходных данных при решении большого числа *задач* выступает вся имеющаяся на текущий момент в системе информация, то есть вся *база знаний*. Кроме того, при невозможности решения задачи в текущем состоянии базы знаний интеллектуальная система должна иметь возможность понять, чего именно не хватает для продолжения процесса решения и попытаться добыть недостающие сведения во внешней среде (например, запросить у пользователя).

К настоящему времени в рамках различных направлений *Искусственного интеллекта* разработано большое количество различных *моделей решения задач*, каждая из которых позволяет решать задачи определенного класса. Расширение областей применения *интеллектуальных систем* требует от них возможности решать так называемые *комплексные задачи*, решение каждой из которых требует комбинирования нескольких моделей решения задач, при этом априори неизвестно, в каком порядке и сколько раз будет применяться так или иная модель. *решатели задач*, в рамках которых комбинируются несколько *моделей решения задач*, получили название *гибридных решателей задач*, а интеллектуальные системы, в рамках которых комбинируются различные *виды знаний* и различные *модели решения задач – гибридных интеллектуальных систем* (см. *Колесников А.В.ГибриИСТиТ-2001кн*).

Повышение эффективности разработки и эксплуатации *гибридных интеллектуальных систем* требует унификации моделей представления различных *видов знаний* и *моделей обработки знаний*, которая бы позволила легко интегрировать на ее основе компоненты, соответствующие различным моделям решения задач.

§ 3.3.1. Современное состояние, проблемы в области разработки гибридных решателей задач и предлагаемый подход к их решению

⇒ *подраздел**:
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач

Существующее многообразие подходов к решению *задач* в *компьютерных системах* можно разделить на два класса:

- **решение задач с использованием хранимых программ.** В данном случае предполагается, что в системе заранее присутствует программа решения задачи заданного класса и решение сводится к поиску такой программы и интерпретации ее на заданных входных данных. К системам, ориентированным на такой подход к решению задач, относятся в том числе системы, использующие:
  - программы, написанные на языках программирования, относящихся как к императивной, так и к декларативной парадигме, в том числе логических и функциональных (см. *Пратт Т..ЯзыкиПРиР-2002кн*);

- реализации генетических алгоритмов (см. *Гладков Л.А..ГенетАУП-2006кн*, *Емельянов В.В..Теори иПЭМ-2003кн*);
- нейросетевые модели обработки знаний (см. *Беркинблит М.Б.НейроСЭУП-1993кн*, *Головко В.А.НейроСООиП-2001кн*, *Горбань А.Н..НейроСнПК-1996кн*).

Следует отметить, что даже в случае использования хранимой *программы* решение *задачи* далеко не всегда тривиально, поскольку, во-первых, требуется найти такую хранимую *программу* на основе некоторой спецификации, во-вторых, обеспечить ее интерпретацию.

- **решение задач в условиях, когда программа решения не известна.** В этом случае предполагается, что в системе необязательно присутствует готовая *программа* решения для *класса задач*, которому принадлежит некоторая сформулированная задача, подлежащая решению. В связи с этим необходимо применять дополнительные методы поиска путей решения задачи, не рассчитанные на какой-либо узкий *класс задач* (например, разбиение задачи на подзадачи, методы поиска решений в глубину и ширину, метод случайного поиска решения и метод проб и ошибок, метод деления пополам и другие), а также различные модели *логического вывода*: классические дедуктивные (см. *Вагин В.Н..Досто иПВвИС-2008кн*), индуктивные (см. *Кулик Б.А.ЛогикЕР-2001кн* , *Пойа Д.Матем иПР-1975кн*), абдуктивные (см. *Вагин В.Н..Досто иПВвИС-2008кн*); модели, основанные на *нечетких логиках* (см. *Батыршин И.З.ОсновОНЛиниО-2001кн*, *Деменков Н.П.НечетУвТСУП-2005кн*, *Поспелов Д.А.МоделРОАМА-1989кн*), *логике умолчаний* (см. *Reiter R.aLogic fDR-1980ст*), *темпоральной логике* (см. *Еремеев А.П.ПострРФнБТ-1997ст*), и многие другие.

Подробный обзор *решателей задач*, разработанных в период до 1982 года, таких как *GPS*, *STRIPS*, *QA3*, *ПРИЗ* (см. *Кахро М.В..ИнстрСПЕСЭВМ-1988кн*), *ППР* приведен в книге *Ефимов Е.И.РешатИЗ-1982кн* . Среди современных работ, исследующих вопросы применения *моделей решения задач*, не ориентированных на конкретную предметную область, можно выделить *Раговский А.П.ИнтелМСДВ-2011ст* . Среди наиболее заметных представителей класса *интеллектуальных решателей задач*, разработанных в более поздний период, можно отметить *Компьютерный решатель математических задач* (см. *Подколзин А.С.КомпьМЛПА-2008кн*), *Решатель задач по планиметрии НИЦ ЭВТ* (см. *Курбатов С.С..ПрогрОдАРЗ-2016ст*), *Программный комплекс "УДАВ"* (см. *Владимиров А.Н..ПрогрКУПР-2010ст*).

Отдельного внимания заслуживают популярные в настоящее время *системы компьютерной алгебры*, такие как *Wolfram Mathematica*, *Maple*, *MathCAD* и другие. Указанные программные комплексы обладают мощной функциональностью как для проведения различного рода вычислений и экспериментов, так и для построения на их основе систем различного назначения, например обучающих. Более подробно возможности применения систем данного семейства для решения *задач* в рамках *Экосистемы OSTIS* рассмотрены в *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*.

Однако при всем многообразии решаемых рассмотренными системами *задач* множество *классов задач* ограничивается имеющимся в системе набором жестко заданных приемов и алгоритмов решения *задач*, явно используемых при решении той или иной *задачи*. В то же время построение сложных систем, например, систем комплексной автоматизации, невозможно без обеспечения согласованного использования различных *видов знаний* и *моделей решения задач* в рамках одной системы при решении одной и той же *комплексной задачи*. Кроме того, становится актуальной *задача* поддержки такой системы в состоянии, соответствующем текущему уровню развития технологий, дополнения ее более совершенными *моделями* и *методами решения задач*. При этом очевидно, что подобная реконфигурация системы должна осуществляться непосредственно в процессе эксплуатации системы, а не требовать каждый раз, например, полной остановки всего производства или отдельных его частей.

Сказанное выше позволяет сформулировать требования *гибридному решателю задач*:

- в каждый момент времени *решатель задач* должен обеспечивать решение задач из оговоренного класса за оговоренное время, при этом результат решения задачи должен удовлетворять некоторым известным требованиям. Другими словами, как и в случае современных *компьютерных систем*, корректность результатов решения задач на этапе разработки системы должна верифицироваться специальными методами, в том числе для этого могут быть использованы такие современные подходы, как *unit-тестирование*, *тестирование методом «черного ящика»* и другие. Более детально рассмотрим положения, уточняющие сформулированное требование:
  - для явно сформулированных *задач* система всегда должна давать какой-либо ответ за оговоренное время, при этом ответ может быть отрицательным (система не смогла решить поставленную задачу), возможно, с объяснением причин, по которым решение в текущий момент оказалось невозможным. Одним из факторов безуспешности решения является выход за рамки установленного промежутка времени;
  - если явно сформулированная *задача* решена, то все *информационные процессы*, направленные на ее решение, должны быть уничтожены. Особенно актуальным данное требование становится в ситуации, когда для решения одной и той же задачи параллельно используются сразу несколько подходов и заранее неизвестно, какой из них приведет к результату раньше других;
  - после решения задачи вся временная информация, сгенерированная в процессе решения этой *задачи* и имеющая ценность только в контексте решения указанной *задачи*, должна быть удалена из памяти;

- ***гибридный решатель*** должен обеспечивать возможность **согласованного использования различных моделей решения задач** при решении одной и той же *комплексной задачи* в случае необходимости;
- *решатель задач* должен быть легко **модифицируемым**, то есть трудоемкость внесения изменений в уже разработанный *решатель задач* должна быть минимальна. Путями повышения модифицируемости *решателя задач* являются обеспечение локальности вносимых изменений, в том числе – за счет стратификации *решателя задач* на независимые уровни и обеспечение максимальной независимости компонентов *решателя задач* друг от друга, а также наличие готовых компонентов, которые могут быть встроены в *решатель задач* при необходимости. При этом внесение изменений должно осуществляться непосредственно в процессе эксплуатации системы;
- для того чтобы *интеллектуальная система* имела возможность анализировать и оптимизировать имеющийся *решатель задач*, интегрировать в его состав новые компоненты (в том числе самостоятельно), оценивать важность тех или иных компонентов и применимость их для решения той или иной задачи, спецификация *решателя задач* должна быть описана языком, понятным системе, например, при помощи тех же средств, что и обрабатываемые *знания*. Другими словами, *интеллектуальная система* и, соответственно, *решатель задач* должны обладать *рефлексивностью*.

Несмотря на то что в настоящее время существует большое число *моделей решения задач*, многие из которых реализованы и успешно используются на практике в различных системах, остается актуальной проблема низкой согласованности принципов, лежащих в основе реализации таких моделей, и отсутствия единой унифицированной основы для реализации и интеграции различных *моделей решения задач*, что приводит к тому, что:

- затруднена возможность одновременного использования различных *моделей решения задач* в рамках одной системы при решении одной и той же комплексной задачи; практически невозможно комбинировать различные модели с целью решения *задачи*, для которой априори отсутствует *алгоритм* ее решения;
- практически невозможно использовать технические решения, реализованные в одной системе, в других системах, то есть возможности использования компонентного подхода при построении *решателей задач* сильно ограничены. Как следствие, велико количество дублирований аналогичных решений в разных системах;
- фактически отсутствуют комплексные методики и средства построения *решателей задач*, которые бы обеспечивали возможность проектирования, реализации и отладки *решателей задач* различного вида.

Следствиями указанных проблем являются:

- высокая трудоемкость разработки каждого *решателя задач*, увеличение сроков их разработки, а значит, и увеличение затрат на разработку и поддержку соответствующих *интеллектуальных систем*;
- высокая трудоемкость внесения изменений в уже разработанные *решатели задач*, то есть отсутствует или сильно затруднена возможность дополнения уже разработанного *решателя задач* новыми компонентами и внесения изменений в уже существующие компоненты в процессе эксплуатации системы. Таким образом, высока трудоемкость поддержки разработанных *решателей задач*;
- высокий уровень профессиональных требований к разработчикам *решателей задач*, что обусловлено, в частности:
  - высокой сложностью существующих формализмов в области решения *задач*, рассчитанных на их интерпретацию *компьютерной системой*, а не человеком;
  - отсутствием возможности рассматривать разрабатываемые *решатели задач* на разных уровнях детализации, выделения на каждом уровне достаточно независимых компонентов, что затрудняет процесс проектирования, тестирования и отладки таких *решателей задач*, а также снижает эффективность попыток объединения разработчиков *решателей задач* в коллективы по причине увеличения накладных расходов на согласование их деятельности;
  - низким уровнем информационной поддержки разработчиков и автоматизации их *деятельности*.

Для решения перечисленных проблем необходимо разработать комплекс моделей, методики и средств разработки *гибридных решателей задач*, удовлетворяющих перечисленным ранее требованиям.

Исторически сложились два основных подхода к построению *решателей задач интеллектуальных компьютерных систем*.

Первый подход предполагает наличие в системе фиксированного *решателя задач* (например, машины логического вывода), к которому впоследствии добавляется *база знаний*, наполнение которой определяется *предметной областью*, в которой должна работать система. Такие системы получили название "пустых" *экспертных систем* (см. *Попов Э.В..ИскусИССОиЭС-1990кн*) или "оболочек" (expert system shells, см. *Jackson P.Intro tES-1998bk*). Данный подход, как правило, использовался для разработки относительно несложных систем и в настоящее время не имеет широкого применения.

Второй подход, широко используемый в настоящее время, предполагает наличие программных средств доступа к информации, хранящейся в некоторой базе, совместимых с различными популярными *языками программирования*. Данный подход широко используется, например, в системах, построенных на основе стандартов *W3C* (см. *World-2023el*), таких как *RDF* (см. *RDFCAS-2023el*), *OWL* (см. *OWL2WOLDO-2023el*), *SPARQL* (см. *SPARQLO-2023el*), а также *графовых с.у.б.д.*, таких как *Neo4j* (см. *Neo4jGDPGDMS-2023el*). Структура *решателя задач*, построенного

на базе таких средств, определяется разработчиком в каждом конкретном случае и не фиксируется какими-либо стандартами. Такой подход обладает большей гибкостью, но отсутствие унификации в структуре и процессе разработки *решателей задач* приводит к отсутствию совместимости компонентов *решателей задач*, созданных разными разработчиками, большому количеству дублирований одних и тех же решений, повышению накладных расходов в процессе разработки и поддержки *решателя задач*. Также существует большое количество реализаций так называемых *ризонеров* (semantic reasoners), осуществляющих *логический вывод* на *онтологиях*, представленных в формате *OWL 2*, а также средств разработки и редактирования таких *онтологий*. Полный список таких средств, признанных консорциумом *W3C*, можно найти на сайте *OWLI-2023el*. Как видно из приведенной на нем таблицы, подавляющее большинство средств способно осуществлять только прямой *логический вывод* на основе *отношений*, описанных в *онтологии*.

Среди комплексных подходов к построению *решателей задач*, разрабатываемых русскоязычными авторами, можно выделить проект *IACPaaS* (см. *Грибова В.В..БазовТРИС-2015ст*, *Грибова В.В..ПроекIACP-2011ст*), активно развивающийся в настоящее время. Целью данного проекта является разработка облачной платформы для построения на ее основе *интеллектуальных сервисов* различного назначения. В данном проекте активно используются *библиотеки многократно используемых компонентов интеллектуальных систем*. Конкретно для построения *решателей задач*, а также *пользовательских интерфейсов* таких систем используется *многоагентный подход*. Несмотря на близость некоторых технологических решений, реализуемых в проекте *IACPaaS* и в рамках *Технологии OSTIS*, основной целью указанного проекта является предоставление пользователю большого числа разнородных сервисов, выбор которых осуществляется самим пользователем, в то время как одним из ключевых принципов *Технологии OSTIS* является разработка общей формальной основы для интеграции различных *моделей решения задач* с целью их комбинирования при решении одной и той же *комплексной задачи*.

Задачи интеграции различных подходов, в том числе связанных с решением задач, исследуются также в работе *Филипов А.А..ЕдинаОПИАД-2016ст* и других работах тех же авторов.

Компонентному проектированию *интеллектуальных систем, основанных на знаниях*, посвящена работа *Борисов А.Н.ПострИСОнЗсП-2014ст* , в которой обосновывается необходимость накопления и повторного использования различных компонентов *интеллектуальных систем*, предлагаются возможные решения данной проблемы с использованием *онтологий*.

Состояние работ англоязычных авторов, посвященных вопросам решения задач в *системах, основанных на знаниях*, и актуальных на момент начала 1990-х годах, отражено в обзорных публикациях *Dutta S.KnowlPaAAI-1993bk* , *Рао L.F.KnowlBP-1990art*. Более поздние англоязычные работы в данной области в основном ориентированы на решение конкретных частных *задач* в системах, построенных на основе стандартов *W3C*, о которых более подробно было сказано выше.

Таким образом, можно сказать, что существует ряд конкретных разработок в направлении построения *комплексных технологий разработки интеллектуальных систем* различных классов, в том числе с использованием *библиотек многократно используемых компонентов*, однако проблема разработки комплексной технологии построения *гибридных решателей задач* в рамках рассмотренных подходов не решена. Во многом это обусловлено отсутствием унифицированной формальной основы для представления любых *видов знаний*, в том числе различного рода программ, отсутствием строгих принципов, регламентирующих процесс построения *решателей задач*, а также средств поддержки разработчиков таких *решателей задач* и их компонентов.

Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах

Рассмотрим принципы, лежащие в основе предлагаемого подхода к разработке *гибридных решателей задач*:

- в качестве основы для построения модели гибридного *решателя задач* предлагается использовать *многоагентный подход*. Данный подход позволяет обеспечить основу для построения параллельных асинхронных систем, имеющих распределенную архитектуру, повысить модифицируемость и производительность разработанных *решателей задач*;
- процесс решения любой *задачи* предлагается декомпозировать на *логически атомарные действия*, что также позволит обеспечить совместимость и модифицируемость *решателей задач*;
- *решатель задач* (как объединенный, так и *решатель задач* частного вида) предлагается рассматривать как иерархическую систему, состоящую из нескольких взаимосвязанных уровней. Такой подход позволяет обеспечить возможность проектирования, отладки и верификации компонентов на разных уровнях независимо от других уровней, что существенно упрощает задачу построения *решателя задач* за счет снижения накладных расходов;
- предлагается записывать всю информацию о решателе и решаемых им задачах при помощи *SC-кода* в той же *базе знаний*, что и собственно предметные *знания* системы. В общем случае такая информация включает:

- *спецификацию агентов*, входящих в состав *решателя задач*;
- *спецификацию методов*, интерпретируемых *агентами решателя задач*;
- спецификацию всех *информационных процессов*, выполняемых агентами в *семантической памяти*, в том числе – конструкции, обеспечивающие синхронизацию выполнения параллельных процессов;
- спецификацию всех *задач*, на решение которых направлены указанные *информационные процессы*.

Описание всей указанной информации в единой семантической памяти позволит, во-первых, обеспечить независимость разрабатываемых *решателей задач* от *ostis-платформы* (см. *Главу 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*), во-вторых, обеспечить возможность системы анализировать происходящие в ней процессы, оптимизировать и синхронизировать их выполнение, то есть обеспечить *рефлексивность* проектируемых *интеллектуальных систем*.

Ориентация на *многоагентный подход* как основу для построения *гибридных решателей задач* обусловлена следующими основными преимуществами такого подхода (см. *Wooldridge M.aIntro tMS-2009bk*):

- автономность (независимость) *агентов* в рамках такой системы, что позволяет локализовать изменения, вносимые в *решатель задач* при его эволюции, и снизить соответствующие трудозатраты, а также обеспечить устойчивость такой системы к отказам некоторых агентов;
- децентрализация обработки, то есть отсутствие единого контролирующего центра, что также позволяет локализовать вносимые в *решатель задач* изменения;
- возможность параллельной работы разных *информационных процессов*, соответствующих как одному *агенту*, так и разным агентам, как следствие, – возможность распределенного решения задач. Однако возможность параллельного выполнения *информационных процессов* подразумевает наличие средств синхронизации такого выполнения, разработка которых является отдельной задачей и подробно рассматривается ниже;
- активность *агентов* и *многоагентной системы* в целом, дающая возможность при общении с такой системой не указывать явно способ решения поставленной *задачи*, а формулировать задачу в **декларативном ключе**.

Построение модели *многоагентной системы* требует уточнения модели каждого компонента, входящего в ее состав, а именно:

- **модель собственно агента**, входящего в состав такой системы, включая классификацию таких *агентов* и набор понятий, характеризующих каждый агент в рамках системы. В настоящее время наиболее популярной является модель *BDI* (belief-desire-intention), в рамках которой предполагается описывать на соответствующих языках "убеждения", "желания" и "намерения" каждого агента системы;
- **модель среды**, в рамках которой находятся агенты, на события в которой они реагируют и в рамках которой могут осуществлять некоторые преобразования. Обзор разновидностей сред для многоагентных систем приводится в работе *Weyns D..Envir aaFCAiM-2007art*;
- **модель коммуникации агентов**, в рамках которой уточняется язык взаимодействия *агентов* (структура и классификация сообщений) и способ передачи сообщений между *агентами*.

В свою очередь, для разработки модели коммуникации *агентов* необходимо отдельно уточнить каждый из ее компонентов:

- **Принципы обмена сообщениями между агентами**, то есть то, каким образом эти сообщения передаются от *агента* к *агенту*;
- **Классификацию, семантику и прагматику таких сообщений**, то есть *смысл* передаваемой информации и цель такого взаимодействия. В настоящее время стандартами, описывающими структуру передаваемых агентами сообщений, являются *Agent Communication Language* (*ACL*) (см. *FIPAACL-2023el*), разработанный сообществом *FIPA*, язык *KQML* (см. *Finin T..KQML aaACL-1994art*). Указанные стандарты уточняют базовые компоненты каждого сообщения (кодировка, язык сообщения, используемую онтологию понятий, получателя, отправителя и так далее), не ограничивая при этом *смысл* сообщения в целом. Также для коммуникации между агентами используется язык *KIF*, предназначенный для обмена *знаниями* между любыми программными компонентами (см. *KnowlIFS-2023el*);
- **Принципы координации деятельности агентов** В литературе рассматривается большое число вариантов координации деятельности *агентов*. В работе *Harting R.L..UsingMAtRwMO-2008art* предлагается выделить агенты более высокого уровня (*метаагенты*), *задачей* которых является сбор информации от *агентов* нижнего уровня и их координация, схожие идеи высказываются в работе *Sims M..AutomODfMS-2008art*. В работах *Excelente-Toledo C.B..tDynamSoCM-2004art*, *Nagendra Prasad M.V..LearnSSCiCMAS-1999art* предлагаются варианты автоматического выбора оптимального механизма координации *агентов* для достижения общей цели. Предлагаются также социально-психологические модели координации деятельности *агентов*, например, на основе некоторых общих "законов" (см. *Vasconcelos W.W..NormaCRiMA-2009art*) или эмоций (см. *Rumbell T..Emoti iAACA-2012art*). В работе *Городецкий В.И..БазовОКПА-2015ст* предложен вариант онтологии коллективного поведения автономных *агентов*.

К основным недостаткам большинства популярных современных средств построения *многагентных систем* (см. *Bordini R.H..ProgrMASiAS-2007bk*, *Castillo O..RecenAoHIS-2014bk*, *Eve aWBAP-2023el*, *GAMAP-2023el*, *GOAL-*

2023el, *Evertsz R..ImpleIMAS-2004art*, *JAVAADFE-el*, *Boissier O..MultiAOPwJC-2013art*) можно отнести следующие:

- жесткая ориентация большинства средств на модель *BDI* приводит к существенным накладным расходам, связанным с необходимостью выражения конкретной практической *задачи* в системе понятий *BDI*. В то же время ориентация на модель *BDI* неявно провоцирует искусственное разделение языков, описывающих собственно компоненты *BDI* и знания *агента* о внешней среде, что приводит к отсутствию *унификации представления* и, соответственно, дополнительным накладным расходам;
- большинство современных средств построения *многоагентных систем* ориентированы на представление *знаний агента* при помощи узкоспециализированных языков, зачастую не предназначенных для представления *знаний* в широком смысле. Речь при этом идет как о знаниях агента о себе самом, так и *знаниях* о внешней среде. В некоторых подходах вначале строится онтология, для создания которой, однако, часто используются средства с низкой выразительной способностью, не предназначенные для построения *онтологий* (см. *Evertsz R..ImpleIMAS-2004art*, *JAVAADFE-el*). В конечном итоге такой подход приводит к сильной ограниченности возможностей построенных *многоагентных систем* и их несовместимости;
- абсолютное большинство современных средств предполагает, что взаимодействие *агентов* осуществляется путем обмена сообщениями непосредственно от *агента* к *агенту* или посредством специальных коммуникационных центров (см. *Omicini A..Coord fIAD-1999art*), например, в случае взаимодействия *агентов* в глобальной сети. Такой подход обладает существенным недостатком, связанным с тем, что в этом случае каждый *агент* системы должен иметь достаточно полную информацию о других агентах в системе, что приводит к дополнительным затратам ресурсов, кроме того, добавление или удаление одного или нескольких *агентов* приводит к необходимости оповещения об этом других *агентов*. Данная проблема решается путем организации общения агентов по принципу "доски объявлений" (см. *Jagannathan V..BlackAaA-1989bk*), предполагающему, что сообщения помещаются в некоторую общую для всех агентов область, при этом каждый *агент* в общем случае может не знать, какому из агентов адресовано сообщение и от какого из *агентов* получено то или иное сообщение. Кроме того, в построенной таким образом системе легче обеспечивается параллельное решение несвязанных друг с другом *задач*, поскольку сообщения, относящиеся к одной *задаче*, будут игнорироваться агентами, решающими другую задачу. Однако данный подход не исключает проблему, связанную с необходимостью разработки специализированного языка взаимодействия *агентов*, который в общем случае не связан с языком, на котором описываются *знания агента* о решаемых *задачах* и окружающей среде;
- многие средства построения *многоагентных систем* построены таким образом, что логический уровень взаимодействия *агентов* жестко привязан к физическому уровню реализации *многоагентной системы*. Например, при передаче сообщений от агента к агенту разработчику *многоагентной системы* необходимо помимо семантически значимой информации указывать ip-адрес компьютера, на котором расположен *агент-получатель*, кодировку, с помощью которой закодирован текст сообщения, и другую техническую информацию, обусловленную исключительно особенностями текущей реализации средств;
- в большинстве подходов среда, с которой взаимодействуют *агенты*, уточняется отдельно разработчиком для каждой *многоагентной системы*, что с одной стороны, расширяет возможности применения соответствующих средств, но, с другой стороны, приводит к существенным накладным расходам и несовместимости таких многоагентных систем. Кроме того, в ряде случаев разработчик также обязан учитывать особенности технической реализации средств разработки в плане их стыковки с предполагаемой средой, в роли которой может выступать, например, локальная или глобальная сеть.

Перечисленные недостатки предполагается устранять за счет использования следующих принципов:

- коммуникацию агентов предлагается осуществлять по принципу *"доски объявлений"*, однако в отличие от классического подхода в роли сообщений выступают спецификации в общей семантической памяти выполняемых *агентами действий*, направленных на решение каких-либо задач, а в роли среды коммуникации выступает сама эта *семантическая память*. Такой подход позволяет:
  - исключить необходимость разработки специализированного языка для обмена сообщениями;
  - обеспечить "обезличенность" общения, то есть каждый из *агентов* в общем случае не знает, какие еще агенты есть в системе, кем сформулирован и кому адресован тот или иной запрос. Таким образом, добавление или удаление агентов в систему не приводит к изменениям в других *агентах*, что обеспечивает модифицируемость всей системы;
  - агентам, в том числе конечному пользователю, формулировать задачи в *декларативном ключе*, то есть не указывать для каждой задачи способ ее решения. Таким образом, агенту заранее не нужно знать, каким образом система решит ту или иную задачу, достаточно лишь специфицировать конечный результат;
  - сделать средства коммуникации *агентов* и синхронизации их деятельности более понятными разработчику и пользователю системы, не требующими изучения специальных низкоуровневых типов данных и форматов сообщений. Таким образом повышается доступность предлагаемых решений широкому кругу разработчиков.

  Следует отметить, что такой подход позволяет при необходимости организовать обмен сообщениями между *агентами* напрямую и, таким образом, может являться основой для моделирования многоагентных систем, предполагающих другие способы взаимодействия между *агентами*.

- в роли внешней среды для агентов выступает та же *семантическая память*, в которой формулируются задачи и посредством которой осуществляется взаимодействие *агентов*. Такой подход обеспечивает унификацию среды для различных систем *агентов*, что, в свою очередь, обеспечивает их совместимость;
- спецификация каждого агента описывается средствами *SC-кода* в *базе знаний*, что позволяет:
  - минимизировать число специализированных средств, необходимых для спецификации агентов, как языковых, так и инструментальных;
  - с одной стороны – минимизировать необходимую в общем случае спецификацию агента, которая включает условие его инициирования и *программу*, описывающую алгоритм работы *агента*, с другой стороны – обеспечить возможность произвольного расширения спецификации для каждого конкретного случая, в том числе возможность реализации модели *BDI* и других;
- синхронизацию деятельности *агентов* предполагается осуществлять на уровне выполняемых ими процессов, направленных на решений тех или иных задач в *семантической памяти*. Таким образом, каждый агент трактуется как некий абстрактный процессор, способный решать задачи определенного класса. При таком подходе необходимо решить задачу обеспечения взаимодействия параллельных асинхронных процессов в общей *семантической памяти*, для решения которой можно заимствовать и адаптировать решения, применяемые в традиционной *линейной памяти*. При этом вводится дополнительный класс агентов – *метаагенты*, задачей которых является решение возникающих проблемных ситуаций, таких как *взаимоблокировки*;
- каждый *информационный процесс* в любой момент времени имеет ассоциативный доступ к необходимым фрагментам *базы знаний*, хранящейся в семантической памяти, за исключением фрагментов, заблокированных другими процессами в соответствии с рассмотренным ниже механизмом синхронизации. Таким образом, с одной стороны, исключается необходимость хранения каждым агентом информации о внешней среде, с другой стороны, каждый *агент*, как и в классических *многоагентных системах*, обладает только частью всей информации, необходимой для решения задачи.

  Важно отметить, что в общем случае невозможно априори предсказать, какие именно знания, модели и способы решения задач понадобятся системе для решения конкретной задачи. В связи с этим необходимо обеспечить, с одной стороны, возможность доступа ко всем необходимым фрагментам *базы знаний* (в пределе – ко всей *базе знаний*), с другой стороны – иметь возможность локализовать область поиска пути решения *задачи*, например, рамками одной *предметной области*.

  Каждый из *агентов* обладает набором ключевых элементов (как правило, понятий), которые он использует в качестве отправных точек при ассоциативном поиске в рамках *базы знаний*. Набор таких элементов для каждого *агента* уточняется на этапах проектирования *решателя задач*. Уменьшение числа ключевых элементов *агента* делает его более универсальным, однако снижает эффективность его работы за счет необходимости выполнения дополнительных операций.

Кроме *многоагентного подхода*, в основу принципов решения задачи в рамках *Технологии OSTIS* предлагается положить ряд идей, связанных с концепцией *ситуационного управления*, рассмотренной в работе *Д.А. Поспелова Поспелов Д.А.СитуаУТиП-1986кн*. До настоящего времени попытки реализации указанной концепции, несмотря на ее актуальность и востребованность, сводились к частным решениям для конкретных *классов задач* и, к сожалению, не получили широкого распространения. В значительной степени это обусловлено отсутствием универсальной унифицированной основы, которая бы позволила на ее базе создавать языки ситуационного управления в применении к конкретным предметным областям и, что еще более важно, повторно использовать фрагменты описаний на таких языках.

Данную проблему можно решить используя предлагаемый в рамках *Технологии OSTIS SC-код* и семейство *онтологий верхнего уровня*, разработанных на его основе. В частности, реализации идей ситуационного управления способствуют такие принципы, как:

- *SC-код* как базовый язык для описания любой информации в *базе знаний* и, соответственно, для построения языков ситуационного управления на его основе;
- *Базовая денотационная семантика SC-кода*, которая позволяет обеспечить возможность формального уточнения всех используемых понятий в виде формального набора *онтологий*, что позволяет обеспечить совместимость разрабатываемых систем и возможность повторного использования их компонентов;
- *агентно-ориентированный подход* к обработке информации, предполагающий реакцию *агентов* на возникновение в *базе знаний* определенных *ситуаций* и *событий*.

Учитывая рассмотренные выше принципы реализации *многоагентного подхода* и *ситуационного управления* в рамках *Технологии OSTIS*, сформулируем более детально основные принципы, лежащие в основе подхода к обработке информации, предлагаемого в *Технологии OSTIS*:

- В основе *решателя задач* каждой *ostis-системы* лежит многоагентная система, агенты которой взаимодействуют между собой только(!) через общую для них *sc-память* посредством спецификации в этой памяти выполняемых ими *действий в sc-памяти*. При этом пользователи *ostis-системы* также считаются *агентами* этой системы. Кроме того, ***sc-агенты*** делятся на внутренние, рецепторные и эффекторные. Взаимодействие между *агентами* через общую ***sc-память*** сводится к следующим видам действий:

- К использованию общедоступной для соответствующей группы *sc-агентов* части хранимой *базы знаний*;
- К формированию (генерации) новых фрагментов *базы знаний* и/или к корректировке (редактированию) каких-либо фрагментов доступной части *базы знаний*;
- К интеграции (погружению) новых (обновленных) фрагментов в состав доступной части *базы знаний*;

Подчеркнем, что *sc-агенты* не общаются между собой напрямую путем отправки сообщений, как это делается в большинстве современных подходов к построению *многоагентных систем*. Кроме того, *sc-агенты* имеют доступ к общей для них базе знаний за счет чего гарантируется семантическая совместимость (взаимопонимание) между агентами, включая и пользователей *ostis-систем*.

- Пользователь *ostis-системы* не может сам непосредственно выполнить какое-либо *действие* в sc-памяти, но он может средствами *пользовательского интерфейса* инициировать построение (генерацию, формирование в *sc-памяти*) *sc-текста*, являющегося спецификацией *действия в sc-памяти*, выполняемого либо одним *атомарным sc-агентом* за один акт, либо одним *атомарным sc-агентом* за несколько актов, либо коллективом *sc-агентов* (*неатомарным sc-агентом*). В спецификации каждого такого *действия в sc-памяти*, инициированного пользователем, этот пользователь указывается как заказчик этого действия. Таким образом, пользователь *ostis-системы* дает поручения (задания, команды) *sc-агентам* этой системы на выполнение различных специфицируемых им действий в *sc-памяти*.
- Каждый *sc-агент*, выполняя некоторое *действие в sc-памяти*, должен "помнить", что *sc-память*, над которой он работает, является общим ресурсом не только для него, но и для всех остальных *sc-агентов*, работающих над этой же *sc-памятью*. Поэтому *sc-агент* должен соблюдать определенную этику поведения в коллективе таких *sc-агентов*, которая должна минимизировать помехи, которые он создает другим *sc-агентам*.
- Деятельность каждого агента *ostis-системы* дискретна и представляет собой множество элементарных действий (актов). При этом при выполнении каждого акта агент может устанавливать блокировки нескольких типов на фрагменты базы знаний. Указанные блокировки позволяют запретить другим агентам изменять указанный фрагмент базы знаний или вообще сделать его "невидимым" для других агентов. Блокировки устанавливаются самим агентом при выполнении соответствующего акта и снимаются им же на последнем этапе выполнения этого акта или раньше, если это возможно.
- Если некий *sc-агент* выполняет некоторое *действие в sc-памяти*, то он на время выполнения этого действия может:
  - Запретить другим *sc-агентам* изменять состояние некоторых sc-элементов, хранимых в *sc-памяти* – удалять их, изменять тип;
  - Запретить другим *sc-агентам* добавлять или удалять элементы некоторых множеств, обозначаемых соответствующими *sc-узлами*;
  - Запретить другим *sc-агентам* доступ на просмотр некоторых *sc-элементов*, то есть эти *sc-элементы* становятся полностью "невидимыми" (полностью заблокированными) для других *sc-агентов* но только на время выполнения соответствующего *действия*.

  Указанные блокировки должны быть полностью сняты до завершения выполнения соответствующего *действия*. Подчеркнем, что число *sc-элементов*, блокируемых на время выполнения некоторого действия, в основном входят атомарные и неатомарные связки, и не должны входить *sc-узлы*, обозначающие бесконечные классы каких-либо сущностей, и, тем более, *sc-узлы*, обозначающие различные *понятия* (ключевые классы различных предметных областей). Этичное (неэгоистичное) поведение *sc-агента*, касающееся блокировки *sc-элементов* (то есть ограничения к ним доступа другим *sc-агентам*) предполагает соблюдение следующих правил:
  - Не следует блокировать больше *sc-элементов*, чем это необходимо для решения задачи;
  - Как только для какого-либо *sc-элемента* необходимость его блокировки отпадает до завершения выполнения соответствующего действия, этот *sc-элемент* желательно сразу деблокировать;

  Для того, чтобы *sc-агент* имел возможность работы с каким-либо произвольным *sc-элементом*, он должен либо убедиться в том, что этот *sc-элемент* не входит во фрагмент базы знаний, входящий в *полную блокировку*, либо убедиться в том, что эта блокировка не установлена самим этим агентом. Особой группой полностью заблокированных *sc-элементов* (на время выполнения действия *sc-агентом*) являются вспомогательные *sc-элементы* ("леса"), создаваемые только на время выполнения этого действия. Эти sc-элементы в конце выполнения действия должны не деблокироваться, а удаляться.
- Если *действие в sc-памяти*, выполняемое *sc-агентом*, завершилось (то есть стало прошлой сущностью), то *sc-агент* оформляет результат этого *действия*, указывая (1) удаленные *sc-элементы* и (2) сгенерированные sc-элементы. Это необходимо, если по каким-либо причинам придется сделать откат этого *действия*, т.е возвратиться к состоянию базы знаний до выполнения указанного *действия*.

Перечислим некоторые достоинства предлагаемого подхода к организации обработки знаний в *ostis-системах*:

- поскольку обработка осуществляется *агентами*, которые обмениваются сообщениями только через общую память, добавление нового агента или исключение (деактивация) одного или нескольких существующих *агентов*, как правило, не приводит к изменениям в других *агентах*, поскольку агенты не обмениваются сообщениями напрямую;

- инициирование *агентов* осуществляется децентрализованно и чаще всего независимо друг от друга, таким образом, даже существенное расширение числа агентов в рамках одной системы не приводит к ухудшению ее производительности;
- спецификации *агентов* и, как будет показано ниже, их программы могут быть записаны на том же языке, что и обрабатываемые знания, что существенно сокращает перечень специализированных средств, предназначенных для проектирования таких *агентов* и их коллективов, а также их анализа, верификации и оптимизации, и упрощает разработку системы за счет использования более универсальных компонентов.

§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой

⇒ *ключевое понятие**:
- *действие в sc-памяти*
- *действие в sc-памяти, инициируемое вопросом*
- *действие редактирования базы знаний*
- *задача, решаемая в sc-памяти*
- *класс логически атомарных действий*

Прежде, чем детально рассматривать предлагаемую архитектуру *гибридных решателей задач*, необходимо уточнить общее понятие *действия*, выполняемого в *sc-памяти* и соответствующих *задач*, *классов действий* и *классов задач*.

действие в sc-памяти
:= [внутреннее действие ostis-системы]
:= [действие, выполняемое в sc-памяти]
:= [действие, выполняемое в абстрактной унифицированной семантической памяти]
:= [действие, выполняемое машиной обработки знаний ostis-системы]
:= [действие, выполняемое агентом или коллективом агентов ostis-системы]
:= [информационный процесс над базой знаний, хранимой в sc-памяти]
:= [процесс решения информационной задачи в sc-памяти]
⊂ *процесс в sc-памяти*

Каждое ***действие в sc-памяти*** обозначает некоторое преобразование, выполняемое некоторым *sc-агентом* (или коллективом *sc-агентов*) и ориентированное на преобразование *sc-памяти*. Спецификация действия после его выполнения может быть включена в протокол решения некоторой задачи.

Преобразование состояния *базы знаний* включает, в том числе и *информационный поиск*, предполагающий (1) локализацию в *базе знаний ответа на вопрос*, явное выделение структуры ответа и (2) трансляцию ответа на некоторый *внешний язык*.

Во множество ***действий в sc-памяти*** входят знаки *действий* самого различного рода, семантика каждого из которых зависит от конкретного контекста, то есть ориентации действия на какие-либо конкретные объекты и принадлежности *действия* какому-либо конкретному *классу действий*.

Следует четко отличать:
- Каждое конкретное ***действие в sc-памяти***, представляющее собой некоторый переходный процесс, переводящий *sc-память* из одного состояния в другое;
- Каждый тип ***действий в sc-памяти***, представляющий собой некоторый класс однотипных (в том или ином смысле) *действий*;
- *sc-узел*, обозначающий некоторое конкретное ***действие в sc-памяти***;
- *sc-узел*, обозначающий структуру, которая является описанием, спецификацией, заданием, постановкой соответствующего действия.

Рассмотрим боле детально классификацию *действий в sc-памяти*:

действие в sc-памяти
⊃ *действие в sc-памяти, инициируемое вопросом*
⊃ *действие редактирования базы знаний ostis-системы*
⊃ *действие установки режима ostis-системы*
⊃ *действие редактирования файла, хранимого в sc-памяти*
⊃ *действие интерпретации программы, хранимой в sc-памяти*
  ⊃ *действие интерпретации scp-программы*

действие в sc-памяти, инициируемое вопросом

:= [действие, направленное на формирование ответа на поставленный вопрос]
- ⊃ *действие. сформировать заданный файл*
- ⊃ *действие. сформировать заданную структуру*
  - ⊃ *действие. верифицировать заданную структуру*
    - ⊃ *действие. установить истинность или ложность указываемого логического высказывания*
    - ⊃ *действие. установить корректность или некорректность указываемой структуры*
    - ⊃ *действие. сформировать структуру, описывающую некорректности, имеющиеся в указываемой структуре*
  - ⊃ *действие. уточнить тип заданного sc-элемента*
    - ⊃ *действие. установить позитивность/негативность указываемой sc-дуги принадлежности или непринадлежности*
  - ⊃ *действие. сформировать семантическую окрестность*
    - ⊃ *действие. сформировать полную семантическую окрестность указываемой сущности*
    - ⊃ *действие. сформировать базовую семантическую окрестность указываемой сущности*
    - ⊃ *действие. сформировать частную семантическую окрестность указываемой сущности*
  - ⊃ *действие. сформировать структуру, описывающую связи между указываемыми сущностями*
    - ⊃ *действие. сформировать структуру, описывающую сходства указываемых сущностей*
    - ⊃ *действие. сформировать структуру, описывающую различия указываемых сущностей*
  - ⊃ *действие. сформировать структуру, описывающую способ решения указываемой задачи*
  - ⊃ *действие. сформировать план генерации ответа на указанный вопрос*
  - ⊃ *действие. сформировать протокол выполнения указываемого действия*
  - ⊃ *действие. сформировать обоснование корректности указываемого решения*
  - ⊃ *действие. верифицировать обоснование корректности указываемого решения*
  - ⊃ *действие, направленное на установление темпоральных характеристик указываемой сущности*
  - ⊃ *действие, направленное на установление пространственных характеристик указываемой сущности*

действие редактирования базы знаний

- ⊃ *действие. изменить направление указанной sc-дуги*
- ⊃ *действие. исправить ошибки в заданной структуре*
- ⊃ *действие. преобразовать указанную структуру в соответствии с указанным правилом*
- ⊃ *действие. отождествить два указанных sc-элемента*
- ⊃ *действие. включить множество*
  - := [сделать все элементы множества ***Si*** явно принадлежащими множеству ***Sj***, то есть сгенерировать соответствующие sc-дуги принадлежности]
- ⊃ *действие генерации sc-элементов*
  - ⊃ *действие генерации, одним из аргументов которого является некоторая обобщенная структура*
    - ⊃ *действие. сгенерировать структуру, изоморфную указываемому образцу*
  - ⊃ *действие. сгенерировать sc-элемент указанного типа*
    - ⊃ *действие. сгенерировать sc-коннектор указанного типа*
    - ⊃ *действие. сгенерировать sc-узел указанного типа*
  - ⊃ *действие. сгенерировать файл с заданным содержимым*
  - ⊃ *действие. установить указанный файл в качестве основного идентификатора указанного sc-элемента для указанного внешнего языка*
- ⊃ *действие. обновить понятия*
  - := [действие. заменить неосновные понятия на их определения через основные понятия]
  - := [действие. заменить некоторое множество понятий на другое множество понятий]
- ⊃ *действие. интегрировать информационную конструкцию в текущее состояние базы знаний*
  - ⊃ *действие. интегрировать содержимое указанного файла в текущее состояние базы знаний*
    - ⊃ *действие. протранслировать содержимое указанного файла в sc-память*
  - ⊃ *действие. интегрировать указанную структуру в текущее состояние базы знаний*
- ⊃ *действие. дополнить описание прошлого состояния ostis-системы*
  - ⊃ *действие. дополнить структуру, описывающую историю эволюции ostis-системы*
  - ⊃ *действие. дополнить структуру, описывающую историю эксплуатации ostis-системы*
- ⊃ *действие удаления sc-элементов*
  - ⊃ *действие. удалить указанные sc-элементы*
    - ⊃ *действие. удалить sc-элементы, входящие в состав указанной структуры и не являющиеся ключевыми узлами каких-либо sc-агентов*

действие. отождествить два указанных sc-элемента

:= [действие. совместить два указанных sc-элемента]
:= [действие. склеить два указанных sc-элемента]

⇒ *разбиение**:
{• *действие. физически отождествить два указанных sc-элемента*
• *действие. логически отождествить два указанных sc-элемента*
}

Каждое ***действие. отождествить два указанных sc-элемента*** может быть выполнено как *действие. физически отождествить два указанных sc-элемента* или *действие. логически отождествить два указанных sc-элемента*. В случае логического отождествления в протоколе деятельности агентов сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были сгенерированы, а какие удалены. В случае физического отождествления протокол действия не сохраняется.

Каждое ***действие. обновить понятия*** обозначает переход от какой-то группы понятий, использовавшихся ранее, к другой группе понятий, которые будут использоваться вместо первых, и станут *основными понятиями*. В общем случае ***действие. обновить понятия*** состоит из следующих этапов:

- Определить заменяемые понятия на основе заменяющих;
- Внести соответствующие изменения в *программы sc-агентов*, ключевыми узлами которых являются обновляемые понятия;
- Заменить все конструкции в *базе знаний*, содержащие заменяемые понятия, в соответствии с определениями этих понятий через заменяющие их понятия;
- При необходимости, *sc-элементы*, обозначающие замененные таким образом понятия, могут быть полностью выведены из текущего состояния базы знаний.

Первым аргументом (входящим в знак *действия* под атрибутом *1'*) ***действия. обновить понятия*** является знак множества *sc-узлов*, обозначающих заменяемые понятия, вторым (входящим в знак *действия* под атрибутом *2'*) - знак множества *sc-узлов*, обозначающих заменяющие понятия. В общем случае любое или оба этих множества могут быть *синглетонами*.

действие. удалить указанные sc-элементы
⇒ *разбиение**:
{• *действие. физически удалить указанные sc-элементы*
• *действие. логически удалить указанные sc-элементы*
}

Каждое ***действие. удалить указанные sc-элементы*** может быть выполнено как *действие. физически удалить указанные sc-элементы* или *действие. логически удалить указанные sc-элементы*. В случае логического удаления в протоколе деятельности *агентов* сохраняется само действие с его спецификацией, включающей обязательное указание того, какие элементы были удалены, то есть по сути, элементы просто исключаются из текущего состояния *базы знаний*. В случае физического удаления протокол действия не сохраняется.

В случае удаления какого-либо *sc-элемента*, инцидентные ему *связки*, в том числе *sc-коннекторы*, также удаляются.

Для того, чтобы выполнить ***действие. интегрировать указанную структуру в текущее состояние базы знаний***, необходимо склеить *sc-элементы*, входящие в интегрируемую *структуру* с синонимичными им *sc-элементами*, входящими в текущее состояние *базы знаний*, заменить неиспользуемые (например, устаревшие) понятия, входящие в интегрируемую *структуру*, на используемые (то есть заменить неиспользуемые понятия на их определения через используемые), явно включить все элементы интегрируемой *структуры* в число элементов *согласованной части базы знаний* и явно включить все элементы интегрируемой *структуры* в число элементов одного из атомарных разделов *согласованной части базы знаний*.

Более подробно соотношение между понятиями "*действие*", "*задача*", "*класс действий*" и "*класс задач*" рассмотрено в *Главе 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*. Рассмотрим указанные соотношения в контексте решения задач в *sc-памяти*.

задача, решаемая в sc-памяти
⊂ *задача*
:= [спецификация действия, выполняемого в sc-памяти]
:= [структура, являющая таким описанием (постановкой, заданием) соответствующего действия в sc-памяти, которое обладает достаточной полнотой для выполения указанного действия]
:= [семантическая окрестность некоторого действия в sc-памяти, обеспечивающая достаточно полное задание этого действия]

класс действий
⊃ *класс действий в sc-памяти*

⇐ *семейство подмножеств**:
действие в sc-памяти
⇒ *разбиение**:
{• *класс логически атомарных действий*
:= [класс автономных действий]
⊃ *класс логически атомарных действий в sc-памяти*
• *класс логически неатомарных действий*
:= [класс неавтономных действий]
}

Каждое *действие*, принадлежащее некоторому конкретному *классу логически атомарных действий*, обладает двумя необходимыми свойствами:

- выполнение действия не зависит от того, является ли указанное действие частью декомпозиции более общего действия. При выполнении данного действия также не должен учитываться тот факт, что данное действие предшествует каким-либо другим действиям или следует за ними (что явно указывается при помощи отношения *последовательность действий**);
- указанное действие должно представлять собой логически целостный акт преобразования, например, в семантической памяти. Такое действие по сути является транзакцией, то есть результатом такого преобразования становится новое состояние преобразуемой системы, а выполняемое действие должно быть либо выполнено полностью, либо не выполнено совсем, частичное выполнение не допускается.

В то же время логическая атомарность не запрещает декомпозировать выполняемое действие на более частные, каждое из которых, в свою очередь, также будет являться логически атомарным.

На логически *атомарные действия* предлагается делить всю деятельность, направленную на решение каких-либо задач *ostis-системой*. Соответственно *решатель задач ostis-системы* предлагается делить на компоненты, соответствующие таким *классам логически атомарных действий в sc-памяти*, что является основой для обеспечения его *модифицируемости*. Такие компоненты решателя названы *sc-агентами*.

В свою очередь под *методом* понимается описание того, как может быть выполнено любое или почти любое (с явным указанием исключений) действие, принадлежащее соответствующему *классу действий*. Поскольку конкретному *классу действий* ставится в соответствие некоторый конкретный *класс задач*, то можно сказать, что метод описывает способ решения любых задач принадлежащих заданному классу. Понятие метода можно считать обобщением понятия "программа", в связи с чем в рамках *Технологии OSTIS* термины "метод" и "программа" являются синонимичными (см. *Главу 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*, *Главу 3.2. Семантическая теория программ для ostis-систем*).

Примером конкретного метода может быть *процедурная программа* на конкретном *языке программирования*, или множество логических высказываний, составляющих *формальную теорию* заданной *предметной области* (аналог *логической программы*).

Частным случаем метода является программа атомарного компонента *решателя задач ostis-системы* (*атомарного sc-агента*, см. *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*), в этом случае в качестве *операционной семантики метода* выступает коллектив агентов более низкого уровня, интерпретирующий соответствующую *программу* (в предельном случае это будут агенты, являющиеся частью *ostis-платформы*, в том числе аппаратной реализации).

В свою очередь, навык трактуется как объединение некоторого *метода* и его операционной семантики, то есть информации о том, каким образом должен интерпретироваться данный *метод*.

Таким образом, можно говорить об иерархии *методов* и о *методах* интерпретации других *методов*, а также, соответственно, об иерархии *навыков*, которыми обладает *ostis-система* на данный момент времени.

§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты

⇒ *ключевое понятие**:
- *sc-агент*
- *абстрактный sc-агент*
- *атомарный абстрактный sc-агент*
- *неатомарный абстрактный sc-агент*
- *абстрактный sc-агент, реализуемый на Языке SCP*
- *абстрактный sc-агент, не реализуемый на Языке SCP*

Как было сказано выше, *решатель задач ostis-системы* предлагается делить на компоненты, соответствующие *классам логически атомарных действий в sc-памяти*, называемые *sc-агентами*.

sc-агент
:= [единственный вид *субъектов*, выполняющих преобразования в *sc-памяти*]
:= [*субъект*, способный выполнять *действия в sc-памяти*, принадлежащие некоторому определенному *классу логически атомарных действий*]

Логическая атомарность выполняемых *sc-агентом действий* предполагает, что каждый *sc-агент* реагирует на соответствующий ему класс *ситуаций* и/или *событий*, происходящих в sc-памяти, и осуществляет определенное преобразование *sc-текста*, находящегося в семантической окрестности обрабатываемой *ситуации* и/или *события*. При этом каждый sc-агент в общем случае не имеет информацию о том, какие еще *sc-агенты* в данный момент присутствуют в системе и осуществляет взаимодействие в другими *sc-агентами* исключительно посредством формирования некоторых конструкций (как правило — спецификаций действий) в общей *sc-памяти*. Таким сообщением может быть, например, вопрос, адресованный другим *sc-агентам* в системе (заранее не известно, каким конкретно), или ответ на поставленный другими *sc-агентами* вопрос (заранее не известно, каким конкретно). Таким образом, каждый *sc-агент* в каждый момент времени контролирует только фрагмент *базы знаний* в контексте решаемой данным агентом *задачи*, состояние всей остальной *базы знаний* в общем случае непредсказуемо для *sc-агента*.

Поскольку предполагается, что копии одного и того же *sc-агента* или функционально эквивалентные *sc-агенты* могут работать в разных *ostis-системах*, будучи при этом физически разными *sc-агентами*, то целесообразно рассматривать свойства и классификацию не *sc-агентов*, а классов функционально эквивалентных *sc-агентов*, которые будем называть ***абстрактными sc-агентами***. Под ***абстрактным sc-агентом*** понимается некоторый класс функционально эквивалентных *sc-агентов*, разные экземпляры (то есть представители) которого могут быть реализованы по-разному.

Каждый ***абстрактный sc-агент*** имеет соответствующую ему спецификацию. В спецификацию каждого ***абстрактного sc-агента*** входит:
- указание ключевых *sc-элементов* этого *sc-агента*, то есть тех *sc-элементов*, хранимых в *sc-памяти*, которые для данного *sc-агента* являются «точками опоры»;
- формальное описание условий инициирования данного *sc-агента*, то есть тех *ситуаций* в *sc-памяти*, которые инициируют деятельность данного *sc-агента*;
- формальное описание первичного условия инициирования данного *sc-агента*, то есть такой *ситуации в sc-памяти*, которая побуждает *sc-агента* перейти в активное состояние и начать проверку наличия своего полного условия инициирования (для *внутренних абстрактных sc-агентов*);
- строгое, полное, однозначно понимаемое описание деятельности данного *sc-агента*, оформленное при помощи каких-либо понятных, общепринятых средств, не требующих специального изучения, например на *естественном языке*.
- описание результатов выполнения данного *sc-агента*.

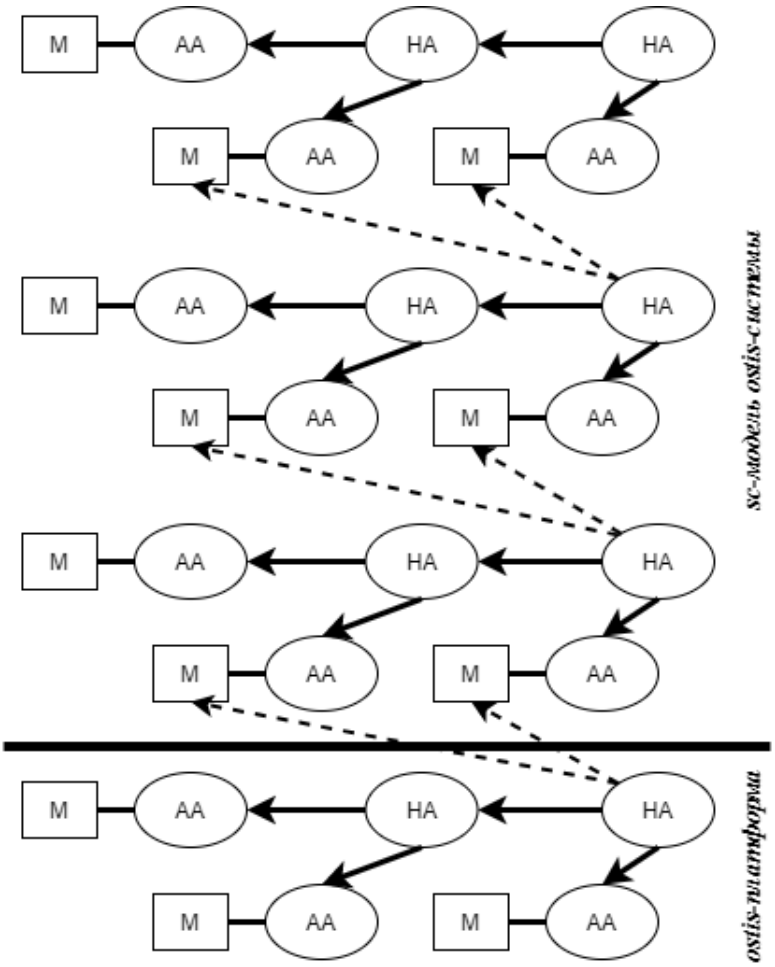
sc-агенты можно классифицировать по различным признакам. Поскольку можно говорить об иерархии *методов* (методах интерпретации других методов) и, соответственно, иерархии навыков, то есть необходимость говорить и об иерархии *sc-агентов*, обеспечивающих интерпретацию того или иного метода. В данном контексте можно говорить об иерархии *sc-агентов* в двух аспектах:
- *абстрактному sc-агенту* (и соответственно, *sc-агенту*) может однозначно соответствовать *метод* (*программа sc-агента*), описывающий деятельность данного *sc-агента*. Такие агенты будем называть *атомарными абстрактными sc-агентами*;
- *абстрактные sc-агенты* иногда целесообразно объединять в коллективы таких агентов, которые можно рассматривать как один целостный *абстрактный sc-агент*, с логической точки зрения работающий по тем же принципам, что и *атомарные абстрактные sc-агенты*, то есть реагирующий на события в *sc-памяти* и описывающий свою деятельность в рамках этой памяти. Такому *абстрактному sc-агенту* не будет соответствовать какой-то конкретный *метод*, хранимый в *sc-памяти*, но остальная часть спецификации *абстрактного sc-агента* (условие инициирования, описание начальной ситуации и результата работы *sc-агента* и так далее) остается такой же, как у *атомарного абстрактного sc-агента*. Таким образом, можно сказать, что понятие атомарности/неатомарности *абстрактного sc-агента* указывает на то, каким образом уточняется реализация данного *абстрактного sc-агента* – посредством указания конкретного метода (программы *sc-агента*) или посредством декомпозиции *абстрактного sc-агента* на более простые. Важно отметить, что *неатомарные абстрактные sc-агенты* тоже могут входить в состав других, более сложных *неатомарных абстрактных sc-агентов*. Таким образом формируется иерархическая система *абстрактных sc-агентов*, в общем случае имеющая произвольное количество уровней.
- В свою очередь, соответствующий *sc-агенту* метод должен интерпретироваться каким-либо другим *sc-агентом* более низкого уровня, а чаще всего – коллективом таких агентов, каждому из которых ставится

в соответствие свой *метод*, описывающий поведение данного агента, но уже на более низком уровне. Таким образом, можно сказать, что понятие атомарности/неатомарности *абстрактных sc-агентов* применимо в рамках одного *языка описания методов*. В свою очередь можно говорить об иерархии *абстрактных sc-агентов* с точки зрения уровня языка описания соответствующих таким агентам методов. В общем случае такая иерархия тоже может иметь неограниченное число уровней, однако очевидно, что при понижении уровня языка описания методов мы рано или поздно должны подойти к языку описания методов, который будет интерпретироваться агентами, реализуемыми на уровне *ostis-платформы*, а спускаясь еще ниже – на уровень языка описания методов, интерпретируемых на аппаратном уровне. Таким образом, для обеспечения платформенной независимости *ostis-систем* целесообразно выделить такой язык описания методов, который бы интерпретировался на уровне *ostis-платформы* и являлся основой для разработки интерпретаторов более высокоуровневых языков. В качестве такого языка предлагается *Язык SCP* (Semantic Code Programming), который рассматривается в качестве ассемблера для *ассоциативного семантического компьютера*.

На рисунке *Рисунок. Иерархия sc-агентов* проиллюстрирована иерархия *абстрактных sc-агентов* и соответствующим *атомарным абстрактным sc-агентам методов*. Буквой “М” на рисунке условно обозначены *методы*, буквами “АА” и “НА” – *атомарные абстрактные sc-агенты* и *неатомарные абстрактные sc-агенты* соответственно, сплошными стрелками показана декомпозиция *неатомарных sc-агентов* на более простые, а пунктирными стрелками – связь между *методами* и их операционной семантикой, то есть *абстрактными sc-агентами*, обеспечивающими интерпретацию этих *методов*. Как показано на приведенной иллюстрации, должна существовать четкая граница между методами, которые описываются на уровне *ostis-платформы*, и методами, которые могут быть описаны и на платформенно-независимом уровне. Более подробно этот вопрос рассматривается в *Главе 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*.

Рисунок. Иерархия sc-агентов

=



Иерархический подход к описанию систем sc-агентов, определяющих операционную семантику *решателей задач ostis-систем*, и соответственно, самих *решателей задач ostis-систем* обладает рядом важных преимуществ, таких как обеспечение модифицируемости решателей и удобство их проектирования и отладки на разных уровнях.

Сказанное выше позволяет описать классификацию *абстрактных sc-агентов* по признаку атомарности:

абстрактный sc-агент
⇒ *разбиение**:
{• *неатомарный абстрактный sc-агент*
• *атомарный абстрактный sc-агент*
}

Под ***неатомарным абстрактным sc-агентом*** понимается *абстрактный sc-агент*, который декомпозируется на коллектив более простых *абстрактных sc-агентов*, каждый из которых в свою очередь может быть как *атомарным абстрактным sc-агентом*, так и ***неатомарным абстрактным sc-агентом***. При этом в каком либо варианте *декомпозиции абстрактного sc-агента** дочерний ***неатомарный абстрактный sc-агент*** может стать *атомарным абстрактным sc-агентом*, и реализовываться соответствующим образом.

Под ***атомарным абстрактным sc-агентом*** понимается *абстрактный sc-агент*, для которого уточняется способ его реализации, то есть существует соответствующая связка отношения *программа sc-агента**. Подчеркнем при этом, что в рамках конкретной *ostis-системы* для каждого *языка представления методов* может существовать своя иерархия *абстрактных sc-агентов*.

В свою очередь, *Язык SCP* позволяет установить границу между логико-семантической моделью *ostis-системы* и *ostis-платформой*. В связи с этим будем считать платформенно-независимыми *абстрактные sc-агенты*, реализованные на *Языке SCP* или более высокоуровневых языках на его основе, а платформенно-зависимыми *абстрактные sc-агенты*, которые реализованы на уровне *ostis-платформы* (например, с целью повышения их производительности). В то же время существует ряд *абстрактных sc-агентов*, которые принципиально не могут быть реализованы на *Языке SCP*. Сказанное отражено в следующей иерархии:

абстрактный sc-агент
⇒ *разбиение**:
{• *внутренний абстрактный sc-агент*
• *эффекторный абстрактный sc-агент*
• *рецепторный абстрактный sc-агент*
}
⇒ *разбиение**:
{• *абстрактный sc-агент, не реализуемый на Языке SCP*
• *абстрактный sc-агент, реализуемый на Языке SCP*
}
⇒ *разбиение**:
{• *абстрактный sc-агент интерпретации scp-программ*
• *абстрактный программный sc-агент*
• *абстрактный sc-метаагент*
}
⇒ *разбиение**:
{• *платформенно-зависимый абстрактный sc-агент*
⊃ *абстрактный sc-агент, не реализуемый на Языке SCP*
• *платформенно-независимый абстрактный sc-агент*
}

абстрактный sc-агент
⇒ *разбиение**:
{• *абстрактный sc-агент, не реализуемый на Языке SCP*
• *абстрактный sc-агент, реализуемый на Языке SCP*
}
⇒ *разбиение**:
{• *платформенно-зависимый абстрактный sc-агент*
⊃ *абстрактный sc-агент, не реализуемый на Языке SCP*
• *платформенно-независимый абстрактный sc-агент*
}

абстрактный sc-агент, не реализуемый на Языке SCP
:= [абстрактный sc-агент, который не может быть реализован на платформенно-независимом уровне]
⇒ *разбиение**:
{• *эффекторный абстрактный sc-агент*
• *рецепторный абстрактный sc-агент*

- *абстрактный sc-агент интерпретации scp-программ*

}

абстрактный sc-агент, реализуемый на Языке SCP
:= [абстрактный sc-агент, который может быть реализован на платформенно-независимом уровне]
⇒ *разбиение**:
{• *абстрактный sc-метаагент*
• *абстрактный программный sc-агент, реализуемый на Языке SCP*
}

абстрактный программный sc-агент
⇒ *разбиение**:
{• *эффекторный абстрактный sc-агент*
• *рецепторный абстрактный sc-агент*
• *абстрактный программный sc-агент, реализуемый на Языке SCP*
}

атомарный абстрактный sc-агент
⇒ *разбиение**:
{• *платформенно-независимый абстрактный sc-агент*
• *платформенно-зависимый абстрактный sc-агент*
}

К ***платформенно-независимым абстрактным sc-агентам*** относят *атомарные абстрактные sc-агенты*, реализованные на базовом языке программирования *Технологии OSTIS*, то есть на *Языке SCP*.

При описании ***платформенно-независимых абстрактных sc-агентов*** под *платформенной независимостью* понимается платформенная независимость с точки зрения *Технологии OSTIS*, то есть реализация на специализированном *языке программирования*, ориентированном на обработку семантических сетей (*Языке SCP*), поскольку *атомарные sc-агенты*, реализованные на указанном языке могут свободно переноситься с одной *ostis-платформы* на другую. При этом языки программирования, традиционно считающиеся платформенно-независимыми, в данном случае не могут считаться таковыми.

Существуют *sc-агенты*, которые принципиально не могут быть реализованы на платформенно-независимом уровне, например, собственно *sc-агенты* интерпретации *sc-моделей* или рецепторные и эффекторные *sc-агенты*, обеспечивающие взаимодействие с внешней средой.

К ***платформенно-зависимым абстрактным sc-агентам*** относят *атомарные абстрактные sc-агенты*, реализованные ниже уровня sc-моделей, то есть не на *Языке SCP*, а на каком-либо другом языке описания *программ*.

Каждый ***внутренний абстрактный sc-агент*** обозначает класс *sc-агентов*, которые реагируют на события в *sc-памяти* и осуществляют преобразования исключительно в рамках этой же *sc-памяти*.

Каждый ***эффекторный абстрактный sc-агент*** обозначает класс *sc-агентов*, которые реагируют на события в *sc-памяти* и осуществляют преобразования во внешней относительно данной *ostis-системы* среде.

Каждый ***рецепторный абстрактный sc-агент*** обозначает класс *sc-агентов*, которые реагируют на события во внешней относительно данной *ostis-системы* среде и осуществляют преобразования в памяти данной системы.

Каждый ***абстрактный sc-агент, не реализуемый на Языке SCP*** должен быть реализован на уровне *ostis-платформы*, в том числе, аппаратной. К таким *абстрактным sc-агентам* относятся *абстрактные sc-агенты интерпретации scp-программ*, а также *эффекторные* и *рецепторные абстрактные sc-агенты*.

Каждый ***абстрактный sc-агент, реализуемый на Языке SCP*** может быть реализован на *Языке SCP*, то есть платформенно-независимом уровне, но при необходимости, может реализовываться и на уровне *ostis-платформы*, например, с целью повышения производительности.

К ***абстрактным sc-агентам интерпретации scp-программ*** относятся не реализуемые на платформенно-независимом уровне *абстрактные sc-агенты*, обеспечивающие интерпретацию *scp-программ* и *scp-метапрограмм*, в том числе создание *scp-процессов*, собственно интерпретацию *scp-операторов*, а также другие вспомогательные действия. По сути, агенты данного класса обеспечивают работу sc-агентов более высоких уровней (*программных sc-агентов* и *sc-метаагентов*), реализованных *на Языке SCP*, в частности, обеспечивают соблюдение указанными *агентами* общих принципов синхронизации.

К ***абстрактным программным sc-агентам*** относятся все *абстрактные sc-агенты*, обеспечивающие основной функционал системы, то есть ее возможность решать те или иные задачи. Агенты данного класса должны работать в соответствии с общими принципами синхронизации деятельности субъектов в sc-памяти.

Задачей ***абстрактных sc-метаагентов*** является координация деятельности *абстрактных программных sc-агентов*, в частности, решение проблемы взаимоблокировок. Агенты данного класса могут быть реализованы на *Языке SCP*, однако для синхронизации их деятельности используются другие принципы, соответственно, для реализации таких агентов требуется Язык SCP другого уровня, типология операторов которого полностью аналогична типологии scp-операторов, однако эти операторы имеют другую операционную семантику, учитывающую отличия в принципах синхронизации (работы с *блокировками**). Программы такого языка будем называть *scp-метапрограммами*, соответствующие им *процессы в sc-памяти – scp-метапроцессами*, операторы – *scp-метаоператорами*.

декомпозиция абстрактного sc-агента*
∈ *отношение декомпозиции*
⇒ *первый домен**:
неатомарный абстрактный sc-агент

Отношение ***декомпозиции абстрактного sc-агента**** трактует *неатомарные абстрактные sc-агенты* как коллективы более простых *абстрактных sc-агентов*, взаимодействующих через *sc-память*.

Другими словами, ***декомпозиция абстрактного sc-агента**** на *абстрактные sc-агенты* более низкого уровня уточняет один из возможных подходов к реализации этого *абстрактного sc-агента* путем построения коллектива более простых *абстрактных sc-агентов*.

sc-агент
:= [агент над sc-памятью]
⊂ *субъект*
⇒ *семейство подмножеств**:
абстрактный sc-агент

Под ***sc-агентом*** понимается конкретный экземпляр (с теоретико-множественной точки зрения - элемент) некоторого *атомарного абстрактного sc-агента*, работающий в какой-либо конкретной интеллектуальной системе.

Таким образом, каждый *sc-агент* - это субъект, способный выполнять некоторый класс однотипных действий либо только над *sc-памятью*, либо над sc-памятью и внешней средой (для эффекторных *sc-агентов*). Каждое такое действие инициируется либо состоянием или ситуацией в sc-памяти, либо состоянием или ситуацией во внешней среде (для рецепторных sc-агентов-датчиков), соответствующей условию инициирования *атомарного абстрактного sc-агента*, экземпляром которого является заданный *sc-агент*. В данном случае можно провести аналогию между принципами объектно-ориентированного программирования, рассматривая *атомарный абстрактный sc-агент* как класс, а конкретный *sc-агент* — как экземпляр, конкретную имплементацию этого класса.

Взаимодействие *sc-агентов* осуществляется только через *sc-память*. Как следствие, результатом работы любого *sc-агента* является некоторое изменение состояния *sc-памяти*, то есть удаление либо генерация каких-либо *sc-элементов*.

В общем случае один *sc-агент* может явно передать управление другому *sc-агенту*, если этот *sc-агент* априори известен. Для этого каждый *sc-агент* в *sc-памяти* имеет обозначающий его *sc-узел*, с которым можно связать конкретную ситуацию в текущем состоянии *базы знаний*, которую инициируемый *sc-агент* должен обработать.

Однако далеко не всегда легко определить того *sc-агента*, который должен принять управление от заданного *sc-агента*, в связи с чем описанная выше ситуация возникает крайне редко. Более того, иногда условие инициирования *sc-агента* является результатом деятельности непредсказуемой группы *sc-агентов*, равно как и одна и та же конструкция может являться условием инициирования целой группы *sc-агентов*.

При этом общаются через *sc-память* не *программы sc-агентов**, а сами описываемые данными программами *sc-агенты*.

В процессе работы *sc-агент* может сам для себя порождать вспомогательные *sc-элементы*, которые сам же удаляет после завершения акта своей деятельности (это вспомогательные *структуры*, которые используются в качестве "информационных лесов" только в ходе выполнения соответствующего акта деятельности и после завершения этого акта удаляются).

sc-агент
⊃ *активный sc-агент*
⇒ *первый домен**:
- *ключевые sc-элементы sc-агента**
- *программа sc-агента**
- *первичное условие инициирования**
- *условие инициирования и результат**

Под ***активным sc-агентом*** понимается *sc-агент* ostis-системы, который реагирует на события, соответствующие его условию инициирования, и, как следствие, его *первичному условию инициирования**. Не входящие во множество ***активных sc-агентов*** *sc-агенты* не реагируют ни на какие события в *sc-памяти*.

Связки отношения ***ключевые sc-элементы sc-агента**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и *sc-узел*, обозначающий множество *sc-элементов*, которые являются ключевыми для данного *абстрактного sc-агента*, то данные *sc-элементы* явно упоминаются в рамках программ, реализующих данный *абстрактный sc-агент*.

Связки отношения ***программа sc-агента**** связывают между собой *sc-узел*, обозначающий *атомарный абстрактный sc-агент* и *sc-узел*, обозначающий множество программ, реализующих указанный *атомарный абстрактный sc-агент*. В случае *платформенно-независимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел*, обозначающий указанный *абстрактный sc-агент* с множеством *scp-программ*, описывающих деятельность данного *абстрактного sc-агента*. Данное множество содержит одну *агентную scp-программу*, и произвольное количество (может быть, и ни одной) *scp-программ*, которые необходимы для выполнения указанной *агентной scp-программы*.

В случае *платформенно-зависимого абстрактного sc-агента* каждая связка отношения *программа sc-агента** связывает *sc-узел*, обозначающий указанный *абстрактный sc-агент* с множеством файлов, содержащих исходные тексты программы на некотором внешнем языке программирования, реализующей деятельность данного *абстрактного sc-агента*.

Связки отношения ***первичное условие инициирования**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, описывающую первичное условие инициирования данного *абстрактного sc-агента*, то есть такой спецификацию *ситуации* в *sc-памяти*, возникновение которой побуждает *sc-агента* перейти в активное состояние и начать проверку наличия своего полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого класса *элементарных событий в sc-памяти**, например, *событие добавления sc-дуги, выходящей из заданного sc-элемента**.

Вторым компонентом данной ориентированной пары является произвольный в общем случае *sc-элемент*, с которым непосредственно связан указанный тип события в *sc-памяти*, то есть, например, *sc-элемент*, из которого выходит либо в который входит генерируемая либо удаляемая *sc-дуга*, либо *файл*, содержимое которого было изменено.

После того, как в *sc-памяти* происходит некоторое событие, активизируются все *активные sc-агенты*, ***первичное условие инициирования**** которых соответствует произошедшему событию.

Связки отношения ***условие инициирования и результат**** связывают между собой *sc-узел*, обозначающий *абстрактный sc-агент* и бинарную ориентированную пару, связывающую условие инициирования данного *абстрактного sc-агента* и результаты выполнения данного экземпляров данного *sc-агента* в какой-либо конкретной системе.

Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на *sc-переменные*, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на *sc-переменные*, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого *абстрактного sc-агента*, то есть конструкции, наличие которой в *sc-памяти* побуждает *sc-агент* начать работу по изменению состояния *sc-памяти*. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Вторым компонентом указанной ориентированной пары является *логическая формула*, описывающая возможные результаты выполнения описываемого абстрактного *sc-агента*, то есть описание произведенных им изменений состояния *sc-памяти*. Данная *логическая формула* может быть как атомарной, так и неатомарной, в которой допускается использование любых связок *логического языка*.

описание поведения sc-агента
⊂ *семантическая окрестность*

описание поведения sc-агента представляет собой *семантическую окрестность*, описывающую деятельность *sc-агента* до какой-либо степени детализации, однако такое описание должно быть строгим, полным и однозначно понимаемым. Как любая другая *семантическая окрестность*, ***описание поведения sc-агента*** может быть протранслировано на какие-либо понятные, общепринятые средства, не требующие специального изучения, например на *естественный язык*.

Описываемый *абстрактный sc-агент* входит в соответствующее ***описание поведения sc-агента*** под атрибутом *ключевой sc-элемент′*.

§ 3.3.4. Принципы синхронизации деятельности sc-агентов

⇒ *ключевое понятие**:
- *действие в sc-памяти*
- *блокировка**
- *тип блокировки*
- *планируемая блокировка**
- *приоритет блокировки**
- *удаляемые sc-элементы**
- *транзакция в sc-памяти*

Одной из важных особенностей *многоагентного подхода* к решению задач является возможность параллельного решения различных задач, что в свою очередь, предполагает параллельность выполнения соответствующих информационных процессов.

Понятия *действие в sc-памяти*, и *процесс в sc-памяти* (*информационный процесс*, выполняемый агентом в семантической памяти), являются синонимичными, поскольку все процессы, протекающие в sc-памяти, являются осознанными и выполняются каким-либо *sc-агентами*. Тем не менее, когда идет речь о синхронизации выполнения каких-либо преобразований в памяти компьютерной системы, в литературе принято использовать именно термины “процесс”, “взаимодействие процессов” (см. *Dijkstra E.W.CoopeSP-2002bk*, *Hoare C.A.CommuSP-1983art*), в связи с чем будем использовать этот термин при описании принципов синхронизации деятельности sc-агентов при выполнении ими параллельных *процессов в sc-памяти*.

действие в sc-памяти
:= *часто используемый sc-идентификатор**:
[*процесс в sc-памяти*]

процесс в sc-памяти
⇒ *разбиение**:
{• *процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту*
• *scp-процесс*
⇒ *разбиение**:
{• *scp-процесс, не являющийся scp-метапроцессом*
• *scp-метапроцесс*
}
}

процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту
⇒ *разбиение**:
{• *процесс в sc-памяти, соответствующий платформенно-зависимому sc-агенту и не являющийся действием абстрактной scp-машины*
• *действие абстрактной scp-машины*
⊃ *действие интерпретации scp-программы*
}

Для синхронизации выполнения *процессов в sc-памяти* предлагается использовать механизм блокировок, построенный на основе существующих алгоритмов синхронизации информационных процессов в традиционных системах (см. *Dijkstra E.W.CoopeSP-2002bk*, *Hoare C.A.CommuSP-1983art*). В качестве возможного направления развития данного подхода можно указать набирающие популярность идеи lock-free алгоритмов (см. *Chatterjee B..nBlockDUGwWCAB-2022art*).

Отношение ***блокировка**** связывает знаки *действий в sc-памяти* со знаками *структур* (ситуативных), которые содержат элементы, заблокированные на время выполнения данного действия или на какую-то часть этого периода. Каждая такая *структура* принадлежит какому-либо из *типов блокировки*.

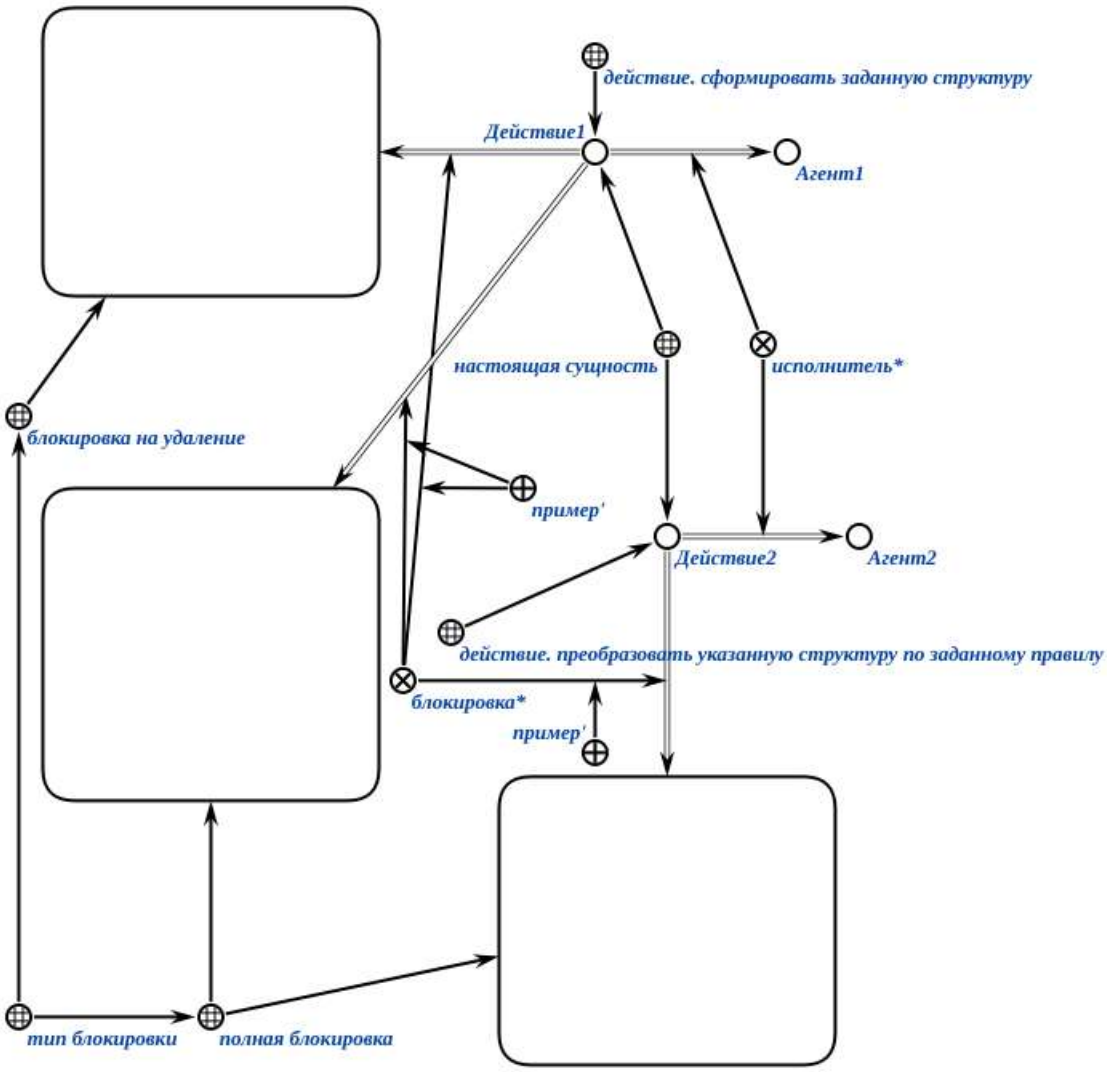
Первым компонентом связок отношения ***блокировка**** является знак *действия в sc-памяти*, вторым — знак заблокированной *структуры*.

блокировка*
∈ *бинарное отношение*

тип блокировки
∋ *полная блокировка*
∋ *блокировка на любое изменение*

∋ *блокировка на удаление*

SCg-текст. Пример использования блокировок
=



Множество ***тип блокировки*** содержит все возможные классы блокировок, то есть *структуры*, содержащие *sc-элементы*, заблокированные каким-либо *sc-агентом* на время выполнения им некоторого *действия в sc-памяти*.

Каждая *структура*, принадлежащая множеству ***полная блокировка*** содержит *sc-элементы*, просмотр и изменение (удаление, добавление инцидентных *sc-коннекторов*, удаление самих *sc-элементов*, изменение содержимого в случае файла) которых запрещены всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**.

Для того, чтобы исключить возможность реализации *sc-агентов*, которые могут внести изменения в конструкции, описывающие блокировки других *sc-агентов*, все элементы этих конструкций, в том числе, сам знак *структуры*, содержащей заблокированные *sc-элементы* (принадлежащей как множеству ***полная блокировка***, так и любому другому *типу блокировки*) и связки отношения *блокировка**, связывающие эту *структуру* и конкретное *действие в sc-памяти*, добавляются в ***полную блокировку***, соответствующую данному *действию в sc-памяти*. Таким образом, каждой ***полной блокировке*** соответствует петля принадлежности, связывающая ее знак с самим собой.

Каждая *структура*, принадлежащая множеству ***блокировка на любое изменение*** содержит *sc-элементы*, изменение (физическое удаление, добавление инцидентных *sc-коннекторов*, физическое удаление самих *sc-элементов*, изменение содержимого в случае файла) которых запрещено всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**. Однако не запрещен просмотр (чтение) этих *sc-элементов* любым *sc-агентом*.

Каждая *структура*, принадлежащая множеству ***блокировка на удаление*** содержит *sc-элементы*, удаление которых запрещено всем *sc-агентам*, кроме собственно *sc-агента*, выполняющего соответствующее данной структуре *действие в sc-памяти*, связанное с ней отношением *блокировка**. Однако не запрещен просмотр (чтение) этих *sc-элементов* любым *sc-агентом*, добавление инцидентных sc-коннекторов.

Рассмотрим принципы работы с блокировками:
- в каждый момент времени одному *процессу в sc-памяти* может соответствовать только одна блокировка каждого типа;
- в каждый момент времени одному *процессу в sc-памяти* может соответствовать только одна блокировка, установленная на некоторый конкретный sc-элемент;
- при завершении выполнения любого *процесса в sc-памяти* все установленные им блокировки автоматически снимаются;
- для повышения эффективности работы системы в целом каждый процесс должен в каждый момент времени блокировать минимально необходимое множество sc-элементов, снимая блокировку с каждого sc-элемента сразу же, как это становится возможным (безопасным);
- В случае когда в рамках *процесса в sc-памяти* явно выделяются более частные подпроцессы (при помощи отношений *темпоральная часть*, поддействие*, декомпозиция действия** и так далее), то каждый такой подпроцесс с точки зрения синхронизации выполнения рассматривается как самостоятельный процесс, которому в соответствие могут быть поставлены все необходимые блокировки.
  - все дочерние процессы в sc-памяти имеют доступ к блокировкам родительского процесса так же, как если бы это были блокировки соответствующие каждому из таких дочерних процессов;
  - в свою очередь, родительский процесс не имеет какого-либо привилегированного доступа к sc-элементам, заблокированным дочерними процессами, и работает с ними так же, как любой другой процесс в sc-памяти. Исключение составляют sc-элементы, обозначающие сами дочерние процессы, поскольку родительский процесс должен иметь возможность управления дочерним, например, приостановки или прекращения их выполнения;
  - все дочерние процессы по отношению друг к другу работают так же, как и по отношению к любым другим процессам;
  - в случае, когда родительский процесс приостанавливает выполнение (становится *отложенным действием*), все его дочерние процессы также приостанавливают выполнение. В свою очередь, приостановка одного из дочерних процессов в общем случае не инициирует явно остановку всего родительского процесса и соответственно других дочерних.

Рассмотрим принципы работы с *полными блокировками*:
- если sc-элемент, инцидентный некоторому sc-коннектору, попадает в какую-либо полную блокировку, то сам этот sc-коннектор по умолчанию также считается заблокированным этой же блокировкой. Обратное в общем случае неверно, так как часть sc-коннекторов, инцидентных некоторому sc-элементу, может быть полностью заблокирована, при этом сам этот элемент заблокирован не будет. Такая ситуация типична, например, для sc-узлов, обозначающих классы понятий;
- каждый процесс в sc-памяти может свободно изменять или удалять любые sc-элементы, попадающие в полную блокировку, соответствующую этому процессу.

Принципы работы с *полными блокировками*, с одной стороны, наиболее просты, поскольку все процессы, кроме установившего такую блокировку, не имеют доступа к заблокированным sc-элементам и конфликты возникнуть не могут. С другой стороны, частое использование блокировок такого типа может привести к тому, что система не сможет использовать в полной мере имеющиеся у нее знания и давать неполные или даже некорректные ответы на поставленные вопросы.

Рассмотрим принципы работы с *блокировками на любое изменение* и *блокировками на удаление*:
- на один и тот же sc-элемент в один момент времени может быть установлена только одна блокировка одного типа, но разные процессы могут одновременно установить на один и тот же элемент блокировки двух разных типов. Это касается случая, когда первый процесс установил на некоторый sc-элемент блокировку на удаление, а второй процесс затем устанавливает блокировку на любое изменение. В других случаях возникает конфликт блокировок;
- установка блокировки любого типа также считается изменением, таким образом, если на некоторый sc-элемент была установлена блокировка на любое изменение, то другой процесс не сможет установить на этот же sc-элемент блокировку любого типа, пока первый процесс не снимет свою;
- если блокировка на удаление устанавливается на некоторый sc-коннектор, то по умолчанию та же блокировка устанавливается на инцидентные этому sc-коннектору sc-элементы, поскольку удаление этих элементов приведет к удалению этого коннектора.

процесс в sc-памяти
⇒ *разбиение**:

Классификация процессов в sc-памяти с точки зрения синхронизации их выполнения
= {• *действие поиска sc-элементов*
• *действие генерации sc-элементов*
• *действие удаления sc-элементов*
• *действие установки блокировки некоторого типа на некоторый sc-элемент*
• *действие снятия блокировки с некоторого sc-элемента*
}

В некоторых случаях для того, чтобы обеспечить синхронизацию, необходимо объединять несколько элементарных действий над sc-памятью в одно неделимое действие (***транзакцию в sc-памяти***), для которого гарантируется, что ни один сторонний процесс не сможет прочитать или изменить участвующие в этом действии sc-элементы, пока действие не завершится. При этом, в отличие от ситуации с полной блокировкой, процесс, пытающийся получить доступ к таким элементам, не продолжает выполнение так, как если бы этих элементов просто не было в sc-памяти, а ожидает завершения транзакции, после чего может выполнять с данными элементами любые действия согласно общим принципам синхронизации процессов. Проблема обеспечения транзакций не может быть решена на уровне SC-кода и требует реализации таких неделимых действий на уровне *ostis-платформы*.

В случае выполнения *действия поиска sc-элементов* все найденные и сохраненные в рамках какого-либо процесса sc-элементы попадают в соответствующую данному процессу *блокировку на любое изменение*. Таким образом, гарантируется целостность фрагмента базы знаний, с которым работает некоторый процесс в sc-памяти. При этом поиск и автоматическая установка такой блокировки должны быть реализованы как *транзакция в sc-памяти*.

Такой подход также позволяет избежать ситуации, когда один процесс заблокировал некоторый sc-элемент на любое изменение, а второй процесс пытается сгенерировать или удалить *sc-коннектор*, инцидентный данному *sc-элементу*. В таком случае второй процесс должен будет предварительно найти и заблокировать указанный *sc-элемент* на любое изменение, что вызовет конфликт блокировок (*взаимоблокировку**).

В случае генерации любого sc-элемента в рамках некоторого процесса он автоматически попадает в полную блокировку, соответствующую данному процессу. При этом генерация и автоматическая установка такой блокировки должны быть реализованы как *транзакция в sc-памяти*. При необходимости сгенерированные элементы могут быть удалены (то есть их временное существование вообще никак не отразится на деятельности других процессов) или разблокированы в случае, когда сгенерирована информация, которая может иметь некоторую ценность в дальнейшем.

В случае если какой-либо процесс пытается установить блокировку любого типа на какой-либо sc-элемент, уже заблокированный каким-либо другим процессом, то, с одной стороны, блокировка не может быть установлена, пока другой процесс не разблокирует указанный sc-элемент; с другой стороны, для того чтобы обеспечить возможность поиска и устранения *взаимоблокировок*, необходимо явно указывать тот факт, что какой-либо процесс хочет получить доступ к какому-либо заблокированному другим процессом sc-элементу. Для того чтобы иметь возможность указать, какие процессы пытаются заблокировать уже заблокированный *sc-элемент*, предлагается наряду с отношением *блокировка** использовать отношение *планируемая блокировка**, полностью аналогичное отношению *блокировка**.

Описанный механизм регулирует также и процессы поиска, поскольку поиск и сохранение некоторого sc-элемента предполагает установку *блокировки на любое изменение*. Кроме того, следует учитывать, что на один sc-элемент *блокировка на любое изменение* может быть установлена после *блокировки на удаление*, соответствующей другому процессу. В этом случае использовать отношение *планируемые блокировки** нет необходимости.

Действие проверки наличия на некотором sc-элементе блокировки и в зависимости от результата проверки, установки блокировки или планируемой блокировки (с указанием приоритета при необходимости) должно быть реализовано как транзакция.

планируемая блокировка*
⊂ *блокировка**

Процесс, которому в соответствие поставлена *планируемая блокировка**, приостанавливает выполнение до тех пор, пока уже установленные блокировки не будут сняты, после чего *планируемая блокировка** становится реальной *блокировкой** и процесс продолжает выполнение в соответствии с общими правилами.

приоритет блокировки*
⇒ *область определения**:
*планируемая блокировка**

В случае, когда на один и тот же sc-элемент планируют установить блокировку сразу несколько процессов, используется отношение *приоритет блокировки**, связывающее между собой пары отношения *планируемая блокировка**.

Как правило, приоритет блокировки определяется тем, какой из процессов раньше попытался установить блокировку на рассматриваемый sc-элемент, хотя в общем случае приоритет может устанавливаться или меняться в зависимости от дополнительных критериев.

В случае попытки удаления некоторого sc-элемента некоторым процессом удаление может быть осуществлено только в случае, когда на данный sc-элемент не установлена (и не планируется) ни одна блокировка каким-либо другим процессом.

В других случаях необходимо обеспечить корректное завершение выполнения всех процессов, работающих с данным sc-элементом, и только потом удалить его физически.

Для реализации такой возможности каждому процессу в соответствие может быть поставлено множество удаляемых данным процессом sc-элементов.

Действие проверки наличия блокировок или планируемых блокировок на удаляемый sc-элемент и собственно его удаление или добавление во множество удаляемых sc-элементов для соответствующего процесса должно быть реализовано как транзакция.

удаляемые sc-элементы*
⇒ *первый домен**:
процесс в sc-памяти

sc-элементы, попавшие во множество удаляемых sc-элементов некоторого процесса в sc-памяти, доступны процессам, уже установившим (или планирующим установить) на эти sc-элементы блокировки ранее (до попытки его удаления), а для всех остальных процессов эти sc-элементы уже считаются удаленными. Процесс, пытающийся удалить sc-элемент, приостанавливает свое выполнение до того момента, пока все заблокировавшие и планирующие заблокировать данный sc-элемент процессы не разблокируют его. В общем случае один sc-элемент может входить во множества удаляемых элементов одновременно для нескольких процессов, в этом случае все такие процессы одновременно продолжат выполнение после снятия с этого sc-элемента всех блокировок. Если удаление пытается осуществить один из процессов, уже установивший на указанный sc-элемент блокировку, то алгоритм действий остается прежним — sc-элемент добавляется во множество удаляемых данным процессом sc-элементов, и будет физически удален, как только все остальные процессы, установившие на данный sc-элемент блокировки, снимут их.

Рассмотрим алгоритм снятия блокировки с некоторого sc-элемента:
- если на данный sc-элемент установлена одна или несколько *планируемых блокировок**, то первая из них по приоритету (или единственная) становится *блокировкой**, соответствующий ей процесс продолжает выполнение (становится настоящей сущностью); связка отношения приоритет выполнения, соответствовавшая удаленной связке отношения *планируемая блокировка** также удаляется, то есть приоритет смещается на одну позицию;
- если *планируемых блокировок**, установленных на данный sc-элемент, нет, но он попадает во множество удаляемых sc-элементов для одного или нескольких процессов, то рассматриваемый sc-элемент физически удаляется, а приостановленные до его удаления процессы продолжают свое выполнение (становится настоящими сущностями);
- если на данный sc-элемент не установлены планируемые блокировки и он не входит во множество удаляемых для какого-либо процесса, то блокировка просто снимается без каких-либо дополнительных изменений.

транзакция в sc-памяти
⇒ *разбиение**:
{
- *поиск некоторой конструкции в sc-памяти и автоматическая установка блокировки на любое изменение на найденные sc-элементы*
- *генерация некоторого sc-элемента и автоматическая установка на него полной блокировки*
- *проверка наличия на некотором sc-элементе блокировки и в зависимости от результата проверки установка блокировки или планируемой блокировки*
- *проверка наличия блокировок или планируемых блокировок на удаляемый sc-элемент и собственно его удаление или добавление во множество удаляемых sc-элементов для соответствующего процесса*
- *снятие блокировки с заданного sc-элемента и при необходимости установка первой по приоритету планируемой блокировки или удаление данного sc-элемента, если он входит во множество удаляемых sc-элементов для некоторого процесса*
- *поиск подпроцессов процесса и добавление их во множество отложенных действий в случае добавления самого процесса в данное множество*
- *поиск подпроцессов процесса и удаление их из множества отложенных действий в случае удаления самого процесса из данного множества*

}

При реализации *абстрактных программных sc-агентов* на *Языке SCP*, соблюдение всех принципов синхронизации соответствующих этим sc-агентам процессов обеспечивается на уровне *sc-агентов интерпретации scp-программ*, то есть средствами *ostis-платформы*. При реализации *абстрактных программных sc-агентов* на уровне *ostis-платформы*, соблюдение всех принципов синхронизации возлагается, во-первых, непосредственно на разработчика агентов, во-вторых, — на разработчика *ostis-платформы*. Так, например, платформа может предоставлять доступ к хранимым в sc-памяти элементам через некоторый *программный интерфейс*, уже учитывающий принципы работы с блокировками, что избавит разработчика агентов от необходимости учитывать все эти принципы вручную.

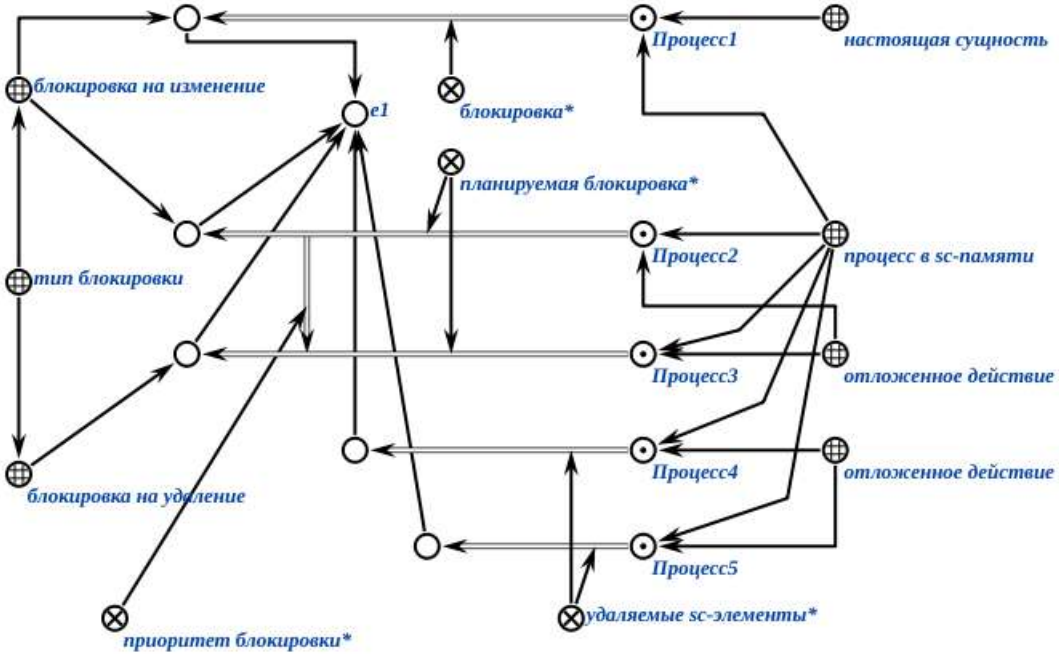
Кроме того, выделяется ряд специфичных принципов работы *абстрактных программных sc-агентов*, реализованных на *Языке SCP*:

- в результате появления в sc-памяти некоторой конструкции, удовлетворяющей условию инициирования какого-либо *абстрактного sc-агента*, реализованного при помощи *Языка SCP*, в *sc-памяти* генерируется и инициируется *scp-процесс*. В качестве шаблона для генерации используется *агентная scp-программа*, соответствующая данному *абстрактному sc-агенту*.
- каждый такой *scp-процесс*, соответствующий некоторой *агентной scp-программе*, может быть связан с набором структур, описывающих блокировки различных типов. Таким образом, синхронизация взаимодействия параллельно выполняемых *scp-процессов* осуществляется так же, как и в случае любых других *действий в sc-памяти*.
- несмотря на то что каждый *scp-оператор* представляет собой атомарное действие в sc-памяти, являющееся поддействием в рамках всего *scp-процесса*, блокировки, соответствующие одному оператору, не вводятся, чтобы избежать громоздкости и избытка дополнительных системных конструкций, создаваемых при выполнении некоторого *scp-процесса*. Вместо этого используются блокировки, общие для всего *scp-процесса*. Таким образом, *агенты интерпретации scp-программ* работают только с учетом блокировок, общих для всего интерпретируемого *scp-процесса*.
- процессы, описывающие деятельность агентов интерпретации *scp-программ*, как правило, не создаются, следовательно, и не вводятся соответствующие им блокировки. Поскольку такие агенты работают с уникальным scp-процессом и их число ограничено и известно, то использование блокировок для их синхронизации не требуется.
- в случае приостановки *scp-процесса* (добавления его во множество *отложенных действий*) в соответствии с общими правилами синхронизации все его дочерние процессы также должны быть приостановлены. В связи с этим все *scp-операторы*, которые в этот момент являются *настоящими сущностями*, становятся *отложенными действиями*.
- во избежание нежелательных изменений в самом теле *scp-процесса*, вся конструкция, сгенерированная на основе некоторой *scp-программы* (весь *sc-текст*, описывающий декомпозицию *scp-процесса* на *scp-операторы*), должна быть добавлена в *полную блокировку*, соответствующую данному *scp-процессу*.
- при необходимости разблокировать или заблокировать некоторую конструкцию каким-либо типом блокировки используются соответствующие *scp-операторы* класса *scp-оператор управления блокировками*.
- после завершения выполнения некоторого scp-процесса его текст, как правило, удаляется из *sc-памяти*, а все заблокированные конструкции освобождаются (разрушаются знаки структур, обозначавших блокировки).
- как правило, частный *класс действий*, соответствующий конкретной *scp-программе*, явно не вводится, а используется более общий класс *scp-процесс*, за исключением тех случаев, когда введение специального *класса действий* необходимо по каким-либо другим соображениям.

В общем случае весь механизм блокировок может описываться как на уровне *SC-кода* (для повышения уровня *платформенной независимости*), так и при необходимости может быть реализован на уровне *ostis-платформы*, например для повышения производительности. Для этого каждому выполняемому в sc-памяти процессу на нижнем уровне может быть поставлена в соответствие некая уникальная таблица, в каждый момент времени содержащая перечень заблокированных элементов с указанием типа блокировки.

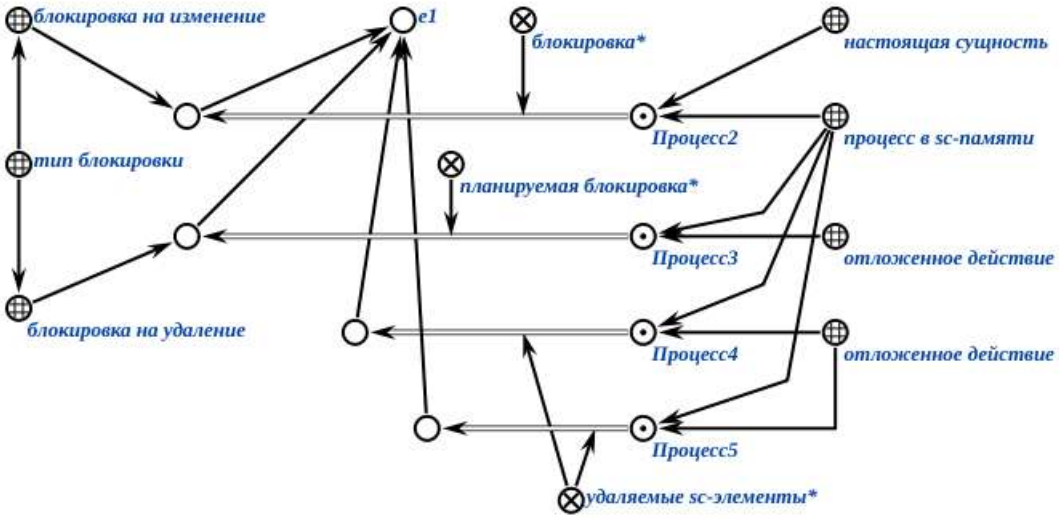
Рассмотрим пример применения описанного механизма.

SCg-текст. Пример использования планируемых блокировок
=



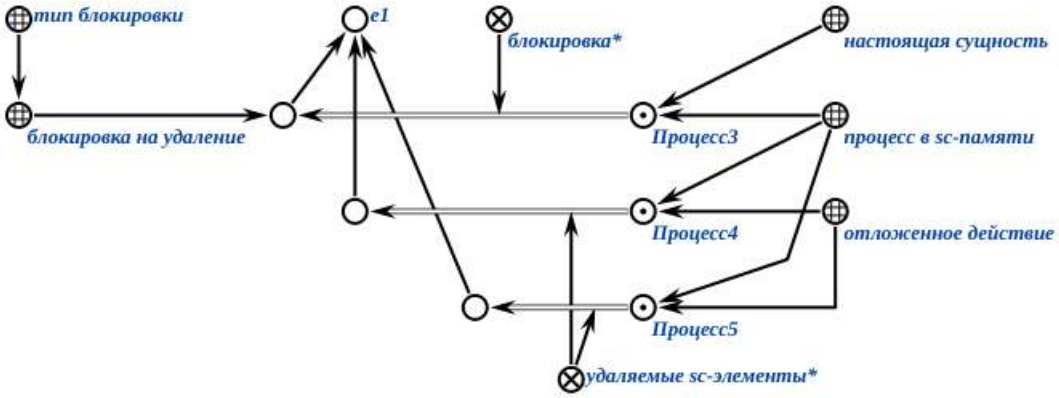
В данном примере *Процесс1* непосредственно работает с sc-элементом ***e1***,*Процесс2* и *Процесс3* планируют установить блокировку на любое изменение и блокировку на удаление соответственно, причем *Процесс2* попытался установить свою блокировку раньше, чем *Процесс3*, поэтому согласно направлению связки отношения *приоритет блокировки**, его блокировка будет установлена раньше. *Процесс4* и *Процесс5* ожидают снятия всех блокировок и планируемых блокировок, после чего ***e1*** будет удален и *Процесс1* и *Процесс2* продолжат свое выполнение. Никакие другие планируемые блокировки установлены быть уже не могут, поскольку ***e1*** попал во множество удаляемых sc-элементов как минимум одного процесса и, в соответствии с изложеннымивыше правилами, все остальные процессы кроме *Процесс1*-*Процесс5*, уже несмогут получить доступ к этому sc-элементу. Выполняемый процесс принадлежит множеству настоящая сущность, приостановленные – множеству отложенное действие.

SCg-текст. Пример использования планируемых блокировок (продолжение)
=



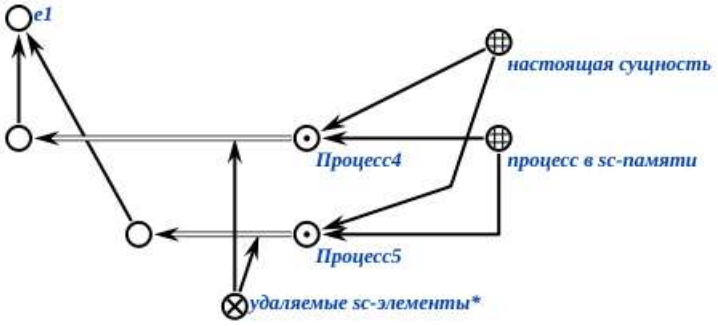
После того как *Процесс1* разблокировал sc-элемент ***e1***, этот элемент будет заблокирован *Процессом2*, и *Процесс2* продолжит выполнение. *Планируемая блокировка**, установленная *Процессом2*, становится обычной *блокировкой**.

SCg-текст. Пример использования блокировки на удаление
=



После того как *Процесс2* разблокировал sc-элемент ***e1***, этот элемент будет заблокирован *Процессом3*, и *Процесс3* продолжит выполнение.

SCg-текст. Удаляемые sc-элементы
=



Когда все процессы снимут блокировки с sc-элемента ***e1***, он может быть физически удален и *Процесс4* и *Процесс5* продолжат выполнение.

В зависимости от конкретных *типов блокировок* установленных паралельно выполняемыми процессами на некоторые sc-элементы и того, какие конкретно действия с этими *sc-элементами* предполагается выполнить далее в рамках выполнения этих процессов, возможны ситуации взаимоблокировки, когда каждый из указанных процессов будет ожидать снятия блокировки вторым процессом с нужного *sc-элемента*, не снимая при этом установленной им самим блокировки с *sc-элемента*, доступ к которому необходим второму процессу.

В случае когда хотя бы одна из блокировок является *полной блокировкой*, ситуация взаимоблокировки возникнуть не может, поскольку *sc-элементы*, попавшие в *полную блокировку* некоторого *scp-процесса*, не доступны другим *scp-процессам* даже для чтения и, таким образом, остальные *scp-процессы* будут работать так, как будто заблокированные *sc-элементы* просто отсутствуют в текущем состоянии *sc-памяти*.

В случаях, когда ни одна из установленных блокировок не является *полной блокировкой*, возможно появление взаимоблокировок.

Устранение *взаимоблокировки* невозможно без вмешательства специализированного *sc-метаагента*, который имеет право игнорировать блокировки, установленные другими процессами.

В общем случае проблема конкретной взаимоблокировки может быть решена путем выполнения специализированным *sc-метаагентом* следующих шагов:

- откат нескольких операций, выполненных одним из участвующих в взаимоблокировке процессов настолько шагов назад, насколько это необходимо для того, чтобы второй процесс получил доступ к необходимым *sc-элементам* и смог продолжить выполнение;
- ожидание выполнения второго процесса вплоть до завершения или до снятия им всех блокировок с *sc-элементов*, доступ к которым необходимо получить первому процессу;

- повторное выполнение в рамках первого процесса отмененных операций и продолжение его выполнения, но уже с учетом изменений в памяти, внесенных вторым процессом.

Для *sc-метаагентов* все sc-элементы, в том числе описывающие блокировки, планируемые блокировки и так далее полностью эквивалентны между собой с точки зрения доступа к ним, то есть любой *sc-метаагент* имеет доступ к любым sc-элементам, даже попавшим в полную блокировку для какого-либо другого процесса. Это необходимо для того, чтобы *sc-метаагенты* смогли выявлять и устранять различные проблемы, например, описанную выше проблему взаимоблокировки.

Таким образом, проблема синхронизации деятельности *sc-метаагентов* требует введения дополнительных правил.

Указанную проблему разделим на две более частные:
- обеспечение синхронизации деятельности *sc-метаагентов* между собой;
- обеспечение синхронизации деятельности *sc-метаагентов* и *программных sc-агентов*.

Первую проблему предлагается решить за счет запрета параллельного выполнения *sc-метаагентов*. Таким образом, в каждый момент времени в рамках одной *ostis-системы* может существовать только один процесс, соответствующий *sc-метаагенту* и являющийся *настоящей сущностью*.

Вторую проблему предлагается решить за счет введения дополнительных привилегий для *sc-метаагентов* при обращении к какому-либо sc-элементу. Для этого достаточно одного правила:

Если некоторый sc-элемент стал использоваться в рамках процесса, соответствующего *sc-метаагенту* (например, стал элементом хотя бы одного scp-оператора, входящего в данный процесс), то все процессы, в блокировки соответствующие которым попадает указанный sc-элемент, становятся отложенными действиями (приостанавливают выполнение). Как только указанный sc-элемент перестает использоваться в рамках процесса, соответствующего *sc-метаагенту*, все приостановленные по этой причине процессы продолжают выполнение.

Рассмотренные ограничения не ухудшают производительность *ostis-системы* существенно, поскольку *sc-метаагенты* предназначены для решения достаточно узкого класса задач, которые, как показал опыт практической разработки прототипов различных *ostis-систем*, возникают достаточно редко.

Стоит отметить, что возможна ситуация, при которой выполнение некоторого процесса в sc-памяти прервано по причине возникновения какой-либо ошибки. В таком случае существует вероятность того, что блокировка, установленная данным процессом не будет снята до тех пор, пока этого не сделает *sc-метаагент*, обнаруживший подобную ситуацию. Однако указанная проблема на уровне sc-модели может быть решена лишь частично, для случаев, когда ошибка возникает при интерпретации *scp-программы*, отслеживается *scp-интерпретатором* и в памяти формируется соответствующая конструкция, сообщающая о проблеме *sc-метаагенту*. Случаи, когда возникла ошибка на уровне *scp-интерпретатора* или *sc-памяти*, должны рассматриваться на уровне *ostis-платформы*.

§ 3.3.5. Базовый язык программирования ostis-систем

⇒ *подраздел**:
- *Пункт 3.3.5.1. Денотационная семантика Базового языка программирования ostis-систем*
- *Пункт 3.3.5.2. Операционная семантика Базового языка программирования ostis-систем*

⇒ *ключевой знак**:
- *Язык SCP*
- *Абстрактная scp-машина*

⇒ *ключевое понятие**:
- *scp-оператор*
- *параметр scp-программы'*

Как было показано в *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*, выделение Базового языка программирования для ostis-систем позволяет обеспечить четкое разделение уровня методов и соответственно, навыков ostis-системы, которые могут быть полностью описаны на уровне базы знаний и более низкоуровневых навыков, обеспечивающих интерпретацию указанных навыков более высокого уровня. Другими словами, выделение такого языка позволяет обеспечить платформенную независимость ostis-систем, как в случае программной реализации *ostis-платформы*, так и в случае *ассоциативного семантического компьютера*.

В качестве базового языка для описания программ обработки текстов *SC-кода* предлагается *Язык SCP*. ***Язык SCP*** – это графовый язык процедурного программирования, предназначенный для эффективной обработки *sc-текстов*. *Язык SCP* является языком параллельного асинхронного программирования.

Язык SCP
:= *часто используемый sc-идентификатор**:
[*scp-программа*]

Языком представления данных для текстов *Языка SCP* (*scp-программ*) является *SC-код* и, соответственно, любые варианты его внешнего представления. *Язык SCP* сам построен на основе *SC-кода*, вследствие чего *scp-программы* сами по себе могут входить в состав обрабатываемых данных для *scp-программ*, в том числе по отношению к самим себе. Таким образом, *Язык SCP* предоставляет возможность построения реконфигурируемых программ. Однако для обеспечения возможности реконфигурирования программы непосредственно в процессе ее интерпретации необходимо на уровне интерпретатора *Языка SCP (Абстрактной scp-машины)* обеспечить уникальность каждой исполняемой копии исходной *программы*. Такую исполняемую копию, сгенерированную на основе *scp-программы*, будем называть *scp-процессом*. Включение знака некоторого *действия в sc-памяти* во множество *scp-процессов* гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (*scp-операторов*), которые может интерпретировать реализация *Абстрактной scp-машины* (интерпретатора scp-программ).

Язык SCP рассматривается как ассемблер для *ассоциативного семантического компьютера*.

Абстрактная scp-машина
∈ *scp-машина*
⇐ *обобщенная модель**:
scp-интерпретатор

Базовая модель обработки sc-текстов включает в себя *Предметную область Базового языка программирования ostis-систем*, то есть описание *Синтаксиса* и *Денотационной семантики Языка SCP*, а также описание *Абстрактной scp-машины*, которая является моделью *scp-интерпретатора*, который должен являться частью *ostis-платформы* (хотя в общем случае могут существовать варианты платформы, не содержащие такого интерпретатора, что, однако, не позволит использовать достоинства предлагаемой базовой модели.

Рассмотрим ключевые особенности и достоинства *Базовой модели обработки sc-текстов*:
- Тексты программ *Языка SCP* записываются при помощи тех же унифицированных семантических сетей, что и обрабатываемая информация, таким образом, можно сказать, что *Синтаксис Языка SCP* на базовом уровне совпадает с *Синтаксисом SC-кода*.
- Подход к интерпретации *scp-программ* предполагает создание при каждом вызове *scp-программы* уникального *scp-процесса*.
- Одновременно в общей памяти могут выполняться несколько независимых *sc-агентов*, при этом разные копии *sc-агентов* могут выполняться на разных серверах, за счет распределенной реализации *ostis-платформы*. Более того, *Язык SCP* позволяет осуществлять параллельные асинхронные вызовы подпрограмм с последующей синхронизацией, и даже параллельно выполнять операторы в рамках одной *scp-программы*.
- Перенос *sc-агента* из одной системы в другую заключается в простом переносе фрагмента *базы знаний*, без каких-либо дополнительных операций, зависящих от *ostis-платформы*.
- Тот факт, что спецификации *sc-агентов* и их программы могут быть записаны на том же языке, что и обрабатываемые знания, существенно сокращает перечень специализированных средств, предназначенных для проектирования машин обработки знаний, и упрощает их разработку за счет использования более универсальных компонентов.
- Тот факт, что для интерпретации *scp-программы* создается соответствующий ей уникальный *scp-процесс*, позволяет по возможности оптимизировать план выполнения перед его реализацией и даже непосредственно в процессе выполнения без потенциальной опасности испортить общий универсальный алгоритм всей программы. Более того, такой подход к проектированию и интерпретации программ позволяет говорить о возможности создания самореконфигурируемых программ.

scp-программа
⊂ *программа в sc-памяти*
⊃ *агентная scp-программа*

Каждая ***scp-программа*** представляет собой обобщенную *sc-структуру*, описывающую один из вариантов декомпозиции действий некоторого класса, выполняемых в sc-памяти. Знак *sc-переменной*, соответствующей конкретному декомпозируемому действию является в рамках ***scp-программы*** *ключевым sc-элементом'*. Также явно указывается принадлежность данного знака множеству *scp-процессов*.

Принадлежность некоторого действия множеству *scp-процессов* гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (*scp-операторов*), которые может интерпретировать реализация *Абстрактной scp-машины*.

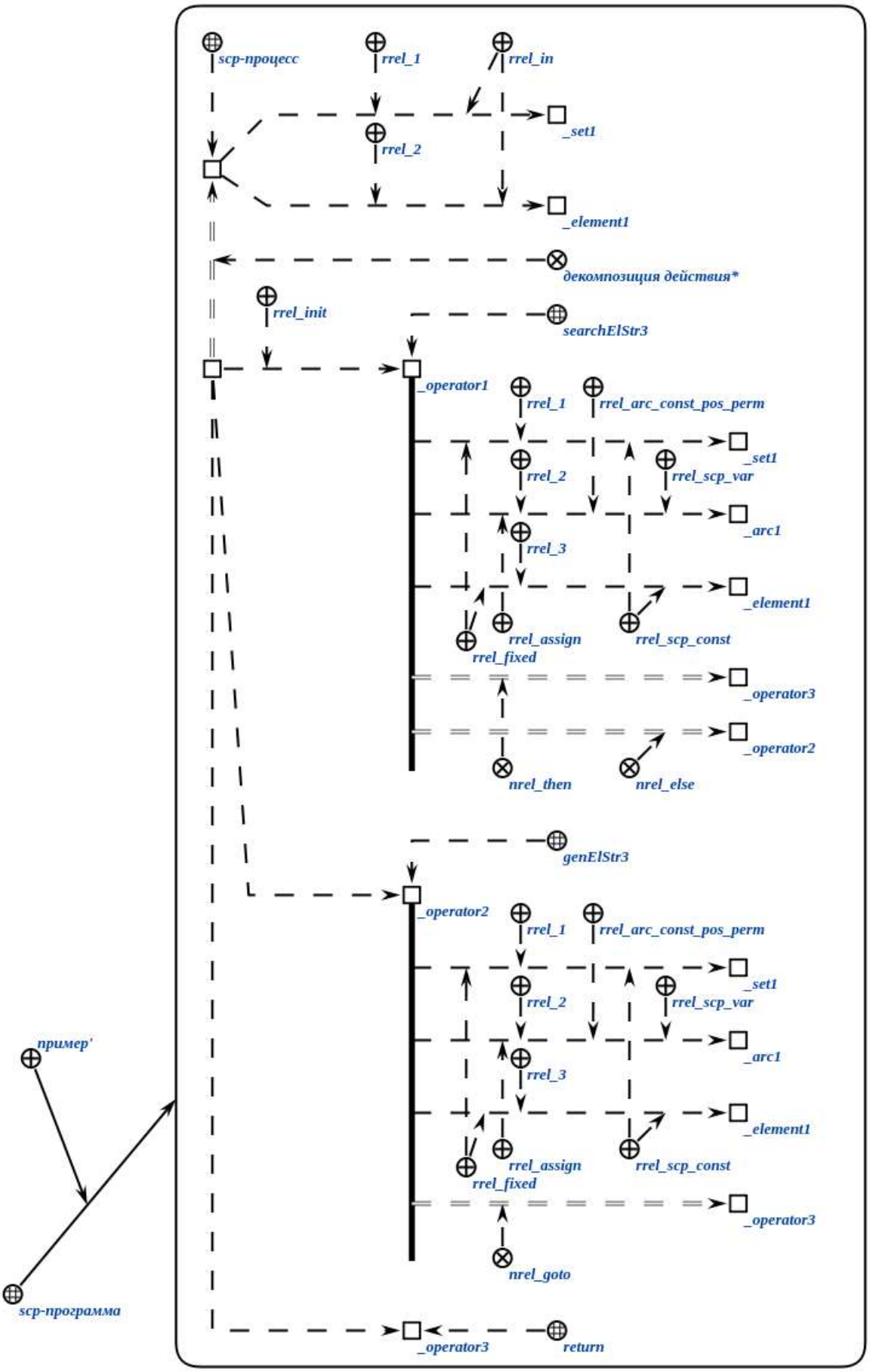
Таким образом, каждая ***scp-программа*** описывает в обобщенном виде декомпозицию некоторого *scp-процесса* на взаимосвязанные *scp-операторы*, с указанием, при их наличии, аргументов для данного *scp-процесса*.

По сути каждая ***scp-программа*** представляет собой описание последовательности элементарных операций, которые необходимо выполнить над семантической сетью, чтобы выполнить более сложное действие некоторого класса.

На рисунке *SCg-текст. Пример scp-программы* приведен пример простой *scp-программы*. В приведенном примере показана *scp-программа*, состоящая из трех *scp-операторов*. Данная программа проверяет, содержится ли в заданном множестве (первый параметр) заданный элемент (второй параметр), и, если нет, то добавляет его в это множество.

SCg-текст. Пример scp-программы

=



агентные scp-программы представляют собой частный случай *scp-программ* вообще, однако заслуживают отдельного рассмотрения, поскольку используются наиболее часто. *scp-программы* данного класса представляют собой реализации программ агентов обработки знаний, и имеют жестко фиксированный набор параметров. Каждая такая программа имеет ровно два *in-параметра'*. Значение первого параметра является знаком бинарной ориентированной пары, являющейся вторым компонентом связки отношения *первичное условие инициирования** для абстрактного *sc-агента*, в множество *программ sc-агента** которого входит рассматриваемая ***агентная scp-программа***, и, по сути, описывает класс событий, на которые реагирует указанный sc-агент.

Значением второго параметра является *sc-элемент*, с которым непосредственно связано событие, в результате возникновения которого был инициирован соответствующий *sc-агент*, то есть, например, сгенерированная либо удаляемая *sc-дуга* или *sc-ребро*.

Рассмотрим принципы реализации *абстрактных sc-агентов, реализуемых на Языке SCP*:

- общие принципы организации взаимодействия *sc-агентов* и пользователей *ostis-системы* через общую *sc-память*;
- в результате появления в sc-памяти некоторой конструкции, удовлетворяющей условию инициирования какого-либо *абстрактного sc-агента*, реализованного при помощи *Языка SCP*, в *sc-памяти* генерируется и инициируется *scp-процесс*. В качестве шаблона для генерации используется *агентная scp-программа*, указанная во множестве программ соответствующего *абстрактного sc-агента*;
- каждый такой *scp-процесс*, соответствующий некоторой *агентной scp-программе*, может быть связан с набором структур, описывающих блокировки различных типов. Таким образом, синхронизация взаимодействия параллельно выполняемых *scp-процессов* осуществляется так же, как и в случае любых других *действий в sc-памяти*;
- В рамках *scp-процесса* могут создаваться дочерние *scp-процессы*, однако синхронизация между ними при необходимости осуществляется посредством введения дополнительных внутренних блокировок. Таким образом, каждый *scp-процесс* с точки зрения *процессов в sc-памяти* является атомарным и законченным актом деятельности некоторого *sc-агента*;
- во избежание нежелательных изменений в самом теле *scp-процесса*, вся конструкция, сгенерированная на основе некоторой *scp-программы* (весь текст *scp-процесса*), должна быть добавлена в *полную блокировку*, соответствующую данному *scp-процессу*;
- все конструкции, сгенерированные в процессе выполнения *scp-процесса*, автоматически попадают в *полную блокировку*, соответствующую данному *scp-процессу*. Дополнительно следует отметить, что знак самой этой структуры и вся метаинформация о ней также включаются в эту структуру;
- при необходимости можно вручную разблокировать или заблокировать некоторую конструкцию каким-либо типом блокировки, используя соответствующие *scp-операторы* класса *scp-оператор управления блокировками*;
- после завершения выполнения некоторого *scp-процесса* его текст как правило, удаляется из *sc-памяти*, а все заблокированные конструкции освобождаются (разрушаются знаки структур, обозначавших блокировки);
- несмотря на то, что каждый *scp-оператор* представляет собой атомарное *действие в sc-памяти*, дополнительные блокировки, соответствующие одному оператору не вводятся, чтобы избежать громоздкости и избытка дополнительных системных конструкций, создаваемых при выполнении некоторого *scp-процесса*. Вместо этого используются блокировки, общие для всего *scp-процесса*. Таким образом, агенты *Абстрактной scp-машины* при интерпретации *scp-операторов* работают только с учетом блокировок, общих для всего интерпретируемого *scp-процесса*;
- как правило, частный *класс действий*, соответствующий конкретной *scp-программе* явно не вводится, а используется более общий класс *scp-процесс*, за исключением тех случаев, когда введение специального *класса действий* необходимо по каким-либо другим соображениям.

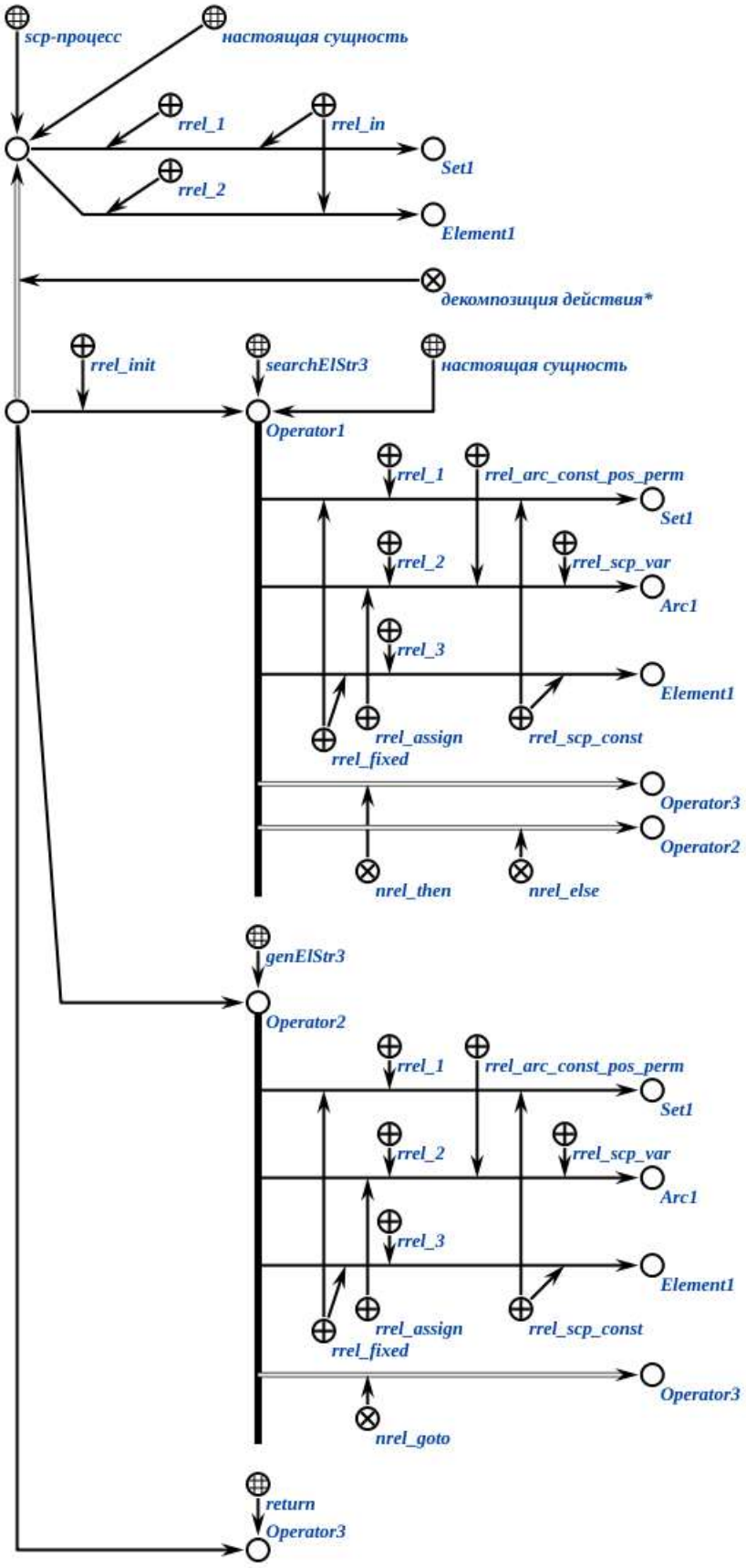
Под ***scp-процессом*** понимается некоторое *действие в sc-памяти*, однозначно описывающее конкретный акт выполнения некоторой *scp-программы* для заданных исходных данных. Если *scp-программа* описывает алгоритм решения какой-либо задачи в общем виде, то *scp-процесс* обозначает конкретное действие, реализующее данный алгоритм для заданных входных параметров.

По сути, ***scp-процесс*** представляет собой уникальную копию, созданную на основе *scp-программы*, в которой каждой *sc-переменной*, за исключением *scp-переменных'*, соответствует сгенерированная *sc-константа*.

Принадлежность некоторого действия множеству *scp-процессов* гарантирует тот факт, что в декомпозиции данного действия будут присутствовать только знаки элементарных действий (*scp-операторов*), которые может интерпретировать реализация *Абстрактной scp-машины*.

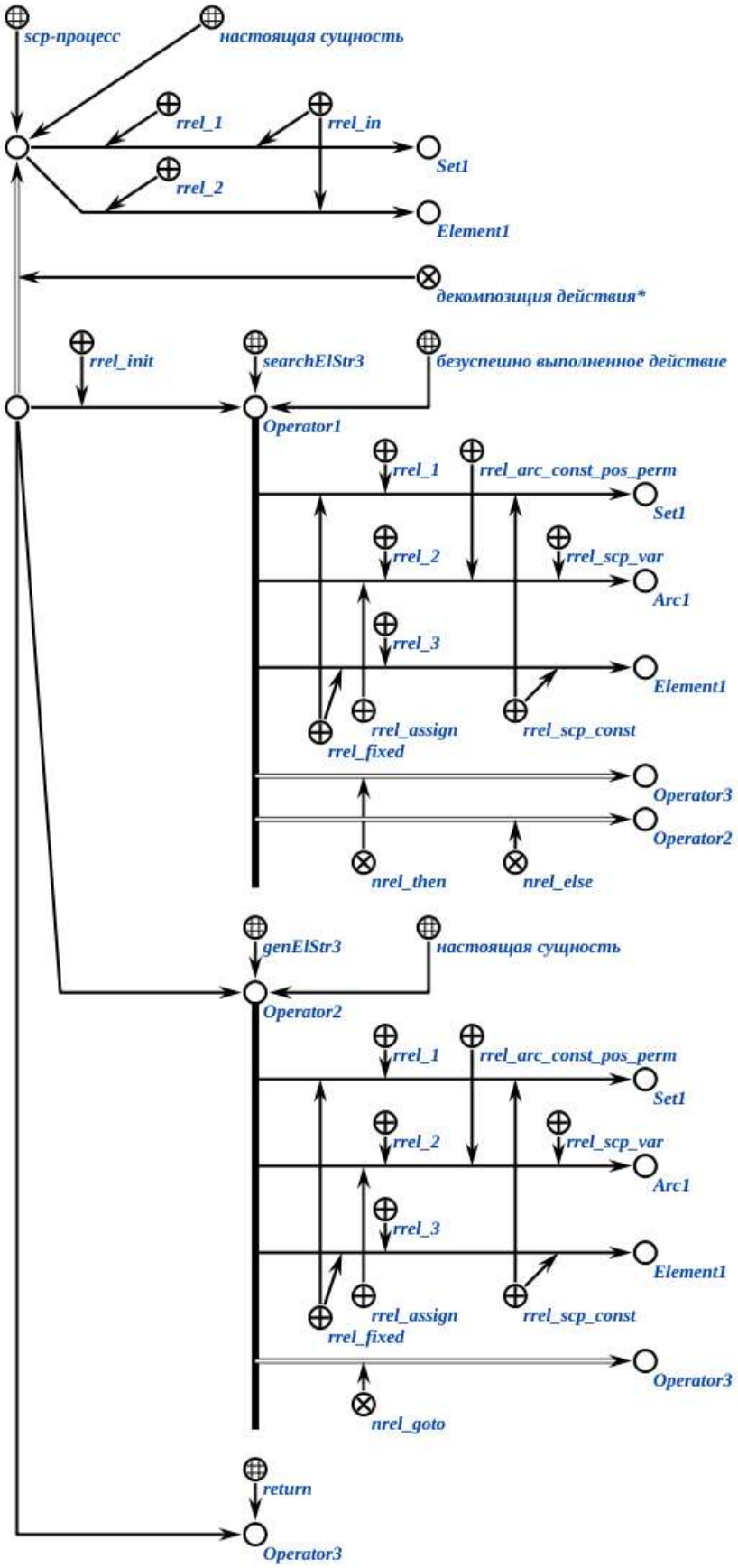
Рассмотрим пример поэтапного выполнения scp-процесса (*SCg-текст. Пример scp-процесса на начальной стадии выполнения – SCg-текст. Пример scp-процесса: выполнение завершено*), соответствующего ранее рассмотренному примеру scp-программы. В приведенном примере последовательно показаны состояния *scp-процесса*, соответствующего *scp-программе*, добавляющей заданный элемент в заданное множество, если он там ранее не содержался. В примере предполагается, что рассматриваемый элемент (*Element1*) изначально не содержится во множестве (*Set1*).

SCg-текст. Пример scp-процесса на начальной стадии выполнения
=



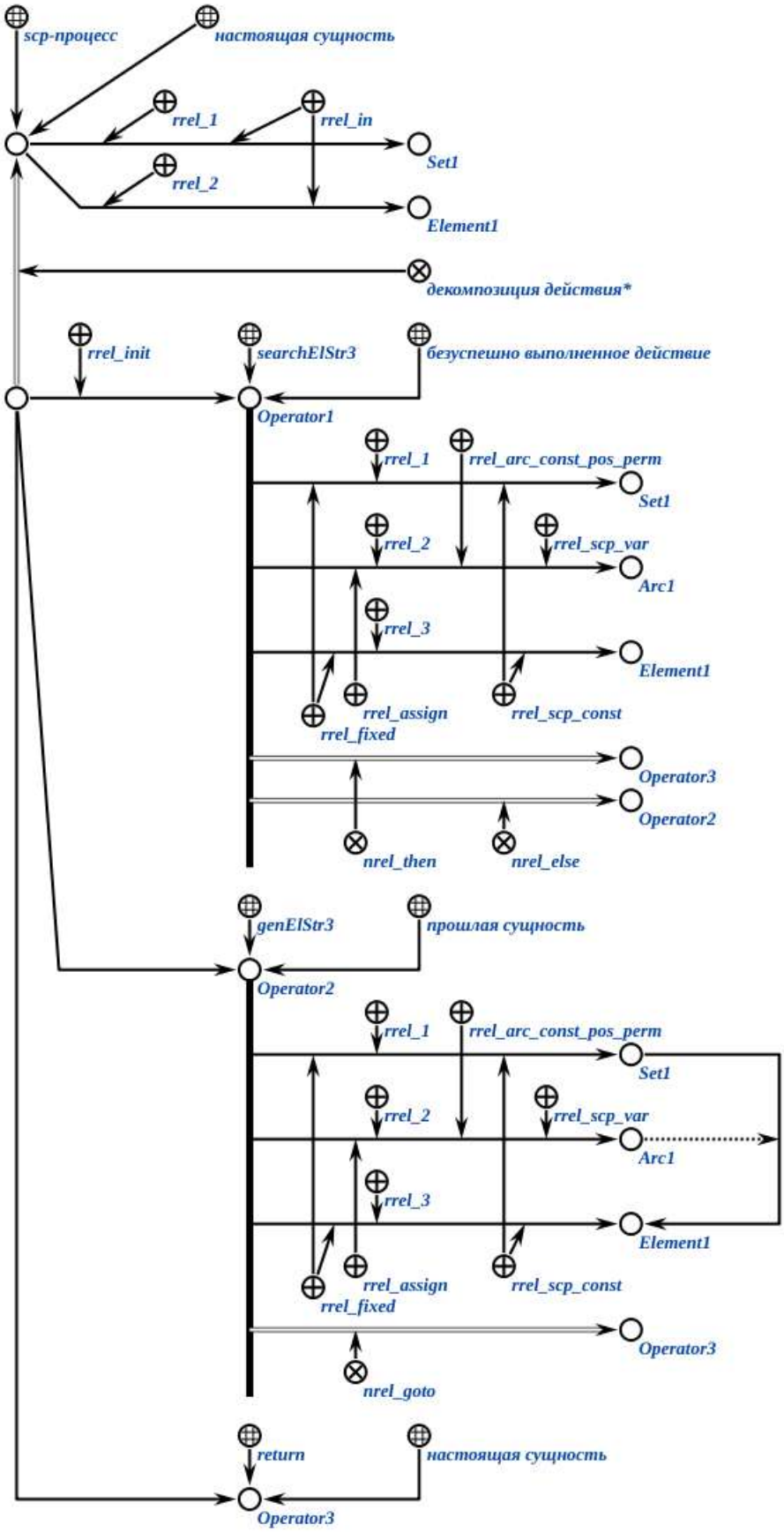
Осуществляется вызов *scp-программы*. Генерируется соответствующий *scp-процесс*. Происходит инициирование начального оператора scp-процесса *Operator1*.

SCg-текст. Пример scp-процесса: безуспешно выполнен оператор поиска
=



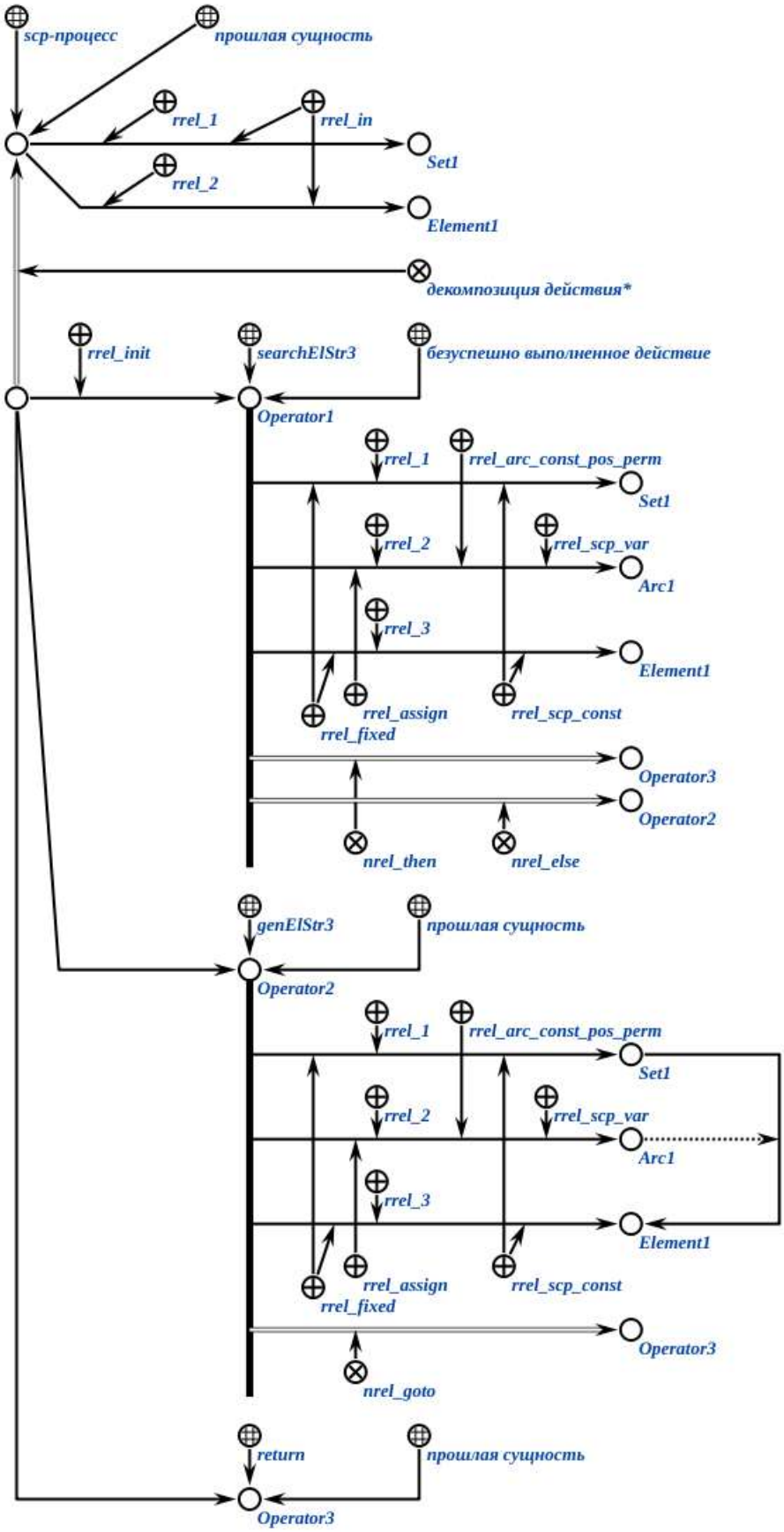
Оператор *Operator1* оказался безуспешно выполненным. Производится инициирование *scp-оператора генерации трехэлементной конструкции Operator2*.

SCg-текст. Пример scp-процесса: выполнен оператор генерации, элемент добавлен во множество =



Оператор *Operator2* выполнился. Производится инициирование *scp-оператора завершения выполнения программы Operator3*.

SCg-текст. Пример scp-процесса: выполнение завершено
=



Оператор *Operator3* выполнился. Выполнение *scp-процесса* завершается.

scp-оператор
⊂ *действие в sc-памяти*
⇐ *семейство подмножеств**:
атомарный тип scp-оператора

Каждый ***scp-оператор*** представляет собой некоторое элементарное *действие в sc-памяти*. Аргументы *scp-оператора* будем называть операндами. Порядок операндов указывается при помощи соответствующих ролевых отношений (*1′*, *2′*, *3′* и так далее). Операнд, помеченный ролевым отношением *1′*, будем называть первым операндом, помеченный ролевым отношением *2′* — вторым операндом, и т.д. Тип и смысл каждого операнда также уточняется при помощи различных подклассов отношения *scp-операнд′*. В общем случае операндом может быть любой *sc-элемент*, в том числе, знак какой-либо *scp-программы*, в том числе самой программы, содержащей данный оператор.

Каждый ***scp-оператор*** должен иметь один и более операнд, а также указание того ***scp-оператора*** (или нескольких), который должен быть выполнен следующим. Исключение их данного правила составляет *scp-оператор завершения выполнения программы*, который не содержит ни одного операнда и после выполнения которого никакие *scp-операторы* в рамках данной программы выполняться не могут.

Каждый ***атомарный тип scp-оператора*** представляет собой класс *scp-операторов*, который не разбивается на более частные, и, соответственно, интерпретируется реализацией *Абстрактной scp-машины*.

начальный оператор′
⊂ *1′*

Ролевое отношение ***начальный оператор′*** указывает в рамках декомпозиции соответствующего *scp-программе scp-процесса* те *scp-операторы*, которые должны быть выполнены в первую очередь, то есть те, с которых собственно начинается выполнение *scp-процесса*.

параметр scp-программы′
⊂ *аргумент действия′*
⇒ *разбиение**:
{• *in-параметр′*
• *out-параметр′*
}

Ролевое отношение ***параметр scp-программы′*** связывает знак соответствующего *scp-программе scp-процесса* с его аргументами.

Параметры типа ***in-параметр′*** хоть и соответствуют *переменным scp-программы′*, не могут менять значение в процессе ее интерпретации. Фиксированное значение переменной устанавливается при создании уникальной копии *scp-программы* (*scp-процесса*) для ее интерпретации, и, таким образом, соответствующая *scp-переменная′* на момент начала ее интерпретации становится *scp-константой′* в рамках каждого *scp-оператора*, в котором встречалась данная *scp-переменная′*. Использование *in-параметров* можно рассматривать по аналогии с использованием варианта механизма передачи по значению в традиционных языках программирования, с тем условием, что значение локальной переменной в рамках дочерней программы не может быть изменено.

Параметры типа ***out-параметр′*** соответствуют *переменным scp-программы′* и обладают всеми теми же соответствующими свойствами. Чаще всего предполагается, что значение данного параметра необходимо родительской *scp-программе*, содержащей оператор вызова текущей *scp-программы*. При этом на момент начала интерпретации в качестве параметра дочернему процессу передается непосредственно узел, обозначающий переменную (а точнее, ее уникальную копию в рамках процесса) родительского процесса. Указанная переменная может при необходимости иметь значение, либо не иметь. После завершения и во время интерпретации дочернего процесса родительский процесс по-прежнему может работать с переменной, переданной в качестве *out-параметра′*, при необходимости просматривая или изменяя ее значение. Использование out-параметра можно рассматривать по аналогии с использованием механизма передачи по ссылке в традиционных *языках программирования*.

Рассмотрим классификацию *sc-конструкций* с точки зрения *Базовой модели обработки sc-текстов*.

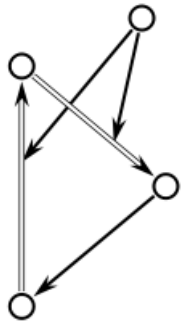
sc-конструкция
⇒ *разбиение**:
Классификация sc-конструкций с точки зрения Базовой модели обработки sc-текстов
= {• *sc-конструкция нестандартного вида*
• *sc-конструкция стандартного вида*
}
⇒ *разбиение**:

{
- *одноэлементная sc-конструкция*
- *трехэлементная sc-конструкция*
- *пятиэлементная sc-конструкция*

}

Каждая *sc-конструкция нестандартного вида* состоит из произвольного количества *sc-элементов* произвольного типа (*SCg-текст. Пример пятиэлементной sc-конструкции в SCg-коде*).

SCg-текст. Пример sc-конструкции нестандартного вида
=



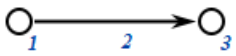
В свою очередь, каждый элемент *sc-конструкции стандартного вида* имеет свою условную строго фиксированную позицию в рамках этой sc-конструкции (первый элемент, второй элемент и так далее). В зависимости от указанной позиции вводятся дополнительные ограничения на тип соответствующего *sc-элемента*.

Каждая *одноэлементная sc-конструкция* состоит из одного *sc-элемента* произвольного типа (Рис. *SCg-текст. Пример одноэлементных sc-конструкций в SCg-коде*).

SCg-текст. Пример одноэлементных sc-конструкций в SCg-коде
=

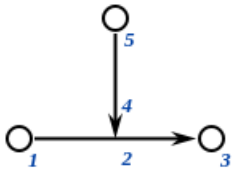
Каждая *трехэлементная sc-конструкция* состоит из трех *sc-элементов* (Рис. *SCg-текст. Пример трехэлементной sc-конструкции в SCg-коде*). Второй элемент всегда является *sc-коннектором*, остальные элементы могут быть произвольного типа.

SCg-текст. Пример трехэлементной sc-конструкции в SCg-коде
=



Каждая *пятиэлементная sc-конструкция* состоит из пяти *sc-элементов* (Рис. *SCg-текст. Пример пятиэлементной sc-конструкции в SCg-коде*). Второй и четвертый элементы обязательно являются *sc-коннекторами*, остальные элементы могут быть произвольного типа.

SCg-текст. Пример пятиэлементной sc-конструкции в SCg-коде
=



Пункт 3.3.5.1. Денотационная семантика Базового языка программирования ostis-систем

⇒ *ключевое понятие**:
- *scp-оператор*
- *scp-операнд'*

scp-оператор
⊂ *действие в sc-памяти*
⇐ *семейство подмножеств**:
атомарный тип scp-оператора
⇒ *разбиение**:
{• *scp-оператор генерации конструкций*
⇒ *разбиение**:
{• *scp-оператор генерации конструкции по произвольному образцу*
• *scp-оператор генерации пятиэлементной конструкции*
• *scp-оператор генерации трехэлементной конструкции*
• *scp-оператор генерации одноэлементной конструкции*
}
• *scp-оператор ассоциативного поиска конструкций*
⇒ *разбиение**:
{• *scp-оператор поиска конструкции по произвольному образцу*
• *scp-оператор поиска пятиэлементной конструкции с формированием множеств*
• *scp-оператор поиска трехэлементной конструкции с формированием множеств*
• *scp-оператор поиска пятиэлементной конструкции*
• *scp-оператор поиска трехэлементной конструкции*
}
• *scp-оператор удаления конструкций*
⇒ *разбиение**:
{• *scp-оператор удаления множества элементов трехэлементной конструкции*
• *scp-оператор удаления одноэлементной конструкции*
• *scp-оператор удаления пятиэлементной конструкции*
• *scp-оператор удаления трехэлементной конструкции*
}
• *scp-оператор проверки условий*
⇒ *разбиение**:
{• *scp-оператор сравнения числовых содержимых файлов*
• *scp-оператор проверки равенства числовых содержимых файлов*
• *scp-оператор проверки совпадения значений операндов*
• *scp-оператор проверки наличия содержимого у файла*
• *scp-оператор проверки наличия значения у переменной*
• *scp-оператор проверки типа sc-элемента*
}
• *scp-оператор управления значениями операндов*
⇒ *разбиение**:
{• *scp-оператор удаления значения переменной*
• *scp-оператор присваивания значения переменной*
}
• *scp-оператор управления scp-процессами*
⇒ *разбиение**:
{• *scp-оператор удаления значения переменной*

- *scp-оператор завершения выполнения программы*
- *конъюнкция предшествующих scp-операторов*
- *scp-оператор ожидания завершения выполнения множества scp-программ*
- *scp-оператор ожидания завершения выполнения scp-программы*
- *scp-оператор асинхронного вызова подпрограммы*

}

- *scp-оператор управления событиями*
  - ⊃ *scp-оператор ожидания события*
- *scp-оператор обработки содержимых файлов*
  - ⇒ *разбиение**:
    - {• *scp-оператор вычисления арксинуса числового содержимого файла*
      - *scp-оператор вычисления арккосинуса числового содержимого файла*
      - *scp-оператор деления числовых содержимых файлов*
      - *scp-оператор умножения числовых содержимых файлов*
      - *scp-оператор вычитания числовых содержимых файлов*
      - *scp-оператор сложения числовых содержимых файлов*
      - *scp-оператор вычисления тангенса числового содержимого файла*
      - *scp-оператор вычисления косинуса числового содержимого файла*
      - *scp-оператор вычисления синуса числового содержимого файла*
      - *scp-оператор вычисления логарифма числового содержимого файла*
      - *scp-оператор возведения числового содержимого файла в степень*
      - *scp-оператор удаления содержимого файла*
      - *scp-оператор копирования содержимого файла*
      - *scp-оператор нахождения остатка от деления числовых содержимых файлов*
      - *scp-оператор нахождения целой части от деления числовых содержимых файлов*
      - *scp-оператор вычисления арктангенса числового содержимого файла*
      - *scp-оператор перевода в верхний регистр строкового содержимого файла*
      - *scp-оператор перевода в верхний регистр строкового содержимого файла*
      - *scp-оператор замены определенной части строкового содержимого файла на содержимое указанной файла*
      - *scp-оператор проверки совпадения конца строкового содержимого файла со строковом содержимым другого файла*
      - *scp-оператор проверки совпадения начальной части строкового содержимого файла со строковом содержимым другого файла*
      - *scp-оператор получения части строкового содержимого файла по индексам*
      - *scp-оператор поиска строкового содержимого файла в строковом содержимом другого файла*
      - *scp-оператор вычисления длины строкового содержимого файла*
      - *scp-оператор разбиения строки на подстроки*
      - *scp-оператор лексикографического сравнения строковых содержимых файлов*
      - *scp-оператор проверки равенства строковых содержимых файлов*

    }
- *scp-оператор управления блокировками*
  - ⇒ *разбиение**:
    - {• *scp-оператор снятия всех блокировок данного scp-процесса*
      - *scp-оператор снятия блокировки с sc-элемента*
      - *scp-оператор установки полной блокировки на sc-элемент*
      - *scp-оператор установки блокировки на изменение sc-элемента*
      - *scp-оператор установки блокировки на удаление sc-элемента*
      - *scp-оператор снятия блокировки со структуры*
      - *scp-оператор установки полной блокировки на структуру*
      - *scp-оператор установки блокировки на изменение структуры*
      - *scp-оператор установки блокировки на удаление структуры*

    }

}

scp-операнд'

⊂ *аргумент действия'*

∈ *неосновное понятие*

∈ *ролевое отношение*

⇒ *разбиение**:

{• *scp-константа'*

- *scp-переменная'*

}

⇒ *разбиение**:

{• *scp-операнд с заданным значением'*
• *scp-операнд со свободным значением'*

}

⇒ *разбиение**:

{• *константный sc-элемент'*
• *переменный sc-элемент'*

}

⇒ *включение**:

- *формируемое множество'*
  - ⇒ *разбиение**:
    {• *формируемое множество 1'*
    • *формируемое множество 2'*
    • *формируемое множество 3'*
    • *формируемое множество 4'*
    • *формируемое множество 5'*
    }
- *удаляемый sc-элемент'*
- *тип sc-элемента'*
  - ⇒ *разбиение**:
    {• *sc-узел'*
    ⇒ *разбиение**:
    {• *структура'*
    • *отношение'*
    ⊃ *ролевое отношение'*
    • *класс'*
    }
    • *sc-дуга'*
    ⇒ *разбиение**:
    {• *sc-дуга общего вида'*
    • *sc-дуга принадлежности'*
    ⊃ *sc-дуга основного вида'*
    = *(константный sc-элемент'∩ позитивная sc-дуга принадлежности'∩ постоянная sc-дуга принадлежности')*
    ⇒ *разбиение**:
    {• *позитивная sc-дуга принадлежности'*
    • *негативная sc-дуга принадлежности'*
    • *нечеткая sc-дуга принадлежности'*
    }
    ⇒ *разбиение**:
    {• *временная sc-дуга принадлежности'*
    • *постоянная sc-дуга принадлежности'*
    }
    }
    • *sc-ребро'*
    • *файл'*
    }

Ролевое отношение *scp-операнд'* является неосновным понятием и указывает на принадлежность аргументов *scp-оператору*. Помимо указания какого-либо класса *scp-операндов'* порядок аргументов *scp-оператора* дополнительно уточняется *ролевыми отношениями 1'*, *2'* и так далее

В рамках *scp-программы* ***scp-константы'*** явно участвуют в *scp-операторах* в качестве элементов (в теоретико-множественном смысле) и напрямую обрабатываются при интерпретации *scp-программы*. Константами в рамках *scp-программы* могут быть *sc-элементы* любого типа, как *sc-константы*, так и *sc-переменные*. Константа в рамках *scp-программы* остается неизменной в течение всего срока интерпретации. Константа *scp-программы* может быть рассмотрена как переменная, значение которой совпадает с самой переменной в каждый момент времени, и изменено быть не может. Таким образом, далее будем считать, что *scp-константа'* и ее значение это одно и то же. Каждый *in-параметр'* при интерпретации каждой конкретной копии *scp-программы* становится *scp-константой'*

в рамках всех ее операторов, хотя в исходном теле данной программы в каждом из этих операторов он является *scp-переменной'*.

В рамках *scp-программы* ***scp-переменные'*** не обрабатываются явно при интерпретации, обрабатываются значения переменных. Каждая переменная *scp-программы* может иметь одно значение в каждый момент времени, то есть представляет собой ситуативный *синглетон*, элементом которого является текущее значение *scp-переменной'*. Значение каждой *scp-переменной'* может меняться в ходе интерпретации *scp-программы*. При этом интерпретатор при обработке *scp-оператора* работает непосредственно со значениями *scp-переменных'*, а не самими *scp-переменными'* (которые также являются узлами той же семантической сети).

Значение операндов, помеченных ролевым отношением ***scp-операнд с заданным значением'***, считается заданным в рамках текущего *scp-оператора*. Данное значение учитывается при выполнении *scp-оператора* и остается неизменным после окончания выполнения *scp-оператора*. Каждая *scp-константа'* по умолчанию рассматривается как *scp-операнд с заданным значением'*, в связи с чем явное использование данного ролевого отношения в таком случае является избыточным. В таком случае в качестве значения рассматривается непосредственно сам операнд. В случае если отношением *scp-операнд с заданным значением'* помечена *scp-переменная'*, то осуществляется попытка поиска значения для данной *scp-переменной'* (ее элемента). Если попытка оказалась безуспешной, то возникает ошибка времени выполнения, которая должна быть обработана соответствующим образом.

Любой *scp-операнд с заданным значением'* независимо от конкретного типа *scp-оператора* может быть *scp-переменной'*.

Значение операндов, помеченных ролевым отношением *scp-операнд со свободным значением'*, считается свободным (не заданным заранее) в рамках текущего *scp-оператора*. В начале выполнения *scp-оператора* связь между *scp-переменной'*, помеченной данным ролевым отношением, и ее элементом (значением) всегда удаляется. В результате выполнения данного оператора может быть либо сгенерировано новое значение *scp-переменной'*, либо не сгенерировано, тогда *scp-переменная'* будет считаться не имеющей значения. Ни одна *scp-константа'* не может быть помечена как *scp-операнд со свободным значением'*, поскольку константа не может изменять свое значение в ходе интерпретации *scp-программы*.

Ролевое отношение *тип sc-элемента'* используется для уточнения типа *sc-элемента*, выступающего в роли значения некоторого операнда. *тип sc-элемента'* имеет смысл указывать только для операндов, помеченных как *scp-операнд со свободным значением'*, тогда данное уточнение типа *sc-элемента* будет использовано для сужения области поиска либо уточнения параметров генерации каких-либо конструкций. Значением *scp-операндов с заданным значением'* является конкретный, известный на момент начала выполнения *scp-оператора sc-элемент* с конкретным типом, не зависящим от указания *типа sc-элемента'*, в связи с чем использование ролевого отношения *тип sc-элемента'* в данном случае является некорректным.

Допускается использование комбинаций семантически непротиворечащих друг другу подмножеств указанного отношения. Например, допускается комбинация *константный sc-элемент'* и *sc-дуга общего вида'*, но не допускается комбинация *sc-узел'* и *sc-дуга'*.

Ролевое отношение ***формируемое множество'*** используется для указания того факта, что в результате выполнения *scp-оператора* должно быть сформировано либо дополнено некоторое множество *sc-элементов*, являющееся значением одного из операндов данного *scp-оператора*. При этом если данный операнд помечен как *scp-операнд со свободным значением'*, то множество будет сформировано с нуля (сгенерирован новый *sc-элемент*, обозначающий данное множество), в противном случае уже существующее множество может быть дополнено. Использование данного ролевого отношения предполагает, что при его отсутствии множество бы не формировалось, а значением указанного операнда стал бы произвольный *sc-элемент* из данного множества.

Ролевое отношение *формируемое множество'* без уточнения порядкового номера используется только в *scp-операторах обработки произвольных конструкций*. Для явного указания номера операнда, которому соответствует *формируемое множество'*, используются подмножества данного ролевого отношения, аналогичные ролевым отношениям, задающим порядок элементов в кортеже (*1', 2', 3'* и так далее), например *формируемое множество 1'*, *формируемое множество 2'* и так далее. Указанные ролевые отношения используются только в *scp-операторах поиска конструкций с формированием множеств*.

Ролевое отношение ***удаляемый sc-элемент'*** используется для указания тех операндов, значение которых должно быть удалено в процессе выполнения *scp-операторов удаления*. Данным ролевым отношением может быть помечен как *scp-операнд с заданным значением'*, так и *scp-операнд со свободным значением'*. При этом удаляемым *sc-элементом* может быть как *scp-константа'*, так и *scp-переменная'* (в случае *scp-переменной'* удаляется не только связка принадлежности между этой *scp-переменной'* и ее значением, но и непосредственно сам *sc-элемент*, являющийся значением).

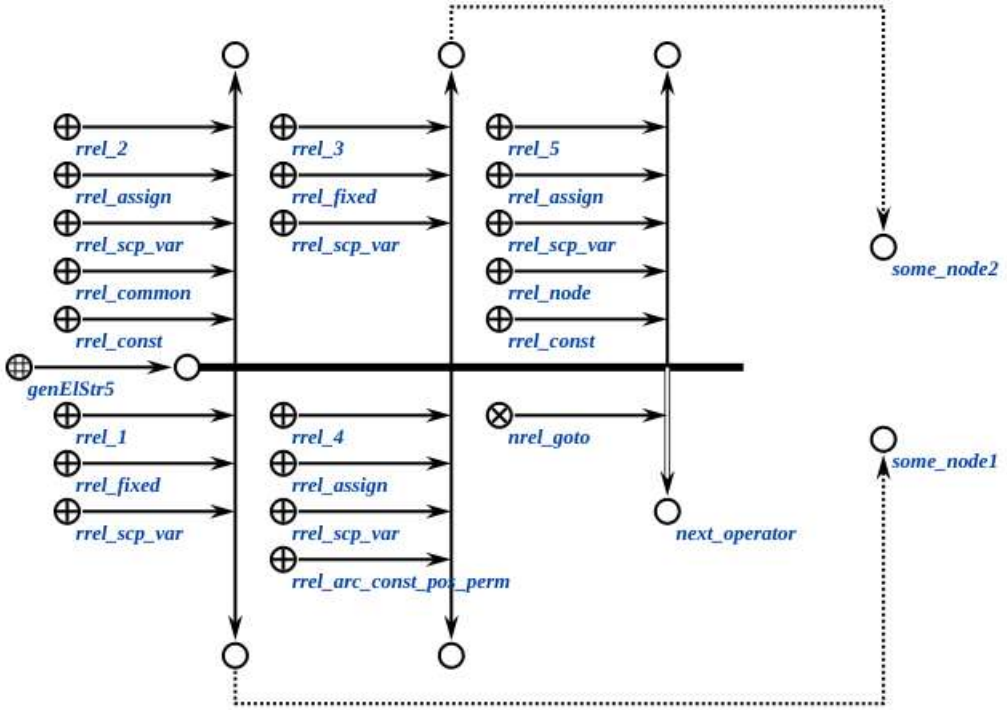
следует отличать*

∋ {• *scp-переменная'*
  • *sc-переменная*

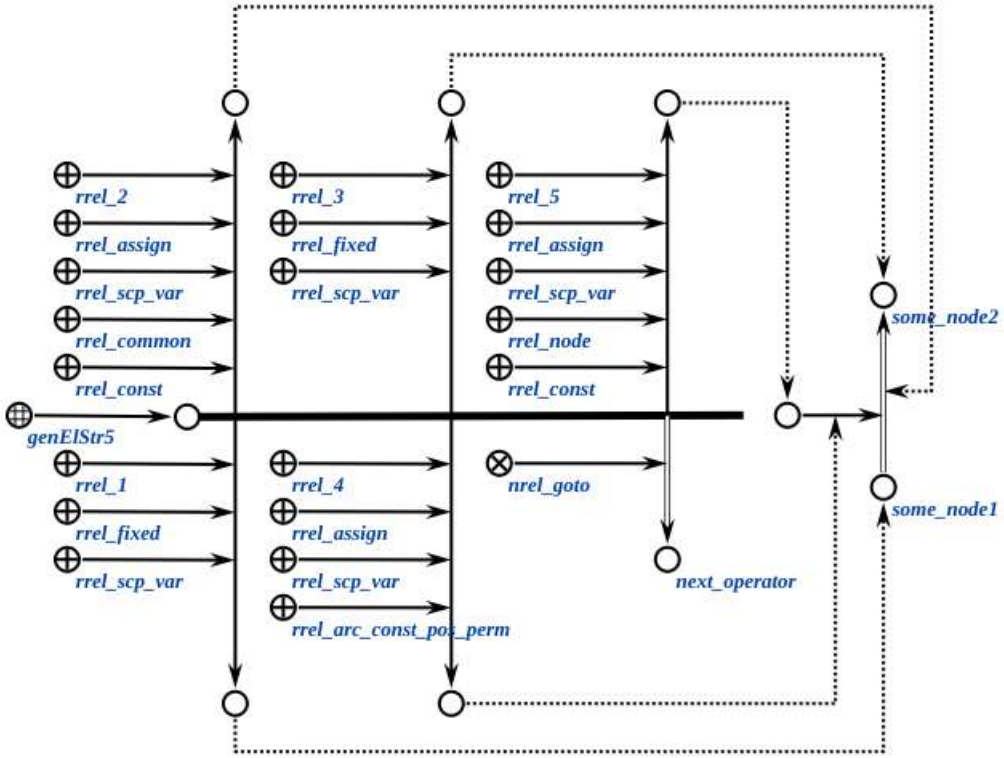
}
∋ {• *scp-константа'*
• *sc-константа*
}

На рисунках *SCg-текст. Пример выполнения scp-оператора генерации пятиэлементной конструкции (вызов scp-оператора)* – *SCg-текст. Пример выполнения scp-оператора генерации пятиэлементной конструкции (результат выполнения scp-оператора)* показан пример работы scp-оператора генерации пятиэлементной конструкции. В приведенном примере выполняется генерация пятиэлементной конструкции, которая имеет два scp-операнда с заданным значением. В примере предполагается, что рассматриваемые элементы (some_node1 и some_node2) изначально никак не связаны между собой.

SCg-текст. Пример выполнения scp-оператора генерации пятиэлементной конструкции (вызов scp-оператора)
=

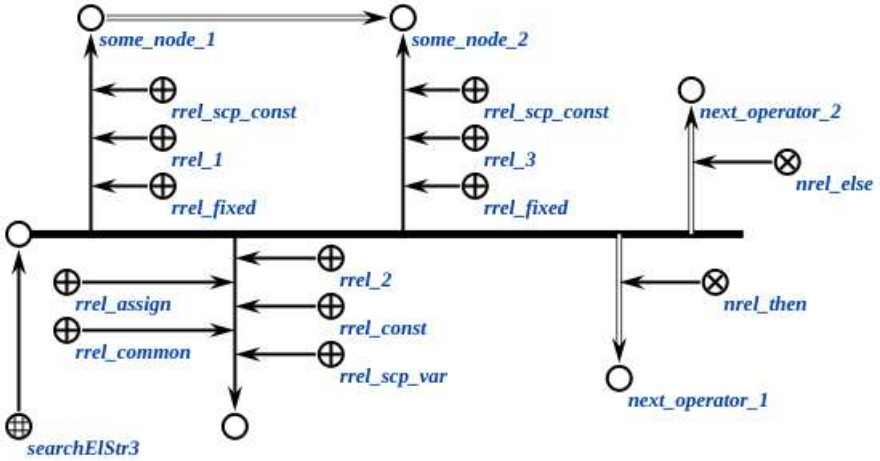


SCg-текст. Пример выполнения scp-оператора генерации пятиэлементной конструкции (результат выполнения scp-оператора)
=



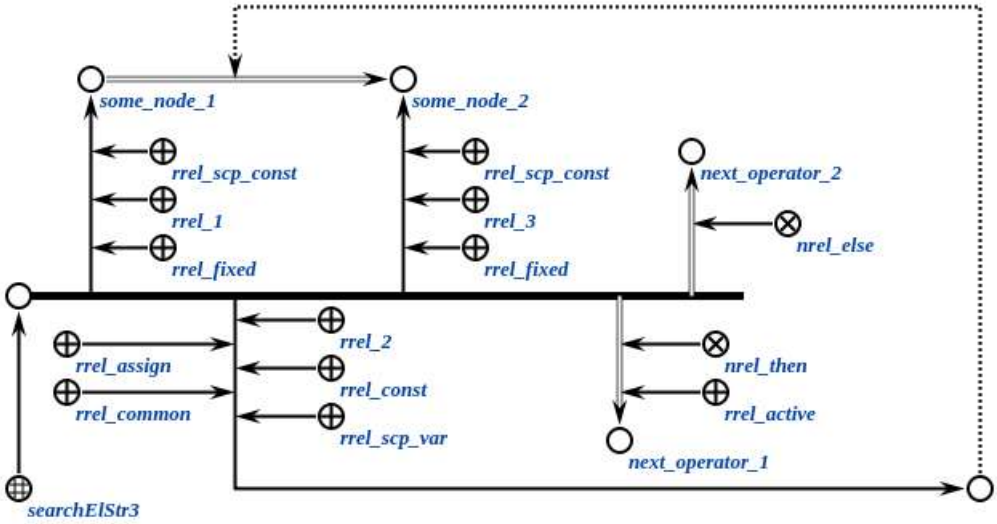
На рисунках *SCg-текст. Пример выполнения scp-оператора поиска трехэлементной конструкции (вызов scp-оператора)* – *SCg-текст. Пример выполнения scp-оператора поиска трехэлементной конструкции (результат выполнения scp-оператора)* приведен пример scp-оператора поиска трехэлементной конструкции, которая имеет два scp-операнда с заданным значением. В примере предполагается, что рассматриваемые элементы (some_node1 и some_node2) изначально связаны между собой константной постоянной sc-дугой.

SCg-текст. Пример выполнения scp-оператора поиска трехэлементной конструкции (вызов scp-оператора)
=



SCg-текст. Пример выполнения scp-оператора поиска трехэлементной конструкции (результат выполнения scp-оператора)

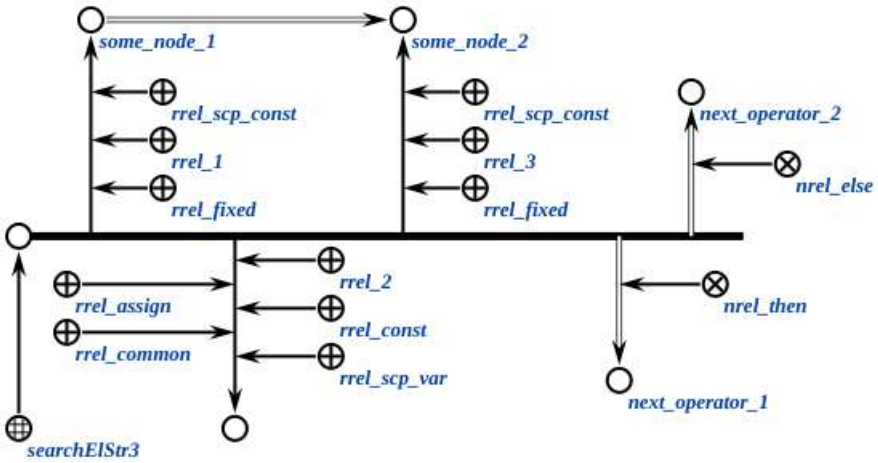
=



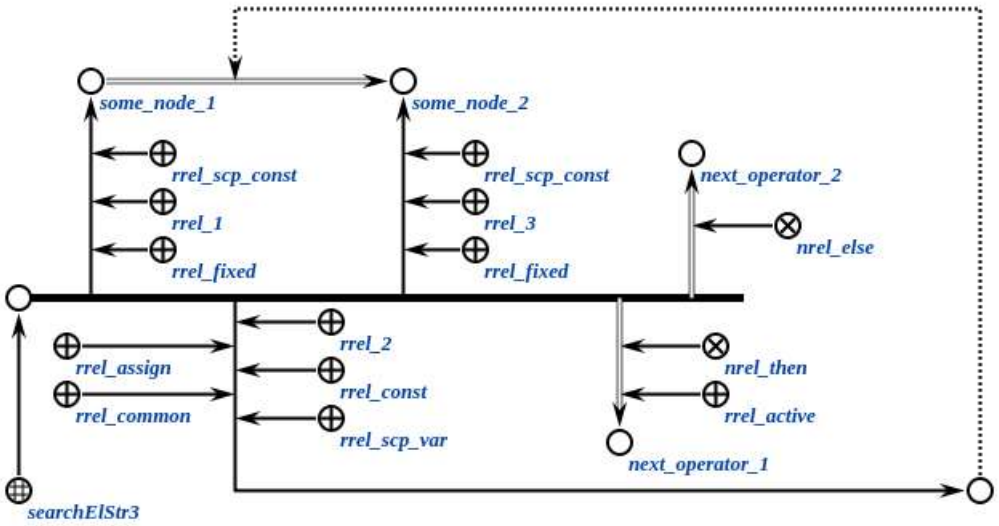
На рисунках *SCg-текст. Пример выполнения scp-оператора удаления одноэлементной конструкции (вызов scp-оператора) – SCg-текст. Пример выполнения scp-оператора удаления одноэлементной конструкции (результат выполнения scp-оператора)* показан пример scp-оператора удаления одноэлементной конструкции. В примере предполагается, что рассматриваемые элементы (***node1*** и ***node2***) изначально связаны между собой базовой sc-дугой принадлежности.

SCg-текст. Пример выполнения scp-оператора удаления одноэлементной конструкции (вызов scp-оператора)

=



SCg-текст. Пример выполнения scp-оператора удаления одноэлементной конструкции (результат выполнения scp-оператора)
=



Пункт 3.3.5.2. Операционная семантика Базового языка программирования ostis-систем

⇒ *ключевой знак**:
- *Абстрактная scp-машина*

Преимущества предложенного многоагентного подхода к обработке информации могут работать не только на платформенно-независимом уровне, но и на более низких уровнях. Так, в частности, интерпретатор *Базового языка программирования ostis-систем* также предлагается строить как *неатомарный абстрактный sc-агент*, обеспечивающий интерпретацию методов, описанных на *Языке SCP*. Таким образом, такой интерпретатор входит в общую иерархию агентов *ostis-системы* и является *абстрактным sc-агентом, не реализуемым на Языке SCP*.

В общем случае вариантов реализации таких интерпретаторов может быть много. В рамках *Стандарта OSTIS* один из них предлагается в качестве стандартного и называется *Абстрактной scp-машиной*.

Абстрактная scp-машина
⇒ *декомпозиция абстрактного sc-агента**:
- {• *Абстрактный sc-агент создания scp-процессов*
- *Абстрактный sc-агент интерпретации scp-операторов*
- *Абстрактный sc-агент синхронизации процесса интерпретации scp-программ*
- *Абстрактный sc-агент уничтожения scp-процессов*
- *Абстрактный sc-агент синхронизации событий в sc-памяти и ее реализации*
  ⇒ *декомпозиция абстрактного sc-агента**:
  - {• *Абстрактный sc-агент трансляции сформированной спецификации события в sc-памяти во внутреннее представление*
  - *Абстрактный sc-агент обработки события в sc-памяти, инициирующего агентную scp-программу*
  - }
- }

Задачей *Абстрактного sc-агента создания scp-процессов* является создание *scp-процессов*, соответствующих заданной *scp-программе*. Данный *sc-агент* активируется при появлении в *sc-памяти инициированного действия*, принадлежащего классу *действие интерпретации scp-программы*. После проверки *sc-агентом* условия инициирования выполняется создание *scp-процесса* с учетов конкретных параметров интерпретации *scp-программы*, после чего осуществляется поиск *начального оператора'scp-процесса* и добавление его во множество *настоящих сущностей*.

Задачей *Абстрактного sc-агента интерпретации scp-операторов* является собственно интерпретация операторов *scp-программы*, то есть выполнение в *sc-памяти* действий, описываемых конкретным *scp-оператором*. Данный *sc-агент* активируется при появлении в *sc-памяти scp-оператора*, принадлежащего классу *настоящих сущностей*.

После выполнения действия, описываемого *scp-оператором*, *scp-оператор* добавляется во множество *прошлых сущностей*. В случае когда семантика действия, описываемого *scp-оператором*, предполагает возможность ветвления *scp-программы* после выполнения данного *scp-оператора*, то используется одно из подмножеств класса *выполненных действий – безуспешно выполненное действие* или *успешно выполненное действие*.

Задачей *Абстрактного sc-агента синхронизации процесса интерпретации scp-программ* является обеспечение переходов между *scp-операторами* в рамках одного *scp-процесса*. Данный *sc-агент* активизируется при добавлении некоторого *scp-оператора* во множество *прошлых сущностей*. Далее осуществляется переход по *sc-дуге*, принадлежащей отношению *последовательность действий** (или более частным отношениям, в случае, если *scp-оператор* был добавлен во множество *успешно выполненных действий* или *безуспешно выполненных действий*). При этом очередной *scp-оператор* становится *настоящей сущностью* (активным *scp-оператором*) в том случае, если хотя бы один *scp-оператор*, связанный с ним входящими *sc-дугами*, принадлежащими отношению *последовательность действий** (или более частным отношениям), стал *прошлой сущностью* (или, соответственно, подмножеством прошлых сущностей). В случае, когда необходимо дождаться завершения выполнения всех предыдущих операторов, для синхронизации используется оператор класса *конъюнкция предшествующих операторов*.

Задачей *Абстрактного sc-агента уничтожения scp-процессов* является уничтожение *scp-процесса*, то есть удаление из *sc-памяти* всех *sc-элементов*, его составляющих. Данный *sc-агент* активируется при появлении в *sc-памяти scp-процесса*, принадлежащего множеству *прошлых сущностей*. При этом уничтожаемый *scp-процесс* необязательно должен быть полностью сформирован. Необходимость уничтожения не до конца сформированного *scp-процесса* может возникнуть в случае, если при создании *scp-процесса* возникли проблемы, не позволяющие продолжить создание *scp-процесса* и его выполнение.

Задачей *Абстрактного sc-агента синхронизации событий в sc-памяти и ее реализации* является обеспечение работы *неатомарных sc-агентов*, реализованных на *Языке SCP*.

Задачей ***Абстрактного sc-агента трансляции сформированной спецификации события в sc-памяти во внутреннее представление*** является трансляция ориентированных пар, описывающих *первичное условие инициирования** некоторого *sc-агента* во внутреннее представление элементарных событий на уровне *sc-хранилища*, при условии, что этот *sc-агент* реализован на платформенно-независимом уровне (с использованием *Языка SCP*). Условием инициирования данного *sc-агента* является появление в *sc-памяти* нового элемента множества *активных sc-агентов*, для которого будет найдена и протранслирована соответствующая ориентированная пара.

Задачей *Абстрактного sc-агента обработки события в sc-памяти, инициирующего агентную scp-программу*, является поиск *агентной scp-программы*, входящей во множество *программ sc-агента** для каждого *sc-агента*, первичное условие инициирования которого соответствует событию, произошедшему в *sc-памяти*, а также генерация и инициирование действия, направленного на интерпретацию этой программы. В результате работы данного *sc-агента* в *sc-памяти* появляется *инициированное действие*, принадлежащее классу *действие интерпретации scp-программы*.

§ 3.3.6. Решатели задач ostis-систем

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *машина обработки знаний*

С учетом того тезиса, что существуют *методы* интепретации других *методов* и, следовательно, иерархия *методов*, а также, соответственно, иерархия *навыков*, можно уточнить и понятие решателя задач, как иерархической системы навыков. Таким образом, определим *решатель задач ostis-системы* определяется как совокупность всех *навыков*, которыми обладает ostis-система на текущий момент времени.

Такой подход к уточнению архитектуры *решателей задач ostis-систем* позволяет обеспечить их модифицируемость, что, в свою очередь, позволяет *ostis-системе* при необходимости легко приобретать новые *навыки*, модифицировать (совершенствовать) уже имеющиеся, и даже избавляться от некоторых навыков с целью повышения производительности системы. Таким образом, имеет смысл говорить не о жестко фиксированном решателе задач, который разрабатывается один раз при создании первой версии системы и далее не меняется, а о совокупности навыков, фиксированной в каждый текущий момент времени, но постоянно эволюционирующей.

решатель задач ostis-системы
⇐ *семейство подмножеств**:
навык
:= [иерархическая система навыков, которыми обладает ostis-система]
⊃ *гибридный решатель задач ostis-системы*
:= [решатель задач ostis-системы, реализующий две и более модели решения задач]

⊃ *объединенный решатель задач ostis-системы*
  ≔ [полный решатель задач ostis-системы]
  ≔ [интегрированный решатель задач ostis-системы]
  ≔ [решатель задач ostis-системы, реализующий все ее функциональные возможности, как основные, так и вспомогательные]

В общем случае *объединенный решатель задач ostis-системы*, решает задачи, связанные с:
- обеспечением основных функциональных возможностей системы (например, решение явно сформулированных задач по требованию пользователя);
- обеспечением корректности и оптимизацией работы самой ostis-системы (перманентно на протяжении всего жизненного цикла ostis-системы);
- обеспечением повышения квалификации конечных пользователей и разработчиков ostis-системы;
- обеспечением автоматизации развития и управления развитием ostis-системы.

В свою очередь, под *машиной обработки знаний* будем понимать совокупность интерпретаторов всех *навыков*, составляющих некоторый *решатель задач*. С учетом многоагентного подхода к обработке информации, используемого в рамках *Технологии OSTIS*, *машина обработки знаний* представляет собой *sc-агент* (чаще всего – *неатомарный sc-агент*), в состав которого входят более простые sc-агенты, обеспечивающие интерпретацию соответствующего множества *методов*. Таким образом, *машина обработки знаний* в общем случае представляет собой иерархическую систему *sc-агентов*.

машина обработки знаний
⊂ *sc-агент*

Рассмотрим классификацию *решателей задач ostis-систем* по различным признакам.

Классификация *решателей задач ostis-систем* по типу соответствующей *ostis-системы*:

решатель задач ostis-системы
∋ *Решатель задач Метасистемы OSTIS*
⊃ *решатель задач вспомогательной ostis-системы*
  ⊃ *решатель задач интерфейса компьютерной системы*
    ⇒ *разбиение**:
      {• *решатель задач пользовательского интерфейса компьютерной системы*
      • *решатель задач интерфейса компьютерной системы с другими компьютерными системами*
      • *решатель задач интерфейса компьютерной системы с окружающей средой*
      }
  ⊃ *решатель задач ostis-подсистемы поддержки проектирования компонентов определенного класса*
    ⊃ *решатель задач ostis-подсистемы поддержки проектирования баз знаний*
      ⊃ *решатель задач повышения качества базы знаний*
        ⊃ *решатель задач верификации базы знаний*
          ⊃ *решатель задач поиска и устранения некорректностей в базе знаний*
          ⊃ *решатель задач поиска и устранения неполноты*
        ⊃ *решатель задач оптимизации структуры базы знаний*
        ⊃ *решатель задач выявления и устранения информационного мусора*
    ⊃ *решатель задач ostis-подсистемы поддержки проектирования решателей задач ostis-систем*
      ⇒ *разбиение**:
        {• *решатель задач ostis-подсистемы поддержки проектирования программ обработки знаний*
        • *решатель задач ostis-подсистемы поддержки проектирования агентов обработки знаний*
        }
  ⊃ *решатель задач подсистемы управления проектирования компьютерных систем и их компонентов*
⊃ *решатель задач самостоятельной ostis-системы*

Классификация *решателей задач ostis-систем* по типу интерпретируемой *модели решения задач*:

решатель задач ostis-системы
⊃ *решатель задач с использованием хранимых методов*
  ≔ [решатель, способный решать задачи тех классов, для которых на данный момент времени известен соответствующий метод решения]
  ⊃ *решатель задач на основе нейросетевых моделей*
  ⊃ *решатель задач на основе генетических алгоритмов*

- ⊃ *решатель задач на основе императивных программ*
  - ⊃ *решатель задач на основе процедурных программ*
  - ⊃ *решатель задач на основе объектно-ориентированных программ*
- ⊃ *решатель задач на основе декларативных программ*
  - ⊃ *решатель задач на основе логических программ*
  - ⊃ *решатель задач на основе функциональных программ*

⊃ *решатель задач в условиях, когда метод решения задач данного класса в текущий момент времени не известен*

- ≔ [решатель, реализующий стратегии решения задач, позволяющие породить метод решения задачи, который в текущий момент времени не известен ostis-системе]
- ≔ [решатель, использующий для решения задач метаметоды, соответствующие более общим классам задач по отношению к заданной]
- ≔ [решатель задач, позволяющий породить метод, который является частным по отношению какому-либо известному ostis-системе методу и интерпретируется соответствующей машиной обработки знаний]
- ⊃ *решатель, реализующий стратегию поиска путей решения задачи в глубину*
- ⊃ *решатель, реализующий стратегию поиска путей решения задачи в ширину*
- ⊃ *решатель, реализующий стратегию проб и ошибок*
- ⊃ *решатель, реализующий стратегию разбиения задачи на подзадачи*
- ⊃ *решатель, реализующий стратегию решения задач по аналогии*
- ⊃ *решатель, реализующий концепцию интеллектуального пакета программ*

Отдельно выделим классификацию машин обработки знаний, которые в общем случае могут соответствовать одним и тем же фрагментам базы знаний, но при этом в совокупности с ними образовывать разные навыки и соответственно разные решатели задач:

машина обработки знаний

⊃ *машина логического вывода*

- ⊃ *машина дедуктивного вывода*
  - ⊃ *машина прямого дедуктивного вывода*
  - ⊃ *машина обратного дедуктивного вывода*
- ⊃ *машина индуктивного вывода*
- ⊃ *машина абдуктивного вывода*
- ⊃ *машина нечеткого вывода*
- ⊃ *машина вывода на основе логики умолчаний*
- ⊃ *машина логического вывода с учетом фактора времени*

Классификация *решателей задач ostis-систем* по типу решаемой *задачи* (цели решения задачи):

решатель задач ostis-системы

⊃ *решатель задач информационного поиска*

- ⇒ *разбиение**:
  - {• *решатель задач поиска информации, удовлетворяющей заданным критериям*
  - • *решатель задач поиска информации, не удовлетворяющей заданным критериям*
  - }

⊃ *решатель явно сформулированных задач*

- ≔ [решатель задач, для которых явно сформулирована цель]
- ⊃ *решатель задач поиска или вычисления значений заданного множества величин*
- ⊃ *решатель задач установления истинности заданного логического высказывания в рамках заданной формальной теории*
- ⊃ *решатель задач формирования доказательства заданного высказывания в рамках заданной формальной теории*
- ⊃ *машина верификации ответа на указанную задачу*
- ⊃ *машина верификации решения указанной задачи*
  - ⊃ *машина верификации доказательства заданного высказывания в рамках заданной формальной теории*

⊃ *решатель задач классификации сущностей*

- ⊃ *машина соотнесения сущности с одним из заданного множества классов*
- ⊃ *машина разделения множества сущностей на классы по заданному множеству признаков*

⊃ *решатель задач синтеза информационных конструкций*

- ⊃ *решатель задач синтеза естественно-языковых текстов*
- ⊃ *решатель задач синтеза изображений*
- ⊃ *решатель задач синтеза сигналов*

- ⊃ *решатель задач синтеза речи*
- ⊃ *решатель задач анализа информационных конструкций*
  - ⊃ *решатель задач анализа естественно-языковых текстов*
    - ⊃ *решатель задач понимания естественно-языковых текстов*
    - ⊃ *решатель задач верификации естественно-языковых текстов*
  - ⊃ *решатель задач анализа изображений*
    - ⊃ *решатель задач сегментации изображений*
    - ⊃ *решатель задач понимания изображений*
  - ⊃ *решатель задач анализа сигналов*
    - ⊃ *решатель задач анализа речи*
      - ⊃ *решатель задач понимания речи*

§ 3.3.7. Принципы решения задач распределенными коллективами ostis-систем

Разработка *решателей задач интеллектуальных систем* на настоящий момент как правило рассматриваются в контексте одиночных (самостоятельных) интеллектуальных систем, функционирующих в некоторой среде (частью которой является и пользователь, если он есть). В то же время очевидна тенденция современных информационных технологий к переходу от одиночных систем к коллективам распределенных взаимодействующих компьютерных систем, в частности, к распределенному хранению данных и распределенным вычислениям. В случае интеллектуальных компьютерных систем важнейшим свойством систем, входящих в такие коллективы, становится интероперабельность, то есть способность системы к согласованному взаимодействию с другими подобными системами с целью решения каких-либо задач. Таким образом, особо актуальным является переход от разработки *решателей задач* отдельно взятых интеллектуальных систем к решателям задач взаимодействующих *интероперабельных интеллектуальных систем*, включая разработку принципов решения задач в таких распределенных коллективах с учетом решения всех обозначенных выше проблем. Важно отметить, что полностью отказаться от распределенности при решении задач даже в сравнительно простых прикладных системах нельзя, поскольку часто интеллектуальные системы вынуждены использовать различные датчики и эффекторы, которые с точки зрения общей архитектуры являются некоторыми внешними модулями (внешними агентами) и, таким образом, привносят распределенность в общую архитектуру системы.

Для решения данной проблемы предлагается рассмотреть такую систему взаимодействующих *интеллектуальных компьютерных систем* как *многоагентную систему* и уточнить принцип поведения *агентов* в такой системе.

Таким образом, можно говорить о двух видах многоагентных систем в рамках *Технологии OSTIS*:
- внутренняя система sc-агентов над общей sc-памятью в рамках некоторой ostis-системы;
- распределенная система ostis-систем в рамках Экосистемы OSTIS.

В обоих случаях можно говорить об иерархии агентов:
- в рамках внутренней системы sc-агентов выделяются *атомарные абстрактные sc-агенты* и *неатомарные абстрактные sc-агенты*, кроме того существует иерархия sc-агентов с точки зрения языка интерпретации методов (см. *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*);
- в рамках *Экосистемы OSTIS* выделяются как *индивидуальные ostis-системы*, так и *коллективные ostis-системы*, которые в свою очередь могут состоять как из *индивидуальных ostis-систем*, так и *коллективных ostis-систем* (см. *§ 1.3.2. Семантически совместимые ostis-системы*).

Ключевым отличием *распределенной системы ostis-систем* от *внутренней системы sc-агентов* в рамках *индивидуальной ostis-системы* является отсутствие общей памяти, хранящей общую для всех *sc-агентов базу знаний* и выступающей в роли среды для коммуникации *sc-агентов*. В общем случае в качестве средства коммуникации между агентами в рамках выделенных систем агентов может использоваться:
- Общая нераспределенная (монолитная) память, как в случае *sc-агентов* над *sc-памятью*;
- Общая распределенная память. В этом случае с логической точки зрения агенты могут считать, что по-прежнему работают над общей памятью, в рамках которой хранится вся доступная база знаний, однако реально *база знаний* будет распределена между несколькими *ostis-системами* и выполняемые преобразования должны будут синхронизироваться между этими ostis-системами;
- Специализированные каналы связи. Очевидно, что при решении задачи в распределенном коллективе *ostis-систем* должны существовать языковые и технические средства, позволяющие осуществлять передачу сообщений от одной *ostis-системы* к другой.

Все перечисленные средства коммуникации в зависимости от класса решаемой задачи, требуемых для ее решения *знаний* и *навыков*, а также существующего (доступного) в данный момент набора *ostis-систем* могут комбинироваться.

В основу решения задач в рамках *распределенного коллектива ostis-систем* предлагается положить идею максимально возможной унификации и конвергенции принципов решения задач в рамках *индивидуальной ostis-системы* и *распределенного коллектива ostis-систем*. Такой подход обладает следующим важным достоинством: если общие принципы решения задач не зависят от того, какой конкретно набор *ostis-систем* участвует в решении той или иной задачи, то становится возможным легко переходить от *индивидуальной ostis-системы* к *распределенному коллективу ostis-систем* при ее усложнении без необходимости существенно пересматривать коллектив *агентов*, входящих в состав такой *ostis-системы* и заново продумывать используемый подход к решению задач того или иного класса. Для перехода от *индивидуальной ostis-системы* к *коллективной ostis-системе* достаточно выполнить следующие шаги:

- Разделить множество классов задач, решаемых данной *ostis-системой*, на семейство подмножеств, каждое из которых обладает некоторой логической целостностью, критерии которой в общем случае определяются разработчиком. При этом указанные подмножества могут пересекаться, но при объединении должны давать исходное множество, таким образом необходимо построить одно из возможных *покрытий** для множества классов задач, решаемых данной *ostis-системой*;
- Для каждого из выделенных подмножеств необходимо сформировать множество *знаний* и *навыков*, необходимых для решения задач данного множества классов. При этом в общем случае может оказаться необходимым пересмотр иерархии навыков и соответствующих им sc-агентов, в частности, преобразование некоторых атомарных sc-агентов в неатомарные. Теоретически избежать такой ситуации невозможно, однако подобные ситуации можно практически исключить на этапе проектирования решателей задач индивидуальных ostis-систем, делая иерархию агентов достаточно глубокой и ставя в соответствие *атомарным sc-агентам* такие *классы задач*, разделение которых на подклассы с практической точки зрения не имеет смысла.
  Аналогичная ситуация может возникнуть и при выделении фрагментов *базы знаний*. В этом случае может потребоваться пересмотр иерархии *предметных областей* и *онтологий* и, возможно, выделение новых предметных областей. Как и в случае с *решателями задач*, избежать такой ситуации на практике возможно в случае, если иерархия предметных областей будет достаточно глубокой для того, чтобы выделение более частных предметных областей было практически нецелесообразным;
- Каждое сформированное таким образом множество знаний и навыков становится соответственно базой знаний и решателем задач новой ostis-системы, которая будет способна реализовать только часть функциональных возможностей исходной ostis-системы.

Такое разделение может выполняться итерационно и для полученных *ostis-систем* в общем случае неограниченное количество раз, создавая на каждой итерации новое "поколение" ostis-систем, полученное путем декомпозиции исходной *ostis-системы*.

Таким образом, предлагаемая идея унификации принципов решения задач в *ostis-системах* любого рода позволяет

- с практической точки зрения снять ограничение на расширение функциональных возможностей (обучение) не только *индивидуальной ostis-системы*, но и *коллективной ostis-системы*, позволяя, таким образом, постоянно наращивать функциональные возможности *Экосистемы OSTIS* в целом.
- с теоретической (архитектурной) точки зрения говорить о фрактальном характере не только внутренней организации *ostis-систем* но и *коллективов ostis-систем*, что, в свою очередь, позволяет обеспечить возможность наследования и других принципов построения *индивидуальных ostis-систем* в *распределенных коллективах ostis-систем*, включая, например, методику проектирования *ostis-систем* и их компонентов и соответствующие средства, а также принципы синхронизации соответствующих sc-агентам параллельных *информационных процессов*.

В основе взаимодействия *sc-агентов* в рамках *индивидуальной ostis-системы* лежит уточненный принцип "доски объявлений" при котором агенты взаимодействуют посредством общей для них sc-памяти (см. *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*). Для реализации той же идеи в случае *распределенной коллективной ostis-системы* необходимо выбрать какую-либо sc-память для выполнения данной роли. При решении задач в *распределенном коллективе ostis-систем* возможны два варианта организации взаимодействия *агентов* (которыми являются и сами *ostis-системы*):

- Если решаемая задача достаточно сложная и требует частого обращения к нескольким отдельным *ostis-системам*, то целесообразно путем объединения раздельных ostis-систем создать *временную ostis-систему*, где все sc-агенты, входившие в состав исходных *ostis-систем*, становятся внутренними, и принципы организации их взаимодействия известны. В этом случае существенно снижаются затраты на решение задачи, но появляются накладные расходы на создание таких *временных ostis-систем*. Таким образом, необходимо отдельно разработать критерии на основании которых будет приниматься решение о целесообразности такого объединения. Отметим, что для того, чтобы иметь возможность сохранить результат и ход решения задачи для последующего применения целесообразно осуществлять объединение *ostis-систем* на базе одной из *ostis-систем*, входящих в такое объединение, а не создавать совершенно новую *ostis-систему*. При этом в такую систему будут копироваться знания и навыки из объединяемых систем, а сами эти объединяемые системы

могут вообще никак не меняться. Тогда после решения задачи из исходной *ostis-системы* необходимо будет исключить те *навыки* и *знания*, которые были нужны только для решения данной задачи.

Важно отметить, что описанная интеграция *ostis-систем* благодаря особенностям их архитектуры выполняется значительно проще, чем в других компьютерных системах, поскольку принципы построения и баз знаний, и *решателей задач ostis-систем* изначально предполагают возможность неограниченного расширения имеющихся в системе знаний и навыков без необходимости внесения изменений в уже имеющуюся *базу знаний* и *решатель задач*. Таким образом, интеграция двух ostis-систем при условии их семантической совместимости сводится к обычному теоретико-множественному объединению их *баз знаний* и *решаталей задач* и последующему исключению продублированных компонентов. Благодаря этому создание таких временных ostis-систем может выполняться автоматически, что делает применение такого подхода к организации решения задач целесообразным во многих случаях.

- Другой возможный вариант предполагает, что в качестве среды для взаимодействия *sc-агентов* (как внешних, так и внутренних, внешняя *ostis-система* с точки зрения процесса решения задачи также рассматривается как *sc-агент*) выбирается *sc-память* одной из *ostis-систем*, входящих в состав *коллектива ostis-систем*. Предлагаются следующие критерии выбора этой sc-памяти:
  - Если задача решается неоднократно в рамках некоторого *ostis-сообщества* (*сообщества ostis-систем* и их пользователей, см. *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*), то для координации действий *sc-агентов* выбирается *sc-память корпоративной ostis-системы* для данного *ostis-сообщества*;
  - Если *коллектив ostis-систем* для решения данной задачи формируется временно (разово), то для координации действий *sc-агентов* выбирается *sc-память* той *ostis-системы*, которая инициировала решение данной задач.

  Недостатком данного варианта является наличие затрат на коммуникацию между *ostis-системами*. Если по каким-либо причинам эти затраты велики (например, из-за низкого качества соединения между системами), то более целесообразно использовать первый из предложенных вариантов.

В любом из предложенных вариантов в конечном итоге определяется некоторая конкретная sc-память, которая становится средой для взаимодействия агентов, осуществляющих решение задачи, по изложенным в *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах* принципам. Тогда можно уточнить понятие *sc-агента* как компонента решателя задач в контексте распределенного решения задач *коллективом ostis-систем* и считать sc-агентом не только компонент *решателя задач индивидуальной ostis-системы*, но и любую ostis-систему, входящую в постоянный либо временный *коллектив ostis-систем*, решающих какие-либо задачи, поскольку принципы взаимодействия *ostis-систем* в таком коллективе полностью совпадают с принципами взаимодействия *sc-агентов* в составе *решателя задач индивидуальной ostis-системы*.

Таким образом, можно говорить о фрактальной иерархической структуре распределенного *гибридного решателя задач*, в рамках которой выделяется два варианта иерархии *sc-агентов*:

- Иерархия sc-агентов с точки зрения уровня *языков представления методов*, на которых представлены соответствующие этим sc-агентам методы. В рамках этой иерархии в свою очередь можно выделить три уровня, имеющих важные отличия:
  - Уровень *sc-агентов ostis-платформы*, обеспечивающий интерпретацию методов платформенно-независимого уровня в рамках *индивидуальной ostis-системы*, в рамках которого может выделяться иерархия языков представления методов уровня ostis-платформы и соответствующих средств их интерпретации;
  - Уровень *платформенно-независимых sc-агентов* в рамках *индивидуальной ostis-системы*, в рамках которого может выделяться иерархия платформенно-независимых языков представления методов;
  - Уровень *распределенных коллективов ostis-систем*, на котором также можно говорить о *языках представления методов* и их иерархии, но при этом в общем случае даже отдельные методы могут физически храниться распределенно в разных ostis-системах. Например, можно говорить о *языке представления методов* для финансовой деятельности крупных предприятий, но при этом целесообразно выделять подъязыки для описания деятельности отделов различных категорий и иметь отдельные ostis-системы для обслуживания каждого из отделов.
- Иерархия sc-агентов с точки зрения атомарности/неатомарности в рамках одного *языка представления методов*. Формирование такой иерархии может быть целесообразным на любом уровне языка *языка представления методов* и приводит к выделению:
  - *атомарных платформенно-зависимых sc-агентов* и *неатомарных платформенно-зависимых sc-агентов* на уровне *ostis-платформы*;
  - *атомарных платформенно-независимых sc-агентов* и *неатомарных платформенно-независимых sc-агентов* на платформенно-независимом уровне в рамках индивидуальной ostis-системы;
  - *индивидуальных ostis-систем* и *коллективных ostis-систем* на уровне решения задач в рамках *Экосистемы OSTIS*.

Дальнейшее развитие представленных принципов решения задач распределенными коллективами ostis-систем предполагает:

- Разработку формальных критериев для оценки целесообразности или нецелесообразности формирования временных индивидуальных ostis-систем;
- Разработку языка и принципов обмена сообщениями между ostis-системами, входящими в коллектив ostis-систем, решающий какую-либо задачу. Несмотря на то, что с логической точки зрения каждая ostis-система трактуется как sc-агент и принципы их взаимодействия остаются теми же, реализация, например, возможности реагирования на события в базе знаний и внесения изменений в эту базу знаний для внутренних sc-агентов и внешних ostis-систем будет отличаться и требует уточнения.

§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач

В данной главе был детально рассмотрен подход к построению *решателей задач*, позволяющий решить ряд фундаментальных проблем в области построения *решателей задач*, таких как обеспечение совместимости различных *решателей задач* и их компонентов, а также обеспечение обучаемости (модифицируемости и рефлексивности) самих *решателей задач*. В то же время существует существует ряд проблем, остающихся актуальными и требующих решения.

Первая проблема связана с отсутствием достаточно строгой формализованной классификации задач, решаемых интеллектуальными системами, отсутствием унификации описания задач и классов задач, описания целей, хода и результата решения задачи, методов решения задач, связей между классами задач и методами решения задач данного класса. Решение данной проблемы, с одной стороны, позволит обеспечить возможность глубокой интеграции всевозможных *моделей решения задач* различных классов и возможность облегчить процесс интеграции новых моделей решения задач в интеллектуальную систему, а с другой стороны, станет предпосылкой для решения других проблем, описанных ниже.

Вторая проблема заключается в том, что на настоящий момент основное внимание в области разработки *гибридных решателей задач* уделено снижению трудоемкости интеграции различных компонентов решателя задач в *интеллектуальную систему* и реализации возможности накопления многократно используемых компонентов *решателей задач*, однако в общем случае не говорится о том, как конкретно *интеллектуальная система* будет применять те или иные компоненты при решении задач конкретных классов. Таким образом, построение общего плана решения задачи, то есть выбор методов решения задач, определение порядка их применения и выбор исходных данных (аргументов) для применения того или иного метода, фактически определяется разработчиком на этапе проектирования системы или на этапе ее эволюции в процессе эксплуатации. Предпосылкой для решения данной проблемы является решение ранее рассмотренной проблемы унификации представления задач различных классов и методов их решения. Решение же рассматриваемой проблемы предполагает разработку комплекса *стратегий решения задач* (или *метаметодов решения задач*), которые позволят *интеллектуальной системе* самостоятельно формировать план решения задачи с учетом имеющихся в системе методов решения задач и, при возможности, даже запрашивать недостающие для решения задачи компоненты в соответствующих библиотеках. Следует отметить, что попытки разработки универсальных высокоуровневых подходов к решению задач предпринимались еще на заре развития *Искусственного интеллекта*, в 1950-60ые годы, однако не увенчались успехов и вскоре прекратились. Во многом это связано с отсутствием на тот момент унифицированных моделей представления и обработки знаний, которые в настоящий момент предлагаются в рамках *Технологии OSTIS*.

Еще одна актуальная проблема, тесно связанная с рассмотренными выше, заключается в том, что интеллектуальные системы часто вынуждены решать задачи в условиях так называемых не-факторов, то есть неполноты описания задачи и возможных путей ее решения, нечеткости и некорректности имеющихся знаний, отсутствия критериев для оценки оптимальности полученного решения и т.д. (см. *Нариньяни А.С.нФактоКВ-2004ст*). В особенности это актуально при решении поведенческих задач, связанных с изменением состояния объектов среды, внешней по отношению к интеллектуальной системе. Для решения задач в подобных условиях интеллектуальная система должна не только обладать достаточным набором компонентов решателя задач, реализующих модели решения задач в условиях наличия не-факторов (нечеткие логические модели, модели машинного обучения, генетические алгоритмы и так далее), но и реализовывать *стратегии решения задач*, которые бы позволили принимать решения и формировать *план решения задачи* в такого рода условиях.

Рассмотренные проблемы связаны в первую очередь с процессом решения конкретной задачи интеллектуальной системой. В то же время очевидно, что в каждый момент времени интеллектуальная система вынуждена параллельно решать несколько задач, которые могут быть связаны как с непосредственным функциональным назначением системы, так и с обеспечением жизнедеятельности и эволюции самой системы. Во втором случае имеются в виду, в частности, задачи, связанные с актуализацией имеющихся у нее сведений о внешнем мире, поиском и устранением ошибок в базе знаний, оптимизацией структуры *базы знаний* и *решателя задач* системы, поиском и устранением

информационного мусора и многие другие. При этом разные задачи могут иметь разный приоритет, который может меняться в зависимости от ситуации даже в процессе ее решения. В то же время, в ситуации, когда априори не известно, какой из возможных способов решения задачи окажется наиболее эффективным, может оказаться целесообразным параллельное использование нескольких подходов к решению одной и той же задачи. Таким образом, актуальной является проблема организации управления информационными процессами решения задач в интеллектуальной системе и взаимодействия параллельно выполняемых информационных процессов с учетом приоритетности процессов, возможности отслеживать текущее состояние *информационных процессов*, порождать, приостанавливать и уничтожать информационные процессы. Для решения данной проблемы целесообразно заимствовать решения, широко используемые в традиционных компьютерных системах, в частности, реализуемые в современных операционных системах, и адаптировать их к специфике решения задач в интеллектуальных системах. Важно отметить, что реализация модели управления информационными процессами на основе общих унифицированных моделей обработки информации, предлагаемых в рамках *Технологии OSTIS*, позволит сделать одни информационные процессы объектом анализа других информационных процессов, что, в свою очередь, даст возможность анализировать ход решения задачи непосредственно в процессе решения, оценивать эффективность тех или иных методов решения задач, накапливать наиболее удачные решения для применения в дальнейшем для решения аналогичных задач и многое другое.

Решение перечисленных проблем позволит разработать принципиально новую иерархическую модель *гибридного решателя задач*, обладающую рядом существенных преимуществ, которая, в свою очередь, должна будет интерпретироваться на каких-либо платформах. Без унификации требований к *ostis-платформе* и четкого разделения платформенно-независимой модели системы (и в частности решателя) и *ostis-платформы* невозможно говорить о реализации модели *решателя задач*, реализующей рассмотренные выше идеи. Это приведет к необходимости дублирования одних и тех же компонентов модели для разных платформ, значительно усложнит интеграцию компонентов *решателя задач*, поскольку потребует учета при такой интеграции особенностей каждой *ostis-платформы*. Кроме того, четкое разделение уровня модели системы и уровня *ostis-платформы* даст возможность независимо друг от друга развивать различные платформы и модели интеллектуальных систем. Таким образом, предлагается сформулировать унифицированные требования к *ostis-платформе*, а также построить общую модель такой *ostis-платформы*, удовлетворяющую указанным требованиям. Более подробно такая модель *ostis-платформы* рассмотрена в *Главе 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*.

С другой стороны, как уже было сказано, *решатель задач* представляет собой сложную систему, ориентированную на работу со знаниями, а не с данными, в отличие от современных программных систем, в которых изначально известно, где конкретно локализованы нужные данные и в какой форме они представлены. В связи с этим, применение для разработки интеллектуальных систем современных программно-аппаратных платформ, ориентированных на адресный доступ к хранящимся в памяти данным, не всегда оказывается эффективным, поскольку при разработке интеллектуальных систем фактически приходится моделировать нелинейную память на базе линейной. Повышение эффективности решения задач интеллектуальными системами требует разработки специализированных платформ, в том числе аппаратных, ориентированных на унифицированные семантические модели представления и обработки информации. В качестве основы для таких разработок предлагается использовать предложенную в рамках *Технологии OSTIS* общую концепцию *ассоциативного семантического компьютера*, *семантической памяти* и базового языка программирования, ориентированного на обработку информации в такой памяти, и дополнить их идеями *волновых языков программирования*, *инсерционного программирования* и других подходов, направленными на повышение эффективности обработки знаний, в том числе на аппаратном уровне. Более подробно концепция *ассоциативного семантического компьютера* рассматривается в *Главе 6.2. Ассоциативные семантические компьютеры для ostis-систем*.

В случае же распределенного коллектива интеллектуальных систем важнейшей проблемой является не просто обеспечение возможности решения задач таким коллективом в текущий момент времени, а перманентная поддержка семантической совместимости и, как следствие, интероперабельности систем, входящих в такой коллектив на протяжении всего их жизненного цикла. Очевидно, что каждая из систем, входящих в такой коллектив, и, соответственно, ее *решатель задач* может эволюционировать независимо от других систем, но при этом всегда должна сохраняться *интероперабельность* между системами, в противном случае решение задач в таком коллективе станет невозможным. Решение данной проблемы предполагает разработку методов перманентного анализа *семантической совместимости* распределенного коллектива взаимодействующих интеллектуальных систем, выявления и устранения проблем.

Для решения указанных проблем предлагается на основе подхода к построению гибридных решателей задач, рассмотренного в данной главе, разработать:

- Комплексную онтологию действий, задач и методов их решения, а также онтологию *гибридных решателей задач* на основе которой уточнить понятие решателя и его архитектуру. В *Главе 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии* представлена первая версия *Глобальной предметной области действий и задач и соответствующая ей онтологии методов и технологий*, на ее основе предлагается разработать комплексную онтологию действий и задач, решаемых *ostis-системами*;

- Комплекс унифицированных обобщенных стратегий (метаметодов) решения задач в интеллектуальных системах, позволяющий интеллектуальной системе самостоятельно формировать план решения задачи с учетом имеющихся в системе *методов решения задач*. В основу разрабатываемых стратегий кроме опыта аналогичных работ предлагается внести также некоторые общеметодологические идеи, связанные с *Теорией бихевиоризма* и набирающими популярность идеями ее применения в информатике (см. *Cao O.IndepBUaUtBI-2010art*, *Cao O..BehavIaNP-2014art*, *Pavel M..BehavIaCMiSoP-2015art*), ТРИЗ (см. *Альтшуллер Г.С..НайтиИВвТРИ-2010кн*), а также *СМД-методологией*, предложенной школой Г. П. Щедровицкого (см. *Щедровицкий Г.П.СхемаМССС-1995кн*);
- Онтологическую модель формирования плана решения задачи и управления процессом решения задач в гибридных решателях задач в условиях различных не-факторов и отсутствия четких критериев оценки оптимальности полученного решения. Для разработки данной модели предлагается адаптировать теорию *ситуационного управления* (см. *Поспелов Д.А.СитуаУТиП-1986кн*), и реализовать ее в контексте семантической теории *решателей задач*, разрабатываемой в рамках *Технологии OSTIS*;
- Комплексную онтологическую модель управления информационными процессами решения задач в интеллектуальных системах, построенных на базе унифицированных семантических моделей представления и обработки информации;
- Онтологическую модель платформы интерпретации унифицированных семантических моделей представления и обработки информации (*ostis-платформы*). Первая версия данной модели рассмотрена в *Главе 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*;
- Комплексную иерархическую модель *гибридного решателя задач*, основанную на многоагентном подходе и учитывающую необходимость решения задач как в рамках одиночных интеллектуальных систем, так и в рамках распределенных коллективов интероперабельных интеллектуальных систем;
- Комплекс методов анализа качества *гибридных решателей задач* и их компонентов;
- Комплекс методик и средств поддержки проектирования *гибридных решателей задач*. Первая версия такой методики и средств рассмотрена в *Главе 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*.

Заключение к Главе 3.3.

В данной главе рассмотрены актуальные на сегодняшний день проблемы в области разработки *гибридных решателей задач* и предложен общий подход к построению *гибридных решателей задач*, который решает такие проблемы, как обеспечение совместимости и модифицируемости *решателей задач*, а также создает предпосылки к решению других актуальных проблем, подробнее рассмотренных в *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*.

Сформулируем ряд конкретных направлений развития предложенных в данной главе подходов:
- Более тесно и полно интегрировать идеи ситуационного управления в предлагаемый подход;
- Доработать предложенный механизм блокировок, в частности, минимизировать число классов блокировок, учесть и реализовать идеи реализации lock-free алгоритмов;
- Исключить необходимость введения *sc-метаагентов* и *scp-метапрограмм*.
- Доработать *Язык SCP* до того, чтобы иметь возможность описывать в рамках *scp-программ* рецепторное и эффекторное взаимодействие *ostis-систем*.
- При разработке *Абстрактной scp-машины* учесть принципы построения волновых языков программирования (см. *Сапатый П.С.ЯзыкВкОНС-1986ст*, *Moldovan D.I..SNAP aVLSIAfAIP-1985art*) и идеи инсерционного программирования и моделирования (см. *Летичевский А.А..ИнсерП-2003ст*, *Летичевский А.А..ИнсерМ-2012ст*).

Глава 3.4.
Язык вопросов для ostis-систем

⇒ *авторы**:
- *Самодумкин С. А.*
- *Зотов Н. В.*
- *Шункевич Д. В.*
- *Ивашенко В. П.*

⇒ *аннотация**:
[Возможности *баз знаний ostis-систем* позволяют не только представлять и структурировать знания об окружающем мире, но и быстро получать и формировать эти знания о нем, тем самым удовлетворяя информационную потребность пользователя. В данной главе уточнена формальная спецификация *Языка вопросов для ostis-систем*, позволяющая описывать и интерпретировать любые классы *вопросов пользователей ostis-систем*.]

⇒ *подраздел**:
- *§ 3.4.1. Синтаксис Языка вопросов для ostis-систем*
- *§ 3.4.2. Денотационная семантика Языка вопросов для ostis-систем*
- *§ 3.4.3. Операционная семантика языка вопросов для ostis-систем*

⇒ *ключевой знак**:
- *Язык вопросов для ostis-систем*

⇒ *ключевое понятие**:
- *вопрос*
- *ответ на вопрос*
- *знак в рамках заданного вопроса*
- *основной знак в рамках заданного вопроса*
- *неосновной знак в рамках заданного вопроса*
- *отношение в рамках заданного вопроса*
- *базовое отношение в рамках заданного вопроса*
- *интерпретатор Языка вопросов для ostis-систем*

⇒ *библиографическая ссылка**:
- *Аверьянов Л.Я.ПочемЛЗВ-1993ст*
- *Сулейманов Дж.Ш.ИсслеБППС-2001ст*
- *Сулейманов Дж.Ш.СистеСАОТ-2014ст*
- *Бухараев Р.Г..СеманАвВОС-1990кн*
- *Kwok C.CT..ScaliQAttW-2001art*
- *Emel'yanov G.M..Analy oSRiCoSIioS-2007art*
- *Финн В.К.ЛогичПИП-1976кн*
- *Финн В.К.кФормаОПИПС-1981кн*
- *Белнап Н..ЛогикВиО-1981кн*
- *Sosnin P.QuestMfCDoS-2007art*
- *Захаров В.П.ИнфорСДП-2002ст*
- *Хант Э.ИскусИ-1978кн*
- *Любарский Ю.Я.ИнтелИС-1990кн*
- *Самодумкин С.А.СеманТПИВ-2009ст*
- *Самодумкин С.А.ТехноПИВС-2009ст*

Введение в Главу 3.4.

Одна из ключевых особенностей *интеллектуальной системы* состоит в том, что *пользователь* имеет возможность формулировать свою информационную потребность. Способом выражения такой потребности является *вопрос*. В процессе общения всегда существует контекст, который определяет дополнительную информацию, способствующую правильному пониманию *смысла* сообщения. Особенность представления информации в *базах знаний*

ostis-систем упрощает формирование информационной потребности пользователя, так как представленная информация в *базах знаний* уже структурирована и известны отношения, заданные на определенном понятии, в отношении которого разрешается вопросно-проблемная ситуация. В работе *Аверьянов Л.Я.ПочемЛЗВ-1993ст* показано, что вопросно-проблемная ситуация не может быть решена в рамках формальной логики и природа вопроса может быть понятна в системе субъектно-объектных отношений. В связи с тем, что при формировании *баз знаний ostis-систем* происходит формирование субъектно-объектных отношений в рамках заданной *предметной области*, тем самым упрощается выражение информационной потребности пользователем средствами *SC-кода*.

Целью разработки ***Языка вопросов для ostis-систем*** и последующего его развития является реализация возможности понимания действий, осуществляемых *ostis-системой*, при формировании ответа на поставленный *вопрос*. В процессе формирования ответа на поставленный *вопрос* возможны следующие варианты:

- *ответ на* поставленный *вопрос* существует в *базе знаний* и происходит локализация *фрагмента базы знаний* в контексте выраженной средствами *SC-кода* информационной потребности *пользователя*;
- ответ связан с разрешением некоторой задачной ситуации, которая содержится в контексте *вопроса* и формирование *ответа на вопрос* возлагается на *решатель задач* (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*).

Язык вопросов для ostis-систем
:= [Предлагаемый нами вариант языка для описания вопросов и ответов на них в ostis-системах]
∈ *sc-язык*
⇒ *синтаксис языка**:
Синтаксис Языка вопросов для ostis-систем
⊂ *Синтаксис SC-кода*
⇒ *денотационная семантика языка**:
Денотационная семантика Языка вопросов для ostis-систем
:= [Онтология классов знаков и отношений для описания формулировок вопросов на SC-коде]
⊃ *Семантическая классификация вопросов*
⇒ *операционная семантика языка**:
Операционная семантика Языка вопросов для ostis-систем
:= [Коллектив sc-агентов вывода ответов на заданные вопросы пользователя ostis-системы]

§ 3.4.1. Синтаксис Языка вопросов для ostis-систем

Язык вопросов для ostis-систем относится к семейству семантических совместимых языков — *sc-языков* и предназначен для формального описания поискового предписания *ostis-систем* с целью удовлетворения информационной потребности *пользователя*. Поэтому ***Синтаксис Языка вопросов для ostis-систем***, как и *синтаксис* любого другого *sc-языка*, является *Синтаксисом SC-кода*. Такой подход позволяет:

- унифицировать форму представления *вопросов* и *знаний*, с помощью которых строятся ответы на поставленные *вопросы*;
- использовать минимум средств для интерпретации заданных *вопросов пользователей*;
- сводить формирование ответов на большую часть заданных *вопросов* к поиску информации в текущем состоянии *базы знаний ostis-системы*.

§ 3.4.2. Денотационная семантика Языка вопросов для ostis-систем

⇒ *ключевое понятие**:
- *вопрос*
- *ответ на вопрос*
- *знак в рамках заданного вопроса*
- *основной знак в рамках заданного вопроса*
- *неосновной знак в рамках заданного вопроса*
- *отношение в рамках заданного вопроса*
- *базовое отношение в рамках заданного вопроса*

Денотационная семантика Языка вопросов для ostis-систем включает *классы вопросов* и соответствующие *классы ответов*, необходимые для спецификации формулировок *вопросов* и *ответов* на них, а также *классы знаков* и *отношений*, входящих в структуру любого *вопроса*. В *Семантической классификации вопросов Языка вопросов для ostis-систем* заложена идея, описанная в работе *Сулейманов Дж.Ш.ИсслеБППС-2001ст*.

Любой ***вопрос*** на *Языке вопросов для ostis-систем* представляет собой *спецификацию действия* на поиск или синтез *знания*, удовлетворяющего информационную потребность *пользователя*, инициирующего этот *вопрос*. То есть *вопрос* — это ничто иное как *задача*, с помощью которой выражается потребность пользователя в некоторой информации, возможно хранимой или выводимой в *базе знаний ostis-системы* (см. *Главу 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*).

Каждому *вопросу* можно взаимно однозначно сопоставить некоторое множество *ответов на* этот *вопрос*. Каждый *ответ на вопрос* представляет собой некоторую *sc-структуру семантической окрестности основного знака*, раскрываемого в этом *ответе на* заданный *вопрос*.

вопрос
- ≔ [запрос]
- ≔ [непроцедурная формулировка задачи на поиск (в текущем состоянии базы знаний) или на синтез знания, удовлетворяющего заданным требованиям]
- ≔ [запрос метода (способа) решения заданного (указываемого) *класса задач* либо *плана решения* конкретной указываемой *задачи*]
- ≔ [задача, направленная на удовлетворение информационной потребности некоторого субъекта-заказчика]
- ⊂ *задача*

ответ на вопрос
- ≔ [ответ на запрос]
- ≔ [результат запроса]
- ≔ [результат решения задачи на поиск или синтез знания, удовлетворяющий заданным требованиям]
- ≔ [семантическая окрестность *основного знака*, знание в которой удовлетворяет информационную потребность пользователя]
- ≔ [знание в базе знаний ostis-системы, которое удовлетворяет информационную потребность пользователя]
- ⊂ *знание*

Среди всех классов *знаков в рамках заданного вопроса Языка вопросов для ostis-систем* можно выделить наиболее общие по иерархии классы *знаков*:

знак в рамках заданного вопроса
- ⊂ *знак*
- ⇒ *разбиение**:
  - {• *основной знак в рамках заданного вопроса*
    - ≔ [ключевой sc-элемент в рамках заданного вопроса]
    - ≔ [*знак*, относительно которого задан вопрос]
  - • *неосновной знак в рамках заданного вопроса*
    - ≔ [*знак*, стоящий в некотором отношении с *основным знаком в рамках заданного вопроса*]
  - }

знаком в рамках заданного вопроса является любой *знак понятия* или *сущности*, принадлежащий этому *вопросу*. Между *знаками в рамках заданного вопроса* задается множество связей *отношений*, входящих в состав различных *предметных областей*. Кроме того, любое ***отношение в рамках заданного вопроса*** представляет собой *отношение* между *знаками предметной области*, принадлежащих заданному *вопросу*. Среди всех классов *отношений в рамках заданного вопроса* можно выделить класс ***базовых отношений в рамках заданного вопроса*** и класс ***составных отношений в рамках заданного вопроса***.

отношение в рамках заданного вопроса
- ≔ [определенное отношение между знаками *предметной области* в контексте *вопроса*]
- ⊂ *отношение*

базовое отношение в рамках заданного вопроса
- ≔ [*класс отношений*, объединяющий *отношения в заданном вопросе*, отражающие однотипный *смысл* и раскрывающие определенный признак *знаков предметной области*]
- ⊂ *отношение в рамках заданного вопроса*
- ⇒ *декомпозиция**:
  - {• *отношение состояния*
  - • *отношение действия*
  - • *отношение состава*
  - • *теоретико-множественное отношение*
  - • *темпоральное отношение*

- *пространственное отношение*
- *количественное отношение*
- *качественное отношение*

}

Например, *отношения в рамках заданного вопроса* такие, как “играет*”, “спит*”, “плавает*”, объединяются в класс *отношений состояния* по признаку выражать состояние знака (то есть данные отношения раскрывают признак *знака предметной области* — “находиться в некотором состоянии”).

составное отношение в рамках заданного вопроса

:= [устойчивая комбинация двух *отношений действия*: действия, направленного на *параметр вопроса'*, и действия, направленного на *ответ на вопрос**]

Например, элемент *составного отношения в рамках заданного вопроса* между *знаками*: “*Нефтеперерабатывающий завод*”, “*нефть*” и “*нефтепродукты*” — может быть представлен как “Нефтеперерабатывающий завод перерабатывает нефть в нефтепродукты”.

Смысловая классификация *вопросов* дает возможность противопоставить каждому типу вопроса ограниченный набор допустимых, то есть *семантически корректных информационных конструкций*, передающий правильный *смысл вопроса* в зависимости от класса *вопроса*. При этом ***Семантическая классификация вопросов*** позволяет разбить множество *вопросов* на классы, в каждом из которых требуется раскрытие некоторого однотипного *смысла*, заданного классом этого *вопроса*.

вопрос

⇒ *декомпозиция**:

{• *вопрос, требующий вывода семантической окрестности основного знака*

∋ *пример'*:

- *Вопрос. Что такое Город Минск*

- *вопрос, требующий раскрытия в ответе базового отношения основного знака*

∋ *пример'*:

- *Вопрос. Что легче: железо или дерево*

- *вопрос, требующий раскрытия в ответе составного отношения основного знака*

⇒ *пояснение**:

[Такому классу *вопросов* соответствуют классы *ответов*, в которых *главный знак* раскрывается через *составное отношение*.]

∋ *пример'*:

- *Вопрос. Какие Принципы компонентного проектирования в интеллектуальных компьютерных системах нового поколения*

- *вопрос, требующий раскрытия в ответе произвольной комбинации базового отношения и/или составного отношения основного знака*

∋ *пример'*:

- *Вопрос. Как определяется уровень интеллекта кибернетической системы*

- *вопрос, требующий раскрытия в ответе более чем одного основного знака*

∋ *пример'*:

- *Вопрос. Докажите теорему Пифагора*

}

вопрос, требующий раскрытия в ответе базового отношения основного знака

⇒ *декомпозиция**:

{• *вопрос, требующий раскрытия в ответе отношения состава основного знака*

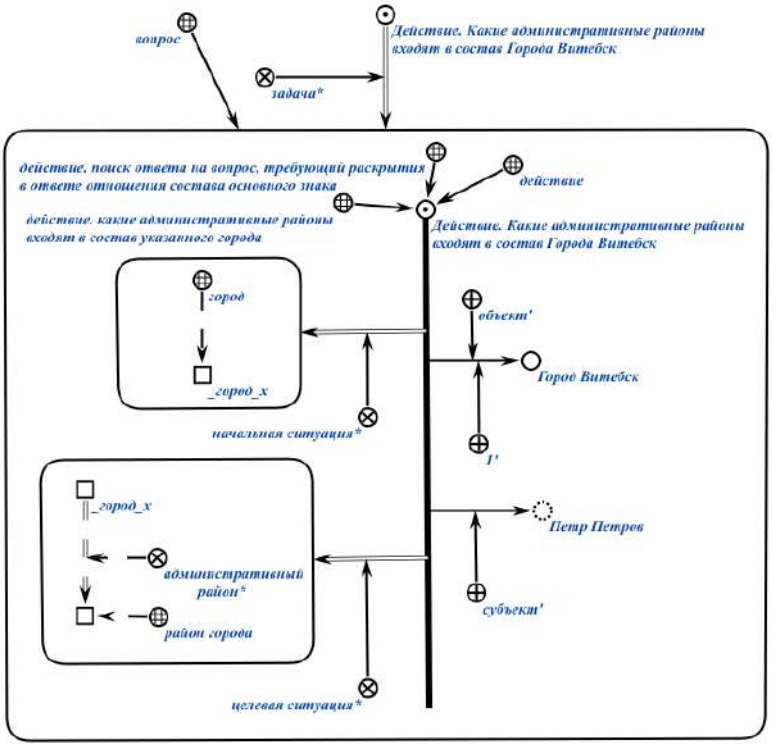
:= [класс вопросов, в ответах на которые *основной знак S* раскрывается через его *отношение состава* в связке с его составляющими знаками *P* и *Q*]

∋ *пример'*:

- *Вопрос. Какие административные районы входят в состав Города Витебск*

=

[



Вопрос. Какие административный районы входят в состав Города Витебск

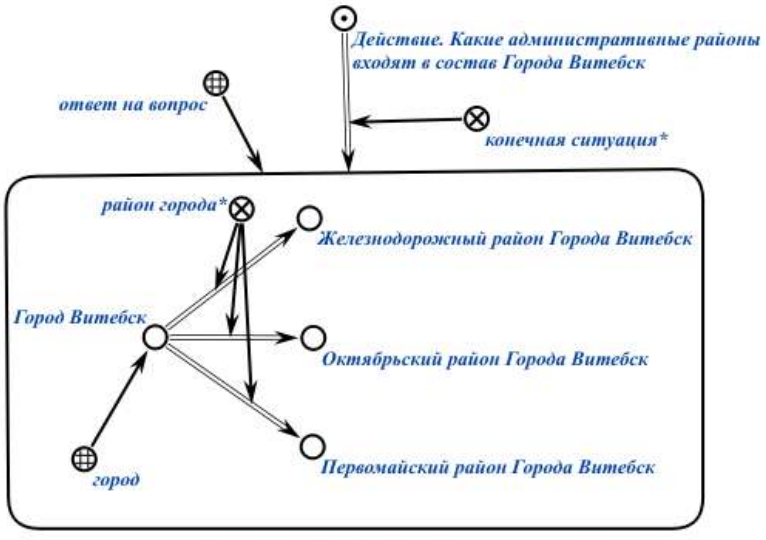
]

⇒ *ответ на вопрос**:

{Железнодорожный район Города Витебск, Октябрьский район Города Витебск, Первомайский район Города Витебск}

=

[



{Железнодорожный район Города Витебск, Октябрьский район Города Витебск, Первомайский район Города Витебск}

]

- *вопрос, требующий раскрытия в ответе теоретико-множественного отношения основного знака*

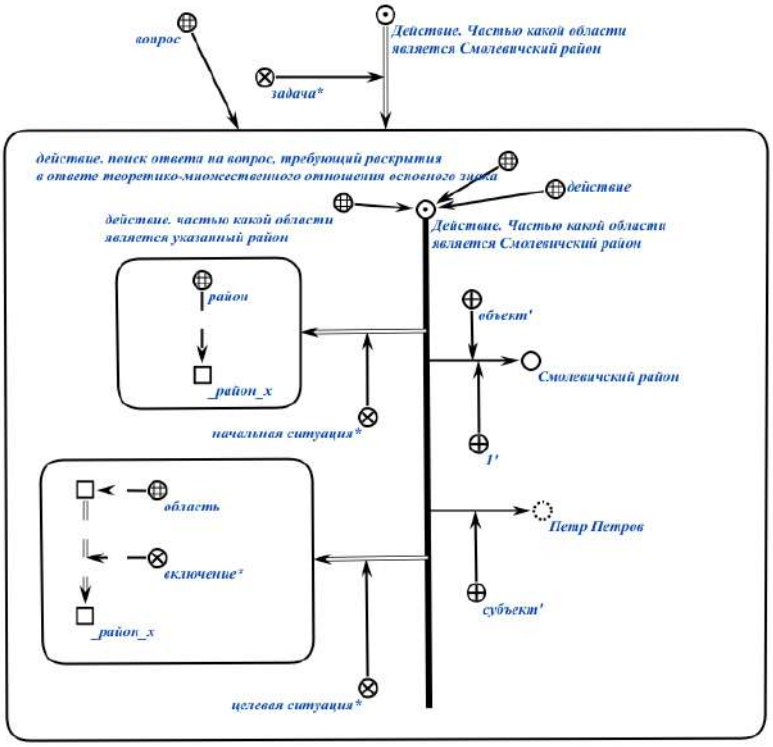
  ≔ [класс вопросов, в ответах на которые *основной знак S* раскрывается через его *теоретико-множественное отношение* в связке с другим знаком *P*, содержащего *S* как часть]

  ∋ *пример'*:

  - *Вопрос. Частью какой области является Смолевичский район*

    =

    [



Вопрос. Частью какой области является Смолевичский район

]

⇒ *ответ на вопрос**:

{Смолевичский район является частью Минской области}

- *вопрос, требующий раскрытия в ответе отношения состояния основного знака*

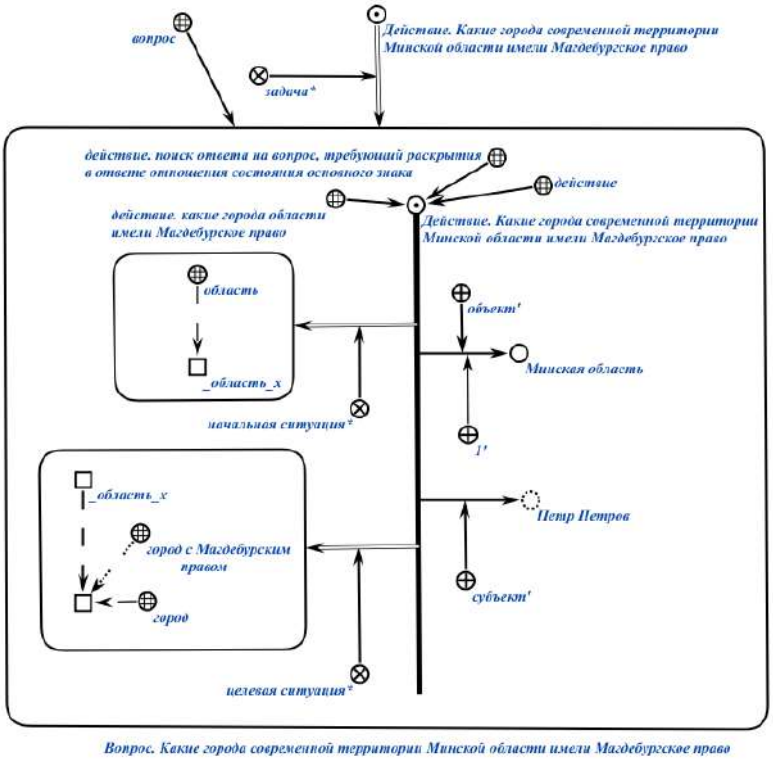
  ≔ [класс вопросов, в ответах на которые *основной знак S* раскрывается через его *отношение состояния*]

  ∋ *пример'*:

  - *Вопрос. Какие города современной территории Республики Беларусь имели Магдебургское право*

    =

    [



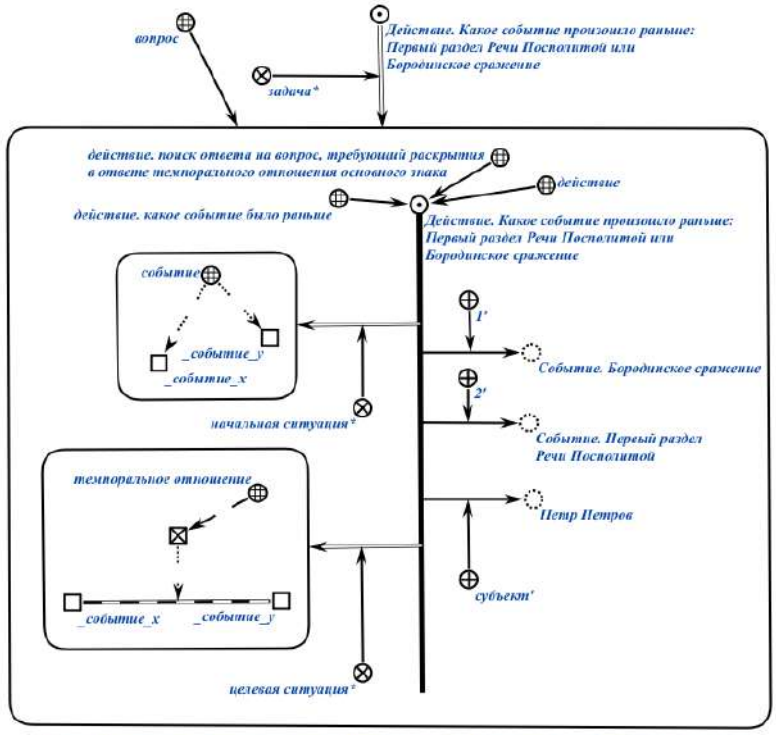
Вопрос. Какие города современной территории Минской области имели Магдебургское право

]

⇒ *ответ на вопрос**:

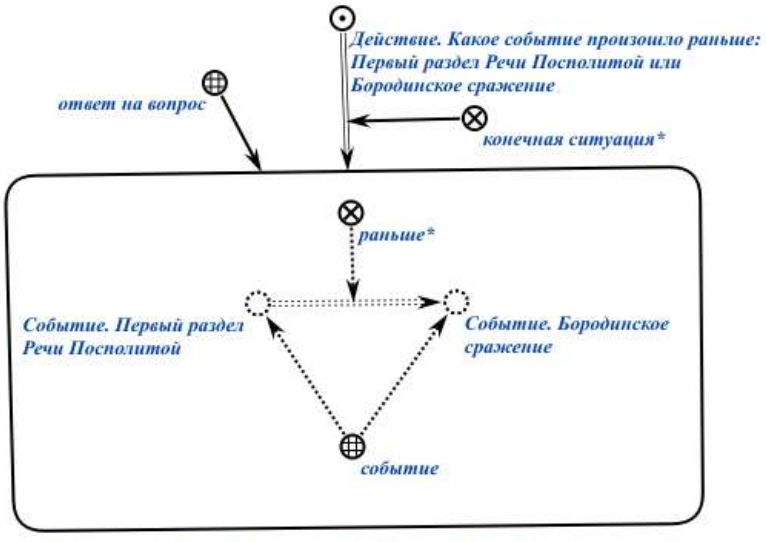
{Волковыск, Гродно, Мозырь и другие имели Магдебургское право}

- *вопрос, требующий раскрытия в ответе отношения действия основного знака*
  ≔ [класс вопросов, в ответах на которые *основной знак S* раскрывается через его *отношение действия* в связке с другим знаком *P*]
- *вопрос, требующий раскрытия в ответе темпорального отношения основного знака*
  ≔ [класс вопросов, в ответах на которые *основной знак S* раскрывается через его *темпоральное отношение* в связке с другим знаком *P* по некоторой временной шкале]
  ∋ *пример'*:
  - *Вопрос. Какое событие произошло раньше: Первый раздел Речи Посполитой или Бородинское сражение*
    =
    [



Вопрос. Какое событие произошло раньше: Первый раздел Речи Посполитой или Бородинское сражение

    ]
    ⇒ *ответ на вопрос**:
    {Первый раздел Речи Посполитой был раньше Бородинского сражения}
    =
    [

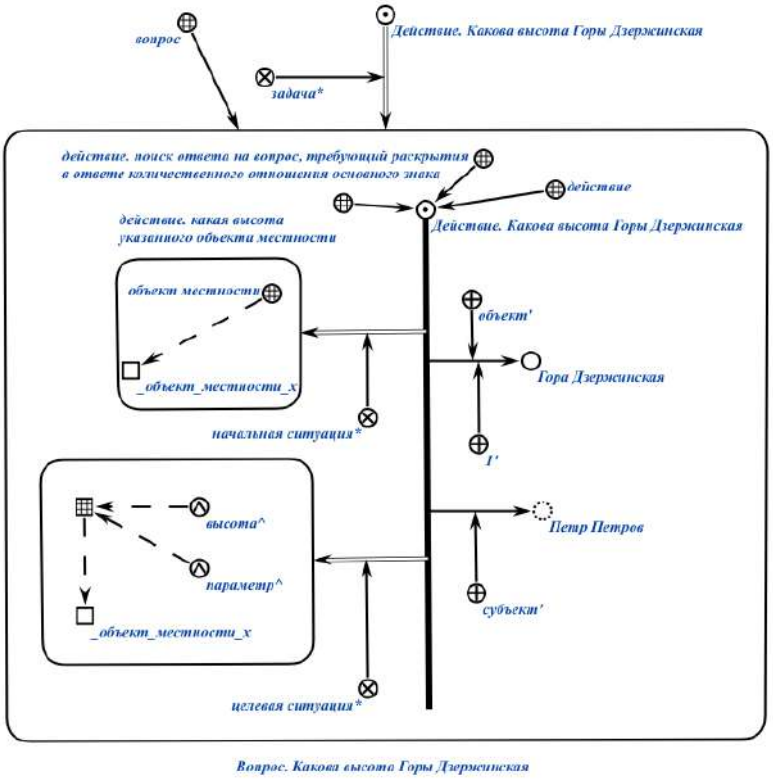


{Первый раздел Речи Посполитой был раньше Бородинского сражения}

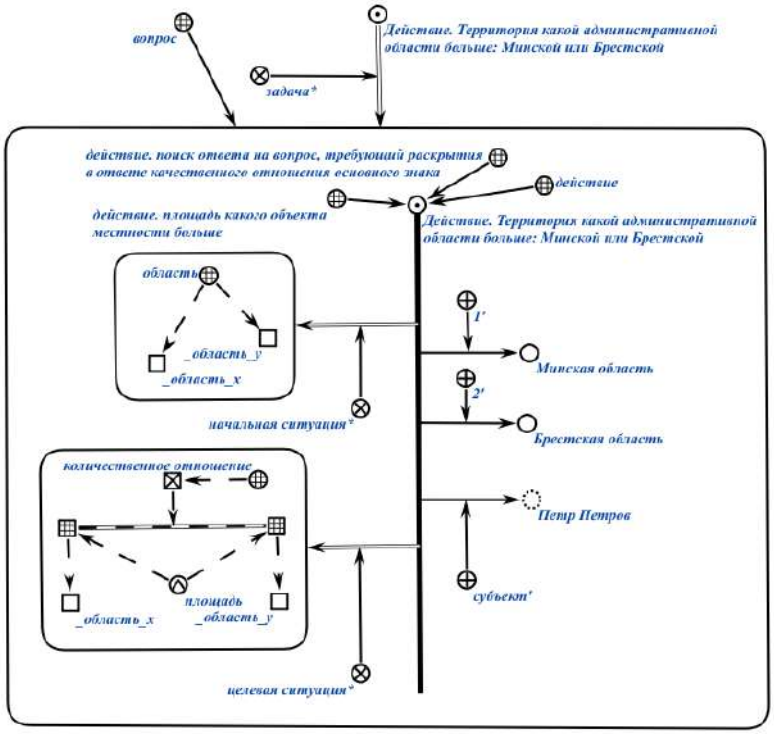
    ]
- *вопрос, требующий раскрытия в ответе пространственного отношения основного знака*

:= [класс вопросов, в ответах на которые *основной знак S* раскрывается через *пространственное отношение*, отражающее его положение в пространстве относительно другого знака *P*]

- *вопрос, требующий раскрытия в ответе количественного отношения основного знака*
  := [класс вопросов, в ответах на которые раскрывается *количественное отношение основного знака*]
  ∋ *пример'*:
  - *Вопрос. Какова высота Горы Дзержинская*
    =
    [



    ]
    ⇒ *ответ на вопрос**:
    {Высота Горы Дзержинская — 345 м}
- *вопрос, требующий раскрытия в ответе качественного отношения основного знака*
  := [класс вопросов, в ответах на которые раскрывается *качественное отношение основного знака S* в связке с другим знаком *P*]
  ∋ *пример'*:
  - *Вопрос. Территория какой административной области больше: Минской или Брестской*
    =
    [



Вопрос. Территория какой административной области больше: Минской или Брестской

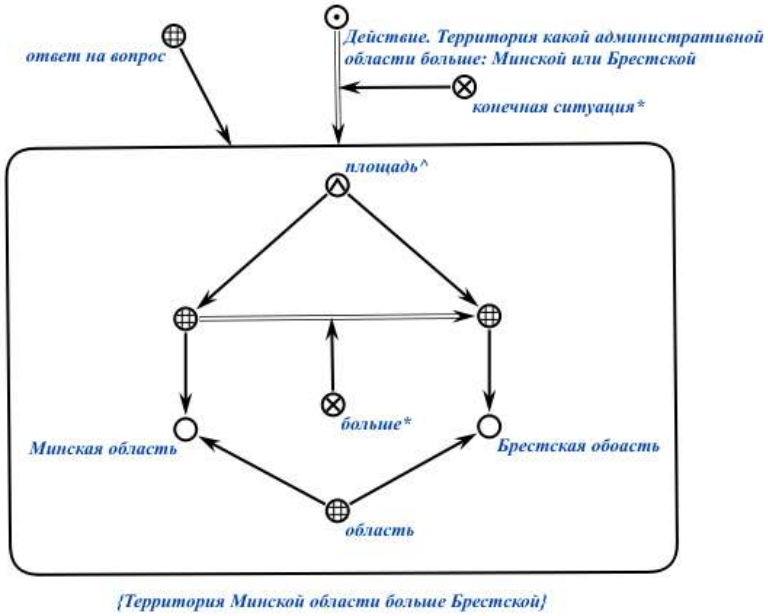
]

⇒ *ответ на вопрос**:

{Территория Минской области больше Брестской}

=

[



{Территория Минской области больше Брестской}

]

}

вопрос, требующий раскрытия в ответе произвольной комбинации базового отношения и/или составного отношения основного знака

⇒ *декомпозиция**:

{• *вопрос, требующий раскрытия в ответе произвольной комбинации составного отношения описания основного знака*

:= [класс вопросов, в ответах на которые раскрываются произвольные комбинации *базового отношения* и/или *составного отношения основного знака S* в связке с другими знаками]

∋ *пример'*:

- *{S состоит из P, Q, W. S переводит X и Y и выполняется раньше Z}*

⇐ *ответ на вопрос**:
Вопрос. Что такое S

- *вопрос, требующий раскрытия в ответе произвольной комбинации составного отношения определения основного знака*
  ≔ [класс ответов, в которых *основной знак S* раскрывается через *первостепенное понятие* и его *описание*]
  ∋ *пример'*:
  - *{Минск — это столица, которая находится в РБ}*
    ⇐ *ответ на вопросы**:
    Вопрос. Как определяется город Минск
- *вопрос, требующий раскрытия в ответе произвольной комбинации составного отношения причины основного знака*
  ≔ [класс вопросов, в ответах на которые раскрывается условие существования некоторых отношений *основного знака S* в связке с другими знаками]
  ∋ *пример'*:
  - *Вопрос. Почему время в пути от города Минска до города Борисова меньше чем время в пути от города Минска до города Орша*
    ⇒ *ответ на вопрос**:
    {Время в пути от города Минска до города Борисова меньше чем время в пути от города Минска до города Орша, потому что расстояние от города Минска меньше до города Борисова, чем до города Орша}
- *вопрос, требующий раскрытия в ответе произвольной комбинации составного отношения следствия основного знака*

≔ [класс вопросов, в ответах на которые раскрывается следствие от существования некоторых отношений *основного знака S* в связке с другими знаками]
∋ *пример'*:
- *Вопрос. Что следует из того, что расстояние от города Минска до города Борисова меньше расстояния от города Минска до города Орша*
  ⇒ *ответ на вопрос**:
  {Расстояние от города Минска до города Борисова меньше расстояния от города Минска до города Орша, поэтому от города Минска до города Борисова время в пути меньше чем до города Орша}

}

вопрос, требующий раскрытия в ответе более чем одного основного знака

⊃ *вопрос, требующий раскрытия в ответе отношение детализации знаков, стоящих в некоторых отношениях с основным знаком*
≔ [класс вопросов, в ответах на которые происходит детализация знаков, стоящих в некоторых отношениях с *основным знаком S*]
∋ *пример'*:
- *Вопрос. Какая связь водной сети существует между городом Минск и городом Светлогорск*
  ⇒ *ответ на вопрос**:
  {Город Минск расположен на реке Свислочь, которая впадает в реку Березина, протекающую через город Светлогорск}

Таким образом, для каждого *вопроса пользователя ostis-системы* можно найти класс *вопросов*, на котором можно реализовывать *вывод ответов* на этот *вопрос*. Описанная *Семантическая классификация вопросов* позволяет:
- автоматически структурировать *вопросы пользователей* по описанию этих *вопросов*;
- а также формировать *ответы на* эти *вопросы* с учетом *непроцедурных формулировок* этих *вопросов*.

§ 3.4.3. Операционная семантика языка вопросов для ostis-систем

⇒ *ключевое понятие**:
- *вопрос*
- *ответ на вопрос*
- *знак в рамках заданного вопроса*
- *основной знак в рамках заданного вопроса*
- *неосновной знак в рамках заданного вопроса*
- *отношение в рамках заданного вопроса*
- *базовое отношение в рамках заданного вопроса*

Каждому классу *вопросов* должен соответствовать определенный *коллектив sc-агентов*, реализующих поиск или синтез из *базы знаний ostis-системы* соответствующих ответов на поставленные *вопросы*. Следует отметить, что в зависимости от степени наполненности *базы знаний ответы* могут содержаться в *базе знаний* либо отсутствовать в текущей версии *базы знаний*. В случае наличия в *базе знаний ответа на* поставленный *вопрос* информационная потребность пользователя реализуется *информационно-поисковыми sc-агентами*, в противном случае — в зависимости от *классов вопросов* формирование ответов осуществляется специализированными *sc-агентами*, которые в процессе работы дополнительно выполняют вычислительные задачи либо осуществляют синтез на основе *логического вывода* (см. *Главу 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*) или других *моделей решения задач*.

интерпретатор Языка вопросов для ostis-систем
- ∈ *неатомарный sc-агент*
- ⇒ *декомпозиция абстрактного sc-агента**:
  - {• *Абстрактный sc-агент поиска ответа на заданный вопрос*
    - ⇒ *декомпозиция абстрактного sc-агента**:
      - {• *Абстрактный sc-агент поиска семантической окрестности основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения состава для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе теоретико-множественного отношения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения состояния для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения действия для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе темпорального отношения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе пространственного отношения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе количественного отношения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе качественного отношения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения описания для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения определения для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения причины для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения следствия для основного знака*
        - *Абстрактный sc-агент поиска ответа на вопрос, требующий раскрытия в ответе отношения детализации для основного знака*
      - }
    - *Абстрактный sc-агент синтеза ответа на заданный вопрос*
  - }

Все *sc-агенты*, выводящие *ответы на* поставленные *вопросы*, формируют *коллектив sc-агентов* — ***интерпретатор Языка вопросов для ostis-систем***, с помощью которого можно интерпретировать любые классы *вопросов*. *интерпретатор Языка вопросов для ostis-систем* может быть реализован по-разному: в виде *коллектива scp-агентов* или *платформенно-зависимых sc-агентов*.

Заключение к Главе 3.4.

Перечислим основные положения данной главы:
- информационная потребность *пользователей ostis-системы* может быть выражена в виде *вопросов*, а удовлетворение этой информационной потребности — в виде *ответов на* заданные *вопросы*;
- вывод *ответов на* заданные *вопросы пользователем ostis-системы* может быть осуществлен путем поиска *знаний* в текущем состоянии *базы знаний* этой *ostis-системы*, либо синтеза новых знаний, отсутствующих в *базе знаний* этой *ostis-системы*;

- каждый *вопрос* может быть представлен в виде некоторой *спецификации задачи*, инициированной *пользователем ostis-системы* для удовлетворения своей информационной потребности, а *ответ на* этот *вопрос* — в виде *семантической окрестности основного знака в рамках заданного вопроса*;
- каждому *вопросу* может быть сопоставлен соответствующий класс *вопросов* в *Семантической классификации вопросов*;
- для синтеза отсутствующих *ответов на* поставленные *вопросы* могут быть использованы различные *модели решения задач*, в том числе *логические модели решения задач*;
- *ответы на* поставленные *вопросы* могут быть транслированы в *естественно-языковой текст* и визуализированы при помощи соответствующих *естественно-языковых интерфейсов* для удобства выдачи информации любому пользователю.

Глава 3.5.

Логические, продукционные и функциональные модели решения задач в ostis-системах

⇒ *автор**:
- *Ивашенко В. П.*
- *Шункевич Д. В.*
- *Василевская А. П.*
- *Орлов М. К.*

⇒ *аннотация**:
[Логические модели решения задач являются основой обработки знаний в *интеллектуальных системах*. В данной главе рассматривается интеграция различных *моделей решения задач*, в том числе принципы *логического вывода*, для решения задач на основе общей формальной модели.]

⇒ *подраздел**:
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

⇒ *библиографическая ссылка**:
- *Lawan A.. tSemanWRLEES-2019art*
- *Голенков В.В..БазовПТЯS-1996кн*
- *Averin A.I..UsingPiDI-2004art*
- *Голенков В.В..ГрафоАМиСП-2004ст*
- *Sethy S.S.MediaIS-2021art*
- *Norton J.D.aDemon otIoCoII-2019art*
- *Yuxuan Z..MissiEAKGIItDGLaT-2022art*
- *Safawi A.R..tDecisPoAI-2015art*
- *Gungov A.tAmpliLiDtAoA-2018art*
- *Geramian A..FuzzyISAfF-2017art*
- *Son L.H..PictuISaNF-2017art*
- *Uehara ..FuzzyIIPaP-2017art*
- *Lupea M.aTheorPfCaRDL-2002art*
- *Weydert E.DefauLaPaTP-2022art*
- *Chen G..TempoLIfFDoS-2021art*
- *Рыбаков В.В.МультВНЛЛ-2020ст*
- *Orlov M.K..NonprPSMiN-2022art*
- *Гаврилова Т.А..БазыЗИСУ-2000кн*
- *ИскусИММ-1990кн*
- *Brownston L..ProgrESiOPS-1985bk*
- *Clips-2015эл*
- *Forgy C..ReteaFAftMPMOPMP-2019art*
- *Владимиров А.Н..ПрогрКУПР-2010ст*

§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами

⇒ *ключевой знак**:
- *Язык SCL*
- *Абстрактная scl-машина*

⇒ *ключевое понятие**:
- *логический вывод*

⇒ *ключевое знание**:
- *Формализация правила вывода Modus ponens*
- *Формализация правила резолюции*

- *Спецификация агента прямого логического вывода*

Логика решает задачи доказательства высказываний, аргументации того или иного высказывания, задачу генерации и опровержения гипотез. Некоторые гипотезы могут быть опровергнуты, однако извлекая причины того, почему гипотеза опровергнута, можно изменить посылку гипотезы так, чтобы создать новую гипотезу, которая впоследствии может стать теоремой.

Современная логика изучает *формальные языки*, служащие для выражения логических рассуждений. *логический язык — формальный язык*, предназначенный для воспроизведения логических форм контекстов *естественного языка*, а также выражения логических законов и способов правильных рассуждений в логических теориях, строящихся в данном языке. Логика не изучает то, как были получены знания, она позволяет представлять знания, а также из существующих знаний вывести новые (то есть из имеющихся формул логики вывести новые формулы этой же логики), установить правильность рассуждений.

На данный момент реализовано много систем *логического вывода*, использующих известные правила прямого заключения и резолюции в различных видах логик, однако остаются актуальными проблема совместимости вышеописанных систем и проблема коллективного решения задач с использованием различных моделей решения задач.

Каждая модель решения задач задается языком, обеспечивающим представление в памяти кибернетической системы некоторого класса методов решения задач, и интерпретатором указанных методов, определяющим операционную семантику указанного языка. Следует рассмотреть языки, которыми может задаваться логическая модель решения задач. Такими языками являются *Rule Interchange Format* (RIF), Semantic Web Rule Language (SWRL), *SHACL Rules* и *Notation3 Rules*, которые используются в *Semantic Web* (см. *Lawan A.. tSemanWRLEES-2019art*). На *Рисунок. Запись правил на языке SWRL* представлен пример правил на *Языке SWRL.*

Рисунок. Запись правил на языке SWRL
=

Expression
hasParent(?x, ?y) ∧ hasSister(?y, ?z) → hasAunt(?x, ?z)
hasSibling(?x, ?y) ∧ Man(?y) → hasBrother(?x, ?y)
hasChild(?x, ?y) ∧ Woman(?x) → hasDaughter(?x, ?y)
hasParent(?x, ?y) ∧ Man(?y) → hasFather(?x, ?y)
hasParent(?x, ?y) ∧ Woman(?y) → hasMother(?x, ?y)
hasSibling(?x, ?y) ∧ hasSon(?y, ?z) → hasNephew(?x, ?z)
hasSibling(?x, ?y) ∧ hasDaughter(?y, ?z) → hasNiece(?x, ?z)
hasConsort(?y, ?z) ∧ hasParent(?x, ?y) → hasParent(?x, ?z)
hasChild(?x, ?y) ∧ hasChild(?z, ?y) ∧ differentFrom(?x, ?z) → hasSibling(?x, ?z)
hasSibling(?x, ?y) ∧ Woman(?y) → hasSister(?x, ?y)
hasChild(?x, ?y) ∧ Man(?x) → hasSon(?x, ?y)
hasParent(?x, ?y) ∧ hasBrother(?y, ?z) → hasUncle(?x, ?z)

Описанные языки не предусматривают возможность представления формул в различных видах логик, поэтому при помощи них невозможно решить описанные проблемы. Языки правил специально построены для вывода следствий. Синтаксис и семантика языков онтологий и языков правил довольно сильно отличаются, поэтому возникает вопрос, как их совмещать.

Пролог — язык и система логического программирования. *база знаний* системы *Пролог* содержит информацию в виде предикатов. В логическом программировании, реализованном в *Прологе*, используется только одно *правило вывода* — правило резолюции. Задача пролог-программы заключается в том, чтобы доказать, является ли заданное целевое высказывание следствием из имеющихся формул и, если является, то каким образом был получен такой вывод. Когда пользователь задает вопрос системе Пролог, система ищет соответствующие предикаты в базе знаний и, если они найдены, сравнивать их с заданными условиями. Система Пролог хорошо справляется с нетрудными задачами, однако ограничен лишь одним принципом логического вывода и не позволяет учитывать сложноструктурированные знания в различных видах логик.

Для решения указанных проблем предлагается *логический язык* на основе *SC-кода* и абстрактная машина логического вывода на основе предложенного языка.

Технология OSTIS позволяет интегрировать любые принципы логического вывода для решения задач в интеллектуальных системах на основе общей формальной модели. Для того, чтобы использовать какую-либо новую или существующую модель, необходимо привести ее к предлагаемому формализму, что позволит интегрировать и синхронизировать ее с уже имеющимися в соответствующей *библиотеке многократно используемых компонентов ostis-систем*. Формализм *SC-кода* позволяет описывать широкий спектр понятий и отношений между ними,

что делает его подходящим вариантом для реализации логического вывода в интеллектуальных компьютерных системах нового поколения. Кроме того, целесообразно воспользоваться принципом наследования, лежащим в основе иерархической структуризации баз знаний ostis-систем. Исходя из этого предлагается следующая иерархия интегрированных предметных областей:

Предметная область логических формул, высказываний и формальных теорий
⇒ *дочерняя предметная область**:
- *Предметная область логических языков*
- *Предметная область логического вывода*

Предметная область логических языков
⇒ *дочерняя предметная область**:
Предметная область языка логики высказываний
⇒ *дочерняя предметная область**:
Предметная область языка логики предикатов

Предметная область логических моделей решения задач
⇐ *дочерняя предметная область**:
- *Предметная область логических языков*
- *Предметная область логического вывода*

Наследование предметных областей позволяет использовать описанные логики и их компоненты при описании любых логик. Базовые понятия позволяют разработчикам интеллектуальной системы добавлять новые логики. Для реализации конкретной логической модели решения задач необходимо создать предметную область, которая будет дочерней по отношению к *Предметной области логических моделей решения задач* и предметной области некоторого *логического языка*, например, языка логики высказываний, языка логики предикатов, языка нечеткой логики и других.

Предметная область логических формул, высказываний и формальных теорий задает денотационную семантику логических формул, высказываний и формальных теорий и содержит формальную спецификацию понятий, необходимых для формирования логических формул и высказываний любых логик, в том числе традиционных, нечетких, правдоподобных, темпоральных, логик умолчания и любых других. Логические формулы и высказывания интерпретируется с помощью понятий, описанных в *Предметной области логических моделей решения задач*, включающую модель и реализацию абстрактных агентов, необходимых для решения логических задач. Эта предметная область включает в себя спецификацию таких понятий, как логический вывод, правила вывода, равносильные преобразования и аксиомные схемы.

Язык SCL — подъязык *SC-кода* для записи логических утверждений (см. *Голенков В.В..БазовПТЯS-1996кн*), он подробно рассматривается в *Главе 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*. Над высказываниями *Языка SCL* можно проводить *логический вывод*.

Выводом в формальной системе называется любая последовательность формул такая, что любая формула либо аксиома этой формальной системы, либо непосредственное следствие каких-либо предыдущих формул по одному из правил вывода. Идея выводимости центральна в логике: в любой формальной аксиоматической теории теорема — это формула, которая выводится из аксиом. Правильность умозаключений вводится и проверяется совершенно формально, без какой-либо связи с истинностью входящих в него посылок, то есть исключительно с точки зрения структуры рассуждения. С практической точки зрения самое важное свойство такой формальной правильности рассуждений заключается в следующем: если нам удалось доказать, пользуясь методами формальной логики, правильность рассуждения, и нам известно из опыта, что все используемые посылки истинны, то мы можем быть уверены в истинности заключения (см. *Averin A.I..UsingPiDI-2004art*). Истинность используемых посылок задается состоянием базы знаний.

Различные логические подходы позволяют проектировать *решатели задач* для *интеллектуальных систем* в разных предметных областях, учитывая их специфику. *решатель задач* каждой конкретной системы во многом зависит от назначения данной системы, множества решаемых задач, предметной области и других факторов (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*). Некоторые операции, необходимые в одной предметной области будут избыточными в другой. Например, в системе, решающей задачи по геометрии, химии и другим естественным наукам обоснованным будет использование дедуктивных методов вывода, поскольку решение задач в таких предметных областях основывается только на достоверных правилах. В системах же медицинской диагностики, к примеру, постоянно возникает ситуация, когда диагноз может быть поставлен только с некоторой долей уверенности и абсолютно достоверным ответ на поставленный вопрос быть не может. В связи с этим возникает необходимость использования различных *решателей задач* в различных системах, при этом их состав и возможности в конкретной системе определяется не только непосредственно разработчиком, а требует консультаций с экспертами в данной предметной области. Тем не менее основой для всех видов логик

является классическая логика и наиболее общие ее методы распространяются на другие логики с некоторыми модификациями, уточнениями и ограничениями (см. *Голенков В.В..ГрафоАМиСП-2004ст*).

Приведем краткую классификацию существующих логических методов решения задач:

- **Классический дедуктивный вывод.** Классический дедуктивный вывод является наиболее популярным при построении автоматических решателей задач, так как всегда дает достоверный результат. Дедуктивный вывод включает в себя прямой и обратный и логический вывод (принцип резолюции, процедуру Эрбрана и так далее) (см. *Averin A.I..UsingPiDI-2004art*), все виды силлогизмов (см. *Sethy S.S.MediaIS-2021art*) и так далее. Основной проблемой дедуктивного вывода является невозможность его использования в ряде случаев, когда отсутствуют достоверные знания.
- **Индуктивный вывод.** Индуктивный вывод предоставляет возможность в процессе решения использовать различные предположения, что делает его удобным для использования в слабоформализованных и трудноформализуемых предметных областях, например при построении систем медицинской диагностики. Подробно принципы индуктивного вывода рассмотрены в работах *Norton J.D.aDemon otIoCoII-2019art* , *Yuxuan Z..MissiEAKGIItDGLaT-2022art*.
- **Абдуктивный вывод.** Под абдуктивным выводом в искусственном интеллекте, как правило, понимается вывод наилучшего абдуктивного объяснения, то есть объяснения некоторого события, ставшего неожиданным для системы. Причем наилучшим считается такое объяснение, которое удовлетворяет специальным критериям, определяемым в зависимости от решаемой задачи и используемой формализации. Абдуктивный вывод подробно рассматривается в работах *Safawi A.R..tDecisPoAI-2015art*, *Gungov A.tAmpliLiDtAoA-2018art*.
- **Нечеткая логика.** Теория нечетких множеств и, соответственно, нечетких логик, также применяется в системах, связанных с трудноформализуемыми предметными областями (см. *Geramian A..FuzzyISAfF-2017art*, *Son L.H..PictuISaNF-2017art*). Здесь импликативные высказывания могут рассматриваться как "если истинна посылка", то с некоторой вероятностью (часто или редко) истинно заключение, в отличие от классической логики, где зачастую используются статические предметные области и выражение "часто или редко" не применимо (корректно использовать только наречие "всегда"). Подробнее теория нечетких логик рассматривается в работе *Uehara ..FuzzyIIPaP-2017art*.
- **Логика умолчаний.** Логика умолчаний применяется, в том числе, для того, чтобы оптимизировать процесс рассуждений, дополняя процесс достоверного вывода вероятностными предположениями в тех случаях, когда вероятность ошибки крайне мала. Подробнее логика умолчаний рассмотрена в статьях *Lupea M.aTheorPfCaRDL-2002art*, *Weydert E.DefauLaPaTP-2022art*.
- **Темпоральная логика.** Применение темпоральной логики является очень актуальным для нестатичных предметных областей, в которых истинность того или иного утверждения меняется со временем, что существенно влияет на ход решения какой-либо задачи (см. *Chen G..TempoLIfFDoS-2021art*, *Рыбаков В.В.МультВНЛЛ-2020ст*). Следует отметить, что используемый в данной работе язык представления знаний предоставляет все необходимые возможности для описания таких динамических предметных областей.

База знаний интеллектуальной системы включает в себя как модель фактографических знаний о предметной области, для которой предназначена система, так и модель знаний, включающая в себя логические формулы об этой предметной области (аксиомы, теоремы и правила вывода).

Абстрактная scl-машина является машиной логического вывода и относится к классу абстрактных sc-машин (см. *Голенков В.В..БазовПТЯS-1996кн*). Внутренним языком *scl-машины* является указанный выше графовый логический *Язык SCL*, ее операции соответствуют правилам *логического вывода*. Семейство специализированных абстрактных графодинамических машин обработки знаний является формальным уточнением операционной семантики указанных выше специализированных графовых языков представления знаний, каждому из которых соответствует одна или несколько абстрактных машин. Эти абстрактные машины соответствуют различным моделям решения задач, различным логикам, различным моделям правдоподобных рассуждений. Агент из семейства агентов логического вывода может представлять собой какое-либо правило вывода, которое можно применять для решения логической задачи. Кроме того, необходимы агенты для выполнения равносильных преобразований логической формулы (например, записать формулу эквиваленции как конъюнкцию двух дизъюнкций) и другие агенты, помогающие применять правила вывода на множестве формул языка логики (см. *Orlov M.K..NonprPSMiN-2022art*).

Абстрактная scl-машина

⇒ *декомпозиция абстрактного sc-агента**:

{• *Абстрактный sc-агент применения правила вывода*

• *Абстрактный sc-агент эквивалентных преобразований логической формулы*

• *Абстрактный sc-агент прямого логического вывода*

• *Абстрактный sc-агент обратного логического вывода*

}

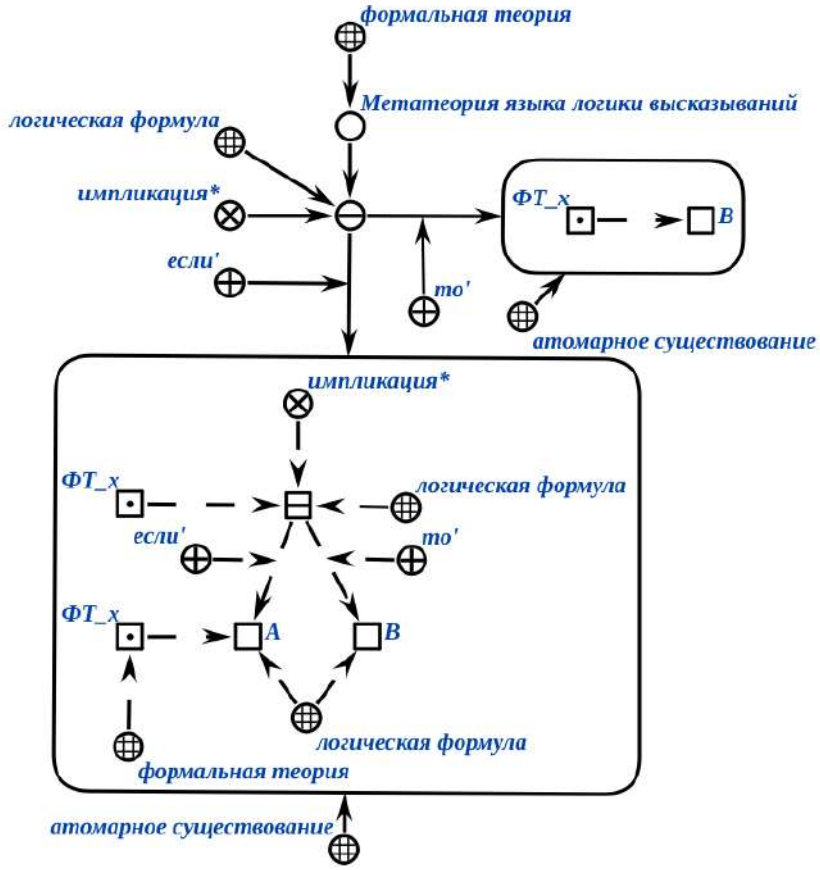
∋ *Реализации интерпретатора логических моделей решения задач*

:= [Реализация scl-машины]

⇒ *адрес компонента**:
[https://github.com/ostis-ai/scl-machine]

Задачей *Абстрактного sc-агента применения правила вывода* является применение заданного правила вывода с заданными логическими формулами. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие применение правила вывода*. После проверки sc-агентом условия инициирования выполняется процесс применения правила вывода, который заключается в проверке, существует ли в sc-памяти структуры, соответствующие условию применения данного правила и генерации sc-конструкций в соответствии с применяемым правилом. В ходе работы агента автоматически выполняется процедура унификации: переменные соответствуют константам, константы соответствуют самим себе. Агент применения правила вывода зачастую используется в процессе работы агентов прямого логического вывода, обратного логического вывода и других агентов. Примером правила вывода может быть правило прямого заключения (Modus ponens), представленное на рисунке *SCg-текст. Формализация правила вывода Modus ponens*.

SCg-текст. Формализация правила вывода Modus ponens
=

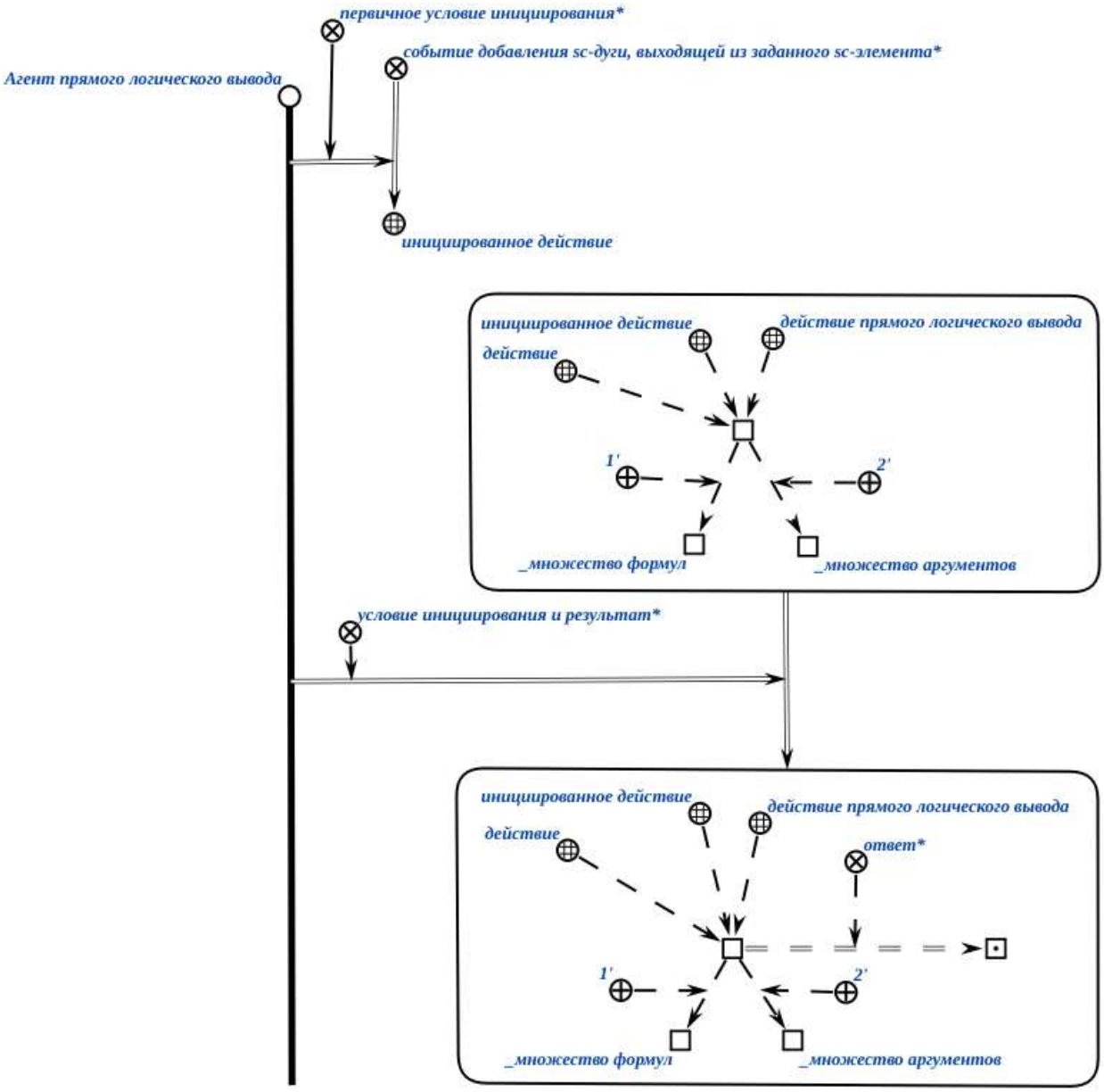


Можно привести еще целый ряд высказываний, которые описывают общие свойства всевозможных формальных теорий, каждая из которых описывает ту или иную предметную область. Свойства всевозможных формальных теорий описываются в рамках специальной метатеории для которой совокупность всевозможных формальных теорий является описываемой предметной областью.

Задачей *Абстрактного sc-агента эквивалентных преобразований логической формулы* является применение некоторых правил, которые приводят логическую формулу в определенный вид. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие эквиалентное преобразование логической формулы*. После проверки sc-агентом условия инициирования выполняется процесс преобразования формулы из одной формы в другую, при этом никакие новые знания в sc-памяти с точки зрения исследуемой предметной области не генерируются. Ответом данного агента является множество формул, эквивалентных по смыслу, но различных по форме представления. Такими формами могут быть, например, конъюнктивная нормальная форма или дизъюнктивная нормальная форма. Агент эквивалентных преобразований зачастую вызывается в процессе работы агента применения правила вывода, так как логические формулы не всегда находятся в той форме, которая доступна для применения того или иного правила вывода, однако может быть приведена к нужной форме.

Задачей *Абстрактного sc-агента прямого логического вывода* является генерации новых знаний на основе некоторых логических утверждений. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие прямого логического вывода*. После проверки sc-агентом условия инициирования выполняется процесс прямого логического вывода, который состоит из циклических операций применения правил вывода, генерации новых знаний в sc-памяти и проверки некоторого условия, например, появление в памяти sc-элементов из целевой sc-структуры (см. *Гаврилова Т.А..БазыЗИСУ-2000кн*). Входными аргументами такого агента является целевая структура, множество формул, которые используются в ходе вывода агентом применения правил вывода, множество правил вывода, входная структура и выходная структура. В результате выполнения агентом логического вывода действия, в sc-памяти формируется sc-структура, представляющая собой дерево решения. Это дерево состоит из последовательности узлов, представляющих собой примененные правила, которые привели к появлению в sc-памяти требуемых знаний. Такое дерево может быть пустым в случае, если требуемую структуру не удалось сгенерировать в ходе логического вывода. На рисунке *SCg-текст. Спецификация агента прямого логического вывода* приведен пример спецификации агента прямого логического вывода.

SCg-текст. Спецификация агента прямого логического вывода
=



Задачей *Абстрактного sc-агента обратного логического вывода* является проверка гипотез. Некоторые гипотезы могут быть опровергнуты, однако извлекая причины того, почему гипотеза опровергнута, можно изменить посыл-

ку гипотезы так, чтобы создать новую гипотезу, которая впоследствии может стать полезной теоремой. Данный sc-агент активируется при появлении в sc-памяти инициированного действия, принадлежащего классу *действие обратного логического вывода*. После проверки *sc-агентом* условия инициирования выполняется процесс *обратного логического вывода*, который схож с процессом *прямого логического вывода* за исключением того, что поиск правил основывается не на посылках формул, а на их следствиях (см. *Гаврилова Т.А..БазыЗИСУ-2000кн*). Ответом данного агента будет также дерево вывода, которое показывает, с использованием каких правил можно доказать или опровергнуть выдвинутую гипотезу.

Абстрактный sc-агент эквивалентных преобразований логической формулы

⇒ *декомпозиция абстрактного sc-агента**:

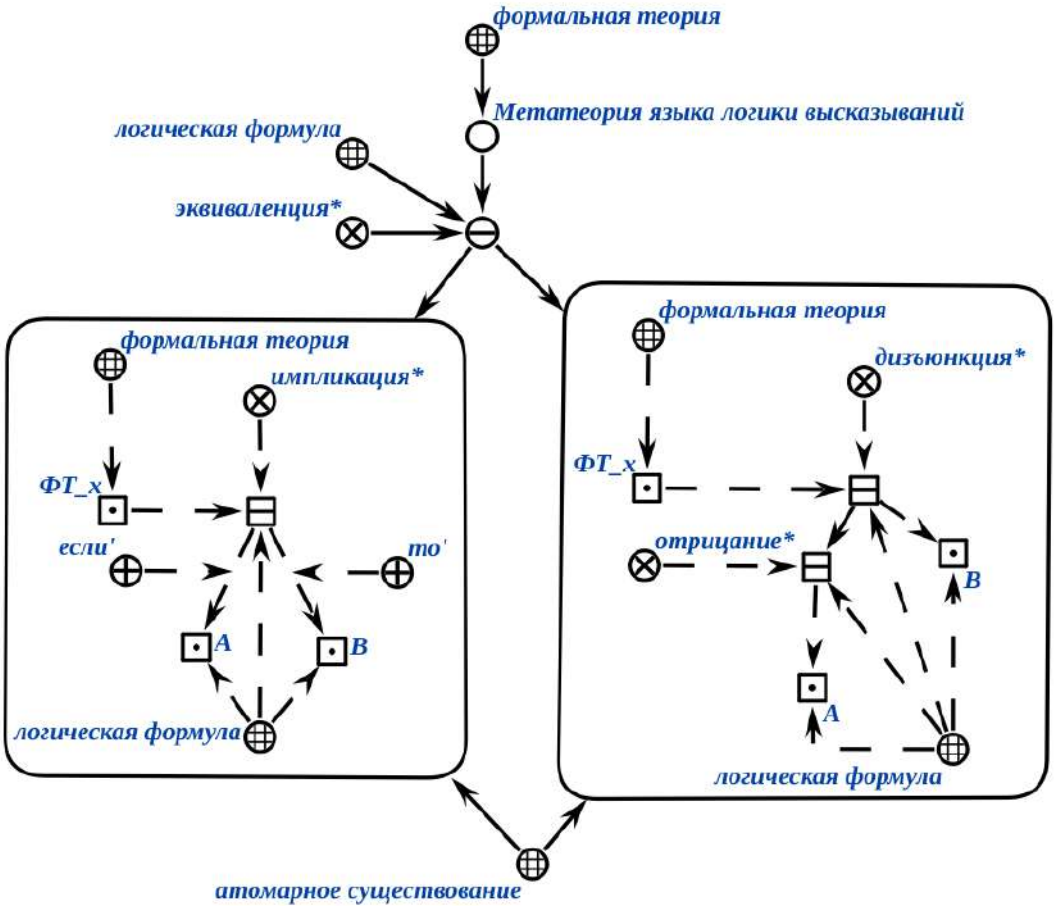
{
- *Абстрактный sc-агент преобразования формулы в конъюнктивную нормальную форму*
- *Абстрактный sc-агент преобразования формулы в дизъюнктивную нормальную форму*
- *Абстрактный sc-агент применения законов Де Моргана*
- *Абстрактный sc-агент эквивалентных преобразований логической формулы по определению*
- *Абстрактный sc-агент применения свойств отрицания логических формул*
- *Абстрактный sc-агент применения закона идемпотентности логических формул*
- *Абстрактный sc-агент применения закона коммутативности логических формул*
- *Абстрактный sc-агент применения закона ассоциативности логических формул*
- *Абстрактный sc-агент применения закона поглощения логических формул*
- *Абстрактный sc-агент применения закона противоречия логических формул*
- *Абстрактный sc-агент применения закона двойного отрицания логических формул*
- *Абстрактный sc-агент применения закона расщепления логических формул*

}

Любая формула семантически эквивалентна некоторой формуле в конъюнктивной нормальной форме, в связи с этим иногда удобно применять правило резолюции. Используя законы Де Моргана можно также получить формулы, пригодные для использования правила резолюции. С помощью правила резолюции можно эффективно доказывать формулы *Языка логики высказываний*.

Однако ничего принципиально нового правильно резолюции не привносит, поскольку формула $A \Rightarrow B$ равносильно $\neg A \vee B$ и из выводимости A и $A \rightarrow B$ следует выводимость B.

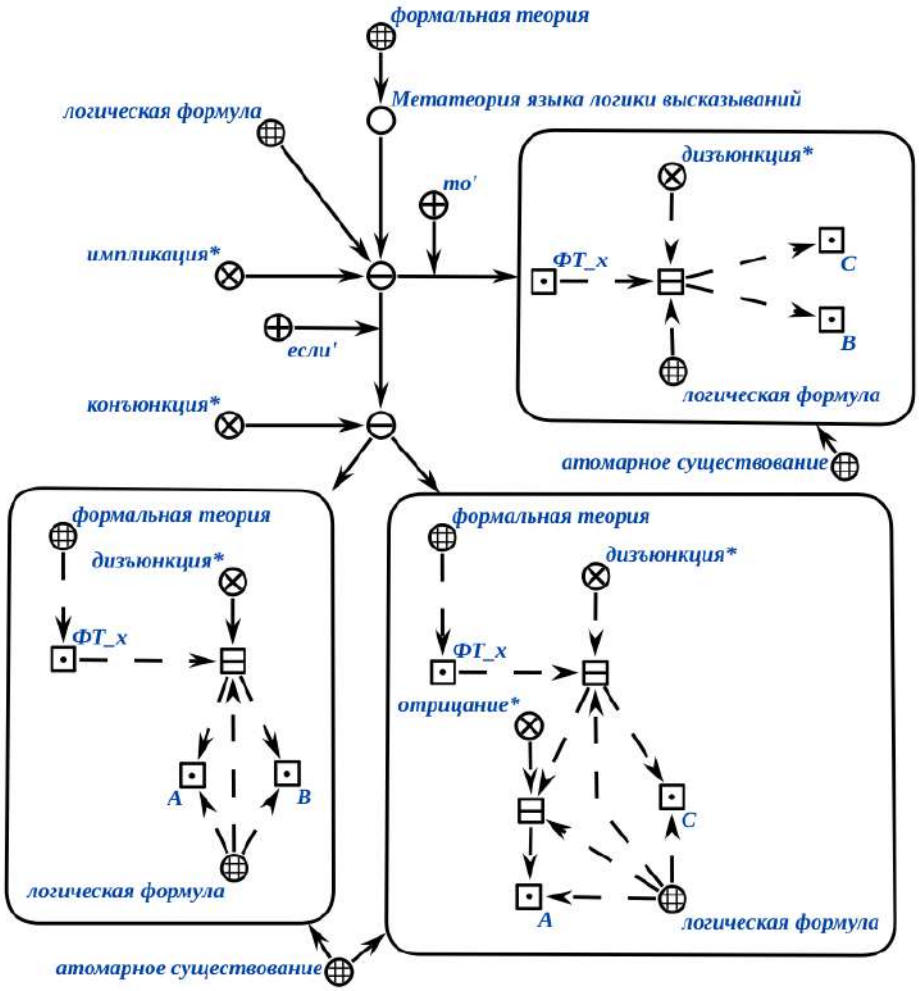
SCg-текст. Формализация конъюнктивной нормальной формы для импликации
=



Если в любых двух дизъюнктах C_1 и C_2 имеется пара формул A и $\neg A$, то можно сформировать новый дизъюнкт из оставшихся частей изначальных дизъюнктов.

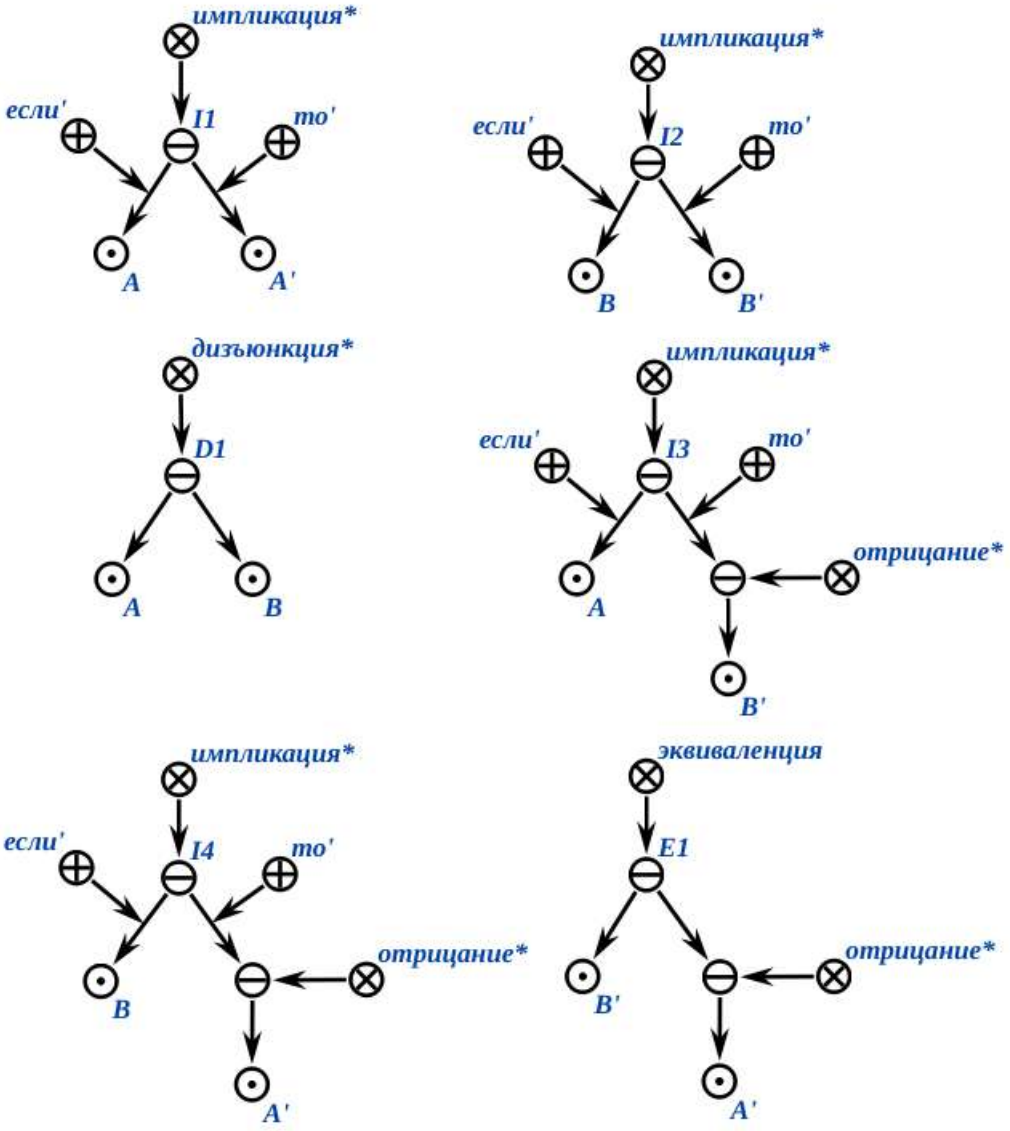
SCg-текст. Формализация правила резолюции

=



Приведем пример вывода формулы из множества посылок, используя правило резолюции. Если команда А выигрывает в футбол, то город А' торжествует, а если выигрывает команда В, то торжествовать будет город В'. Выиграть может или только город А', или только город В'. Однако, если выигрывает команда А, то город В' не торжествует, а если выигрывает команда В, то не торжествует город А'. Следовательно, город В' торжествует тогда и только тогда, когда не будет торжествовать город А'. Цель логического вывода - удостовериться, что город В' торжествует тогда и только тогда, когда не будет торжествовать город А'. Доказать вывода формулы равносильно доказательству противоречивости вывода отрицания этой формулы. При использовании правила резолюции это особенно удобно использовать. Формализация логических формул, соответствующих примеру приведена на рисунке *SCg-текст. Формализация правил для применения правила резолюции*. Каждая неатомарная формула на рисунке принадлежит некоторой формальной теории, то есть считается истинной.

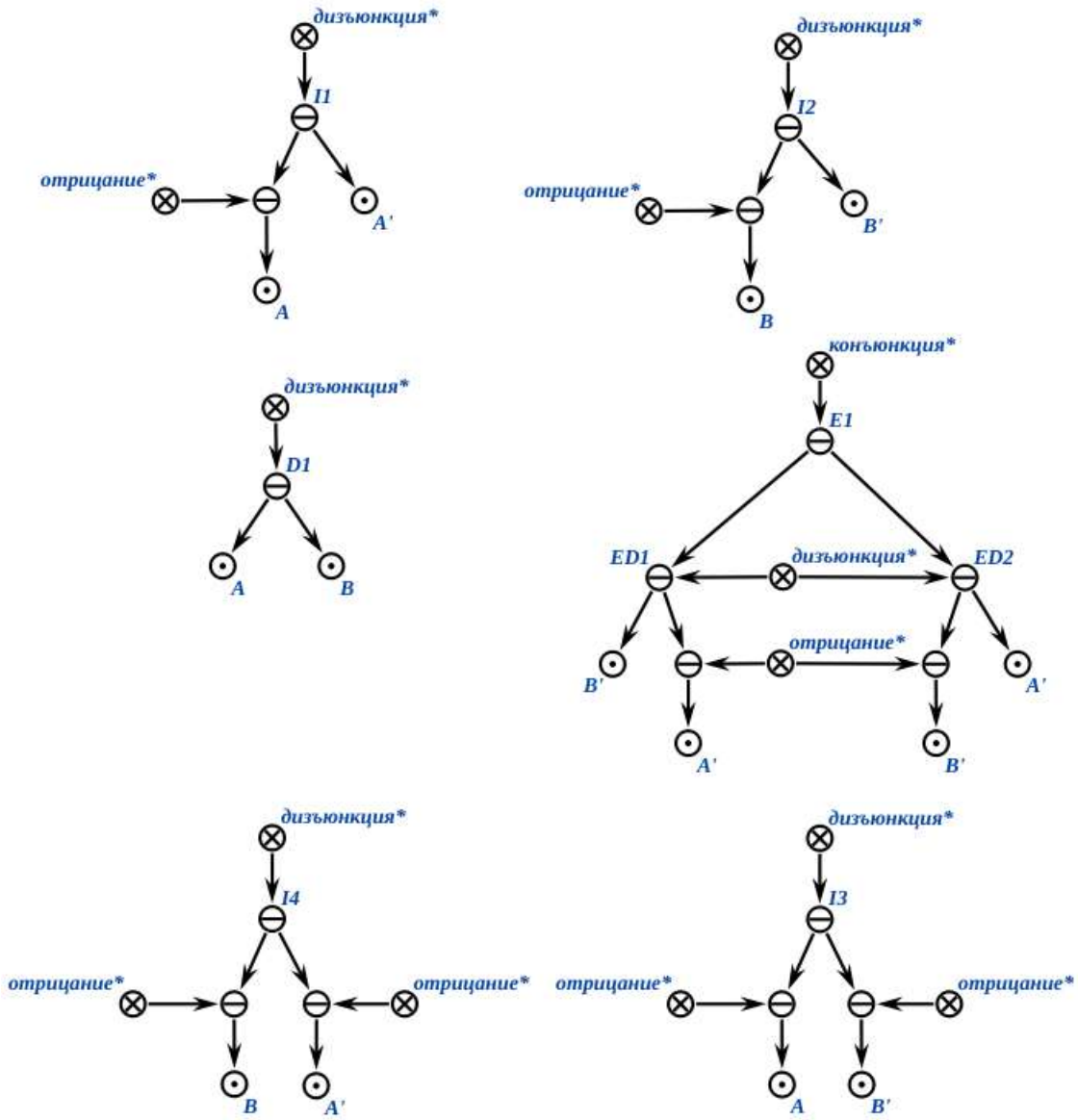
SCg-текст. Формализация правил для применения правила резолюции
=



Структура A представляет собой атомарную логическую формулу, которая обозначает победу команды A, структура A’ представляет формулу, обозначающую торжество города A’. Соответственно, то же самое для структур B и B’. Выразим логические формулы в булевом базисе. Прежде всего необходимо привести импликацию в конъюнктивную нормальную форму по формуле *SCg-текст. Формализация конъюнктивной нормальной формы для импликации* и эквиваленцию по определению. Затем применим отрицание к формуле, которую необходимо вывести (эвиваленция). В результате получим следующие формулы (рисунок *SCg-текст. Формализация правил для применения правила резолюции после преобразования в конъюнктивную нормальную форму*):

SCg-текст. Формализация правил для применения правила резолюции после преобразования в конъюнктивную нормальную форму

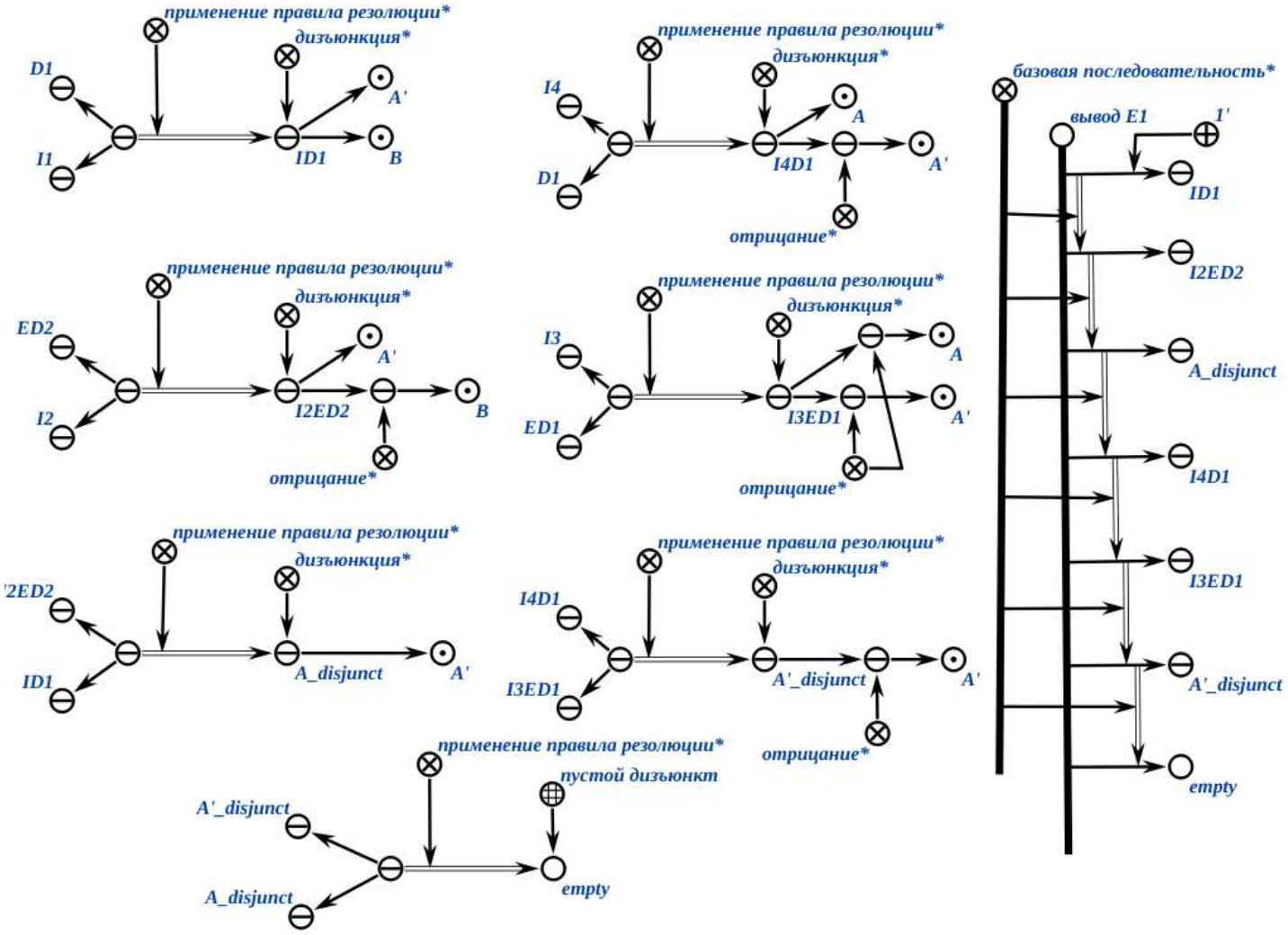
=



Далее применяя правило резолюции для преобразованных формул получаем пустой дизъюнкт, что говорит о противоречивости множества формул и доказывает формулу эквиваленции о том, что город B’ торжествует тогда и только тогда, когда не будет торжествовать город A’.

SCg-текст. Применение принципа резолюции

=



Таким образом, можно использовать различные правила вывода, различные агенты логического вывода в различных видах логик в зависимости от специфики предметной области, в которой решается задача. Каждая модель является совместимой в рамках общей формальной модели решателей задач ostis-систем.

§ 3.5.2. Языки продукционного программирования, используемые ostis-системами

⇒ *подраздел**:

- *Пункт 3.5.2.1. Синтаксис языков продукционного программирования, используемых ostis-системами*
- *Пункт 3.5.2.2. Денотационная семантика языков продукционного программирования, используемых ostis-системами*

Продукции наряду с фреймами являются наиболее популярными средствами представления знаний в интеллектуальных компьютерных системах. Продукции, с одной стороны, близки к логическим моделям, что позволяет организовывать на них эффективные процедуры вывода, а с другой стороны, более наглядно отражают знания, чем классические логические модели. В них отсутствуют жесткие ограничения, характерные для логических исчислений, что дает возможность изменять интерпретацию элементов продукции.

Так как в основе *Технологии OSTIS* лежит *многоагентный подход*, который позволяет легко интерпретировать любую информацию, то можно сделать вывод, что продукционный подход легко интегрируется в *решатели задач ostis-систем* в виде *sc-агентов*. Интеграция *Технологии OSTIS* и продукционного подхода позволяет объединить статические и динамические знания в рамках единого формализма, а также получить их графическое представление.

Пункт 3.5.2.1. Синтаксис языков продукционного программирования, используемых ostis-системами

В общем виде под продукцией понимается выражение следующего вида:

(i); Q; P; $A \Rightarrow B$; N.

Здесь i — имя продукции, с помощью которого данная продукция выделяется из всего множества продукций. В качестве имени может выступать некоторая лексема, отражающая суть данной продукции (например, "покупка книги" или "набор кода замка"), или порядковый номер продукции в их множестве, хранящемся в памяти системы.

Элемент Q характеризует сферу применения продукции или же контекст.

Основным элементом продукции является ее ядро: $A \Rightarrow B$. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции $\Rightarrow$. Обычное прочтение ядра продукции выглядит так: ЕСЛИ A, ТО B, более сложные конструкции ядра допускают в правой части альтернативный выбор, например, ЕСЛИ A, ТО B_1 ИНАЧЕ B_2. Секвенция может истолковываться в обычном логическом смысле как знак логического следования B из истинного A (если A не является истинным выражением, то о B ничего сказать нельзя). Возможны и другие интерпретации ядра продукции, например A описывает некоторое условие, необходимое для того, чтобы можно было совершить действие B.

Элемент P есть условие применимости ядра продукции. Обычно P представляет собой логическое выражение (как правило, предикат). Когда P принимает значение «истина», ядро продукции активизируется. Если P ложно, то ядро продукции не может быть использовано. Например, если в продукции «НАЛИЧИЕ ДЕНЕГ; ЕСЛИ ХОЧЕШЬ КУПИТЬ ВЕЩЬ X, ТО ЗАПЛАТИ В КАССУ ЕЕ СТОИМОСТЬ И ОТДАЙ ЧЕК ПРОДАВЦУ» условие применимости ядра продукции ложно, то есть денег нет, то применить ядро продукции невозможно.

Элемент N описывает постусловия продукции. Они актуализируются только в том случае, если ядро продукции реализовалось. Постусловия продукции описывают действия и процедуры, которые необходимо выполнить после реализации B. Например, после покупки некоторой вещи в магазине необходимо в описи товаров, имеющихся в этом магазине, уменьшить количество вещей такого типа на единицу. Выполнение N может происходить не сразу после реализации ядра продукции (см. *ИскусИММ-1990кн*).

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выбор для выполнения той или иной продукции из числа актуализированных.

Продукционный язык программирования в общем случае может содержать структурные единицы различного характера. Это могут быть определения глобальных переменных, операторы управления макрогенерацией, вставки текстов на алгоритмических языках программирования и целый ряд других элементов. Однако основной единицей продукционного языка программирования, определяющей его лицо и его возможности, является продукция.

В ряде интеллектуальных систем используются комбинации сетевых и продукционных моделей представления знаний. В таких моделях декларативные знания описываются в сетевом компоненте модели, а процедурные знания — в продукционном. В этом случае говорят о работе продукционной системы над семантической сетью. Процедурные знания позволяют системе узнать, как можно использовать те или иные декларативные знания, в частности, знания о закономерностях той части действительности, в которой "живет" интеллектуальная система, для получения нужных системе результатов или тех результатов, которые ожидает от нее пользователь.

Пункт 3.5.2.2. Денотационная семантика языков продукционного программирования, используемых ostis-системами

Одним из наиболее известных языков этого класса является ***OPS5***. *OPS5* можно рассматривать как полноценный язык программирования для продукционного программирования. База данных в языке называется рабочей памятью (working memory) и состоит из нескольких сотен объектов, каждый из которых имеет свой набор атрибутов. Объект вместе с парами <атрибут – значение> называется элементом рабочей памяти (см. *Brownston L..ProgrESiOPS-1985bk*).

Основная задача, которую поставили перед собой разработчики языка *OPS5*, добиться максимально высокой эффективности выполнения продукционной программы. Интерпретатор системы порождает конфликтное множество, каждый элемент которого представляет собой пару <имя продукции, список элементов рабочей памяти, которые являются означиваниями для образцов продукции>. Каждая продукция в *OPS5* состоит из символа P, имени продукции, левой части, символа –> и правой части.

В *OPS5* выделены три типа действий:

- MAKE — создает новый элемент рабочей памяти;

- MODIFY — изменяет один или несколько значений атрибутов у существующего элемента рабочей памяти;
- REMOVE — удаляет элемент рабочей памяти.

Рисунок. Пример правила продукции на OPS5
=

```
(p Holds::Object-Ceiling
  {(goal ^status active ^type holds ^objid <O1>) <goal>}
  {(physical-object
    ^id <O1>
    ^weight light
    ^at <p>
    ^on ceiling) <object-1>}
  {(physical-object ^id ladder ^at <p> ^on floor) <object-2>}
  {(monkey ^on ladder ^holds NIL) <monkey>}
  -(physical-object ^on <O1>)
-->
  (write (crlf) Grab <O1> (crlf))
  (modify <object1> ^on NIL)
  (modify <monkey> ^holds <O1>)
  (modify <goal> ^status satisfied)
)
```

В этом примере данные в рабочей памяти структурированы и переменные появляются в угловых скобках. Название структуры данных, такое как "goal" (цель) и "physical-object" (физический объект), является первым буквальным в условиях; поля структуры начинаются с "^". На негативное состояние указывает "–".

Продукционные правила в *OPS5* применяются ко всем продукциям структур данных, которые соответствуют условиям и соответствуют привязкам переменных. В этом примере, если несколько объектов подвешены к потолку, каждый с другой лестницей рядом, поддерживающей обезьяну с пустыми руками, конфликтный набор будет содержать столько же продукций правил продукции, полученных из одной и той же продукции "Holds::Object-Ceiling". На этапе разрешения конфликта позже будет выбрано, какие продукции запускать.

Связывание переменных, возникающее в результате сопоставления с шаблоном в левой части, используется в правой части для ссылки на данные, подлежащие модификации. Рабочая память содержит явные данные структуры управления в виде экземпляров «целевой» структуры данных. В этом примере, когда обезьяна держит подвешенный объект, статус цели устанавливается на «удовлетворено», и то же производственное правило больше не может применяться, поскольку его первое условие не выполняется.

CLIPS использует продукционную модель представления знаний и поэтому содержит три основных элемента:
- базу фактов (fact base),
- базу правил (rule base),
- механизм логического вывода.

База фактов представляет исходное описание задачи. База правил содержит операторы, которые преобразуют состояния проблемы, приводя его к решению — целевому состоянию (см. *Clips-2015эл*).

Механизм логического вывода *CLIPS* сопоставляет факты из базы фактов и правила из базы правил и выясняет, какие из правил можно активизировать. Это выполняется циклически, причем каждый цикл (так называемый продукционный цикл или цикл распознавания действия) состоит из трех основных фаз:
- сопоставление фактов и правил;
- выбор правила, подлежащего активизации;
- выполнение действий, предписанных активным («зажженным») правилом.

Факты — это одна из основных форм представления информации в системе *CLIPS*. Каждый факт представляет фрагмент информации, который был помещен в текущий список фактов, называемый fact-list. Факт представляет собой основную единицу данных, используемую правилами.

Если при добавлении нового факта к списку обнаруживается, что он полностью совпадает с одним из уже включенных в список фактов, то эта операция игнорируется.

Факт может описываться индексом или адресом. Всякий раз, когда факт добавляется (изменяется), ему присваивается уникальный целочисленный индекс. Факт также может задаваться при помощи адреса.

Идентификатор факта — это короткая запись для отображения факта на экране. Она состоит из символа f и записанного через тире индекса факта. Существует два формата представления фактов: позиционный и непозиционный. Позиционные факты состоят из выражения символьного типа, за которым следует последовательность (возможно, пустая) из полей, разделенных пробелами. Вся запись заключается в скобки. Обычно первое поле определяет "отношение", которое применяется к оставшимся полям.

Алгоритм Rete содержит обобщение логики функционала, ответственного за связь данных (фактов) и алгоритма (продукций) в системах сопоставления с образцом (вид систем: системы основанные на правилах). Продукция состоит из одного или нескольких условий и набора действий, выполняемых если актуальный набор фактов соответствует одному из условий. Условия накладываются на атрибуты фактов, включая их типы и идентификаторы. *Алгоритм Rete* имеет следующие характеристики:

- Уменьшает или исключает избыточность условий за счет объединения узлов.
- Сохраняет частичные соответствия между фактами при слиянии разных типов фактов. Это позволяет избежать полного вычисления всех фактов при любом изменении в рабочей памяти продукционной системы. Система работает только с самими изменениям.
- Позволяет эффективно высвобождать память при удалении фактов.

Алгоритм Rete широко используется для реализации сопоставления с образцом в системах с циклом сопоставление-решение-действие для генерации и логического вывода (см. *Forgy C..ReteaFAftMPMOPMP-2019art*).

Rete II улучшен по двум параметрам: повышена общая производительность сети включая хешированную память для больших массивов данных, добавлен алгоритм обратного вывода, работающий на той же сети. Скорость обратного вывода по сравнению с *Rete I* повышена значительно.

Rete-NT — новое поколение *алгоритма Rete*. Алгоритм был признан более быстрым, чем оригинальный *алгоритм Rete* и в 10 раз более быстрым, чем его предшественник *Rete II*.

Миварный подход объединяет и другие научные области компьютерных наук, информатики и дискретной математики, включая: базы данных, экспертные системы, системы логического вывода на основе развития продукций, теорию графов, матрицы, параллельное выполнение программ на кластерах, проектирование новых архитектур компьютеров, массовое суммирование чисел, техническую защиту информации и информационную безопасность, гносеологию (частично и в плане создания новой наиболее мощной модели данных на основе "тройки" "вещь-свойство-отношение"), сервисно-ориентированные архитектуры, компьютерные сети, информационные инфраструктуры, теоретическую робототехнику, многоагентные системы и некоторые другие (см. *Владимиров А.Н..ПрогрКУПР-2010cm*). *Миварный подход* объединяет две основные технологии накопления данных и обработки информации:

- миварное информационное пространство: накопление данных на основе эволюционной самоорганизующейся миварной модели данных с изменяющейся структурой в теории баз данных,
- миварные сети: обработка информации на основе развития продукционного подхода к логическому выводу с учетом включения возможности автоматического конструирования алгоритмов для "решателей задач" и традиционной вычислительной обработки, а также с использованием идей отношений, правил и процедур, которые теперь принято относить к сервисно-ориентированным архитектурам и многоагентным системам.

Суть *миварного подхода* в объединении баз данных и систем логико-вычислительной обработки в единые эволюционно развивающиеся системы, позволяющие собрать воедино все различные научные разработки на основе сервисно-ориентированных архитектур и технологий интеллектуальных агентов — многоагентных систем.

Миварные сети основаны на продукционном подходе "если, то. . . " с переходом к более сложной структуре правил с предусловиями, условиями, ограничениями, действиями и последействиями. Это позволяет записывать все причинно-следственные отношения, включая и все возможные формы предикатов и подобных логических выражений. Мы не отрицаем значение предикатов и поиска истинных выражений, а только создаем возможность и для их реализации, и для реализации всех возможных других представлений правил в виде: сервисов, процедур, продукций, подпрограмм и так далее. Такой подход позволяет работать одновременно с разными описаниями предметных областей, прибавляя к предикатам и продукции, и нейросети, и генетические алгоритмы, и традиционные вычислительные процедуры, и все другие в виде универсальных миварных отношений, которые представляются и хранятся перед обработкой в нашем миварном пространстве.

Глава 3.6.

Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах

⇒ *автор**:
- *Ковалев М.В.*
- *Крощенко А.А.*
- *Головко В.А.*

⇒ *аннотация**:
[В главе рассмотрен подход к *интеграции* и *конвергенции искусственных нейронных сетей* с *базами знаний* в *интеллектуальных компьютерных системах нового поколения* с помощью представления и интерпретации *искусственных нейронных сетей* в *базе знаний*. Описаны *Синтаксис, Денотационная и Операционная семантика Языка представления нейросетевых методов в базах знаний*. Описаны этапы построения *нейросетевых методов решения задач* с помощью интеллектуальной среды проектирования *искусственных нейронных сетей*.]

⇒ *подраздел**:
- *§ 3.6.1. Модели искусственных нейронных сетей, используемых в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

⇒ *ключевой знак**:
- *Язык представления нейросетевых методов решения задач в базах знаний*

⇒ *ключевое понятие**:
- *нейросетевой метод решения задач*
- *нейросетевая модель решения задач*
- *навык решения задач с помощью искусственных нейронных сетей*
- *действие по построению искусственных нейронных сетей*

⇒ *библиографическая ссылка**:
- *Голенков В.В..СтандОТОП-2021кн*
- *Castelvecchi D.CanWOtBB-2016art*
- *Ribeiro M.T..WhySITY-2016art*
- *Lundberg S.M..Advan iNIPS-2017art*
- *Garcez A.A..NeuraLaRCaC-2015art*
- *Besold T.R..NeuraSLaRaS-2017art*
- *Головко В.А..ПринцПСПР-2019ст*
- *Kroshchanka A..aNeuraSAtCV-2022art*
- *Головко В.А..НейроТОД-2017кн*
- *Kovalev M.V.Conve aIoANN-2022art*
- *Glorot X..Under tDoTDFN-2010art*
- *He K..DelviDiRSHP-2015art*
- *Гудфеллоу Я..ГлубоО-2017кн*
- *Хайкин С.НейроСПК-2006кн*
- *Duchi J..AdaptSMfOL-2011art*
- *Kingma D..Adam aMfSO-2014art*

Введение в Главу 3.6.

Современные *решатели задач интеллектуальных систем* все чаще сталкиваются с необходимостью решения *комплексных задач* с помощью различных традиционных и интеллектуальных *методов решения задач* в едином информационном ресурсе (в пределе — в единой *базе знаний*).

С другой стороны, *интеллектуальные компьютерные системы нового поколения* обладают, среди прочих, следующими способностями (см. *Голенков В.В..СтандОТОП-2021кн*):
- способность постоянно повышать качество решения задач;

- способность приобретать навыки решения принципиально новых задач;
- способность обосновывать свои решения;
- способность находить и устранять ошибки в своих решения (способность к интроспекции).

Представление различных *методов решения задач* в единой *базе знаний* обеспечивает *семантическую совместимость* этих методов. Решая задачу с помощью таких методов, система не взаимодействует с ними по принципу "входов-выходов". Напротив, единая память позволяет отслеживать преобразование входных знаний в реальном времени с помощью любых имеющихся методов, что обеспечивает способность к интроспекции и способность объяснять решения системы.

Перспективными и активно развивающимися *методами решения задач* являются ***искусственные нейронные сети*** (и.н.с.), что обуславливается, с одной стороны, развитием теории *и.н.с.*, а с другой — аппаратных возможностей машин, которые используются для их обучения.

Достоинствами *и.н.с.* можно назвать способность решения задач при неизвестных закономерностях, а так же способность решения задач без необходимости разработки проблемоориентированных подходов.

Большинство нейросетевых моделей работают как "черный ящик" (см. *Castelvecchi D.CanWOtBB-2016art*), что является одним из основных недостатков этого метода решения задач. Большой объем обрабатываемых этими моделями данных создает необходимость мониторинга, объяснения и понимания механизмов их работы с целью вербализации оценки и оптимизации деятельности *и.н.с.*

Современные задачи все чаще требуют обоснования своего решения. Появилось целое направление *Explainable AI*, в рамках которого предпринимаются различные попытки объяснить решения *и.н.с.* (см. *Ribeiro M.T..WhySITY-2016art*, *Lundberg S.M..Advan iNIPS-2017art*). Развиваются подходы, предлагающие интеграцию нейронных сетей с базами знаний (см. *Garcez A.A..NeuraLaRCaC-2015art*, *Besold T.R..NeuraSLaRaS-2017art*, *Головко В.А..ПринцПСПР-2019cm*, *Kroshchanka A..aNeuraSAtCV-2022art*).

Еще одним недостатком *и.н.с.* можно назвать эвристический характер процесса подбора архитектур моделей и параметров их обучения и высокие требования к объему знаний проектировщиков нейросетевых моделей.

Исходя из перечисленных способностей, наличие которых необходимо обеспечивать в *интеллектуальных компьютерных системах нового поколения*, встает проблема разработки подхода к интеграции *и.н.с.* в *базу знаний интеллектуальной системы* как в качестве *метода решения задач*, так и в качестве объекта автоматического проектирования новых методов. Решение этой проблемы позволит преодолеть указанные выше недостатки нейросетевого метода.

Можно выделить два основных направления *интеграции и.н.с. с базами знаний*

- Построение *интеллектуальных систем*, способных использовать *нейросетевые методы решения задач* наравне с другими имеющимися в системе методами для решения *комплексных задач*. Такие системы смогут учитывать семантику решаемых задач на более высоком уровне, что сделает решения более структурированными и прозрачными.
- Построение интеллектуальной среды по разработке, обучению и интеграции различных *и.н.с.*, совместимых с *базами знаний* через представление *и.н.с.* с помощью онтологических структур и их интерпретацию средствами представления знаний. Такая среда предоставит возможность интроспекции *и.н.с.*, возможность сохранения состояний *и.н.с.* после обучения и реконфигурации сети, что позволит производить более глубокий анализ ее работы. Так же формальное описание знаний в рамках предметной области *и.н.с.* поможет уменьшить порог вхождения разработчиков в область методов решения задач с помощью *и.н.с.*

§ 3.6.1. Модели искусственных нейронных сетей, используемых в ostis-системах

⇒ *подраздел**:
- *Денотационная семантика моделей искусственных нейронных сетей, используемых в ostis-системах Денотационная семантика моделей искусственных нейронных сетей, используемых в ostis-системах*
- *Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах*

⇒ *ключевой знак**:
- *Язык представления нейросетевого метода решения задач в базах знаний*
- *Денотационная семантика Языка представления нейросетевого метода решения задач в базах знаний*
- *Операционная семантика Языка представления нейросетевого метода в базах знаний*

⇒ *ключевое понятие**:

- *нейросетевой метод решения задач*
- *навык решения задач с помощью искусственных нейронных сетей*
- *формальный нейрон*
- *синаптическая связь*
- *слой и.н.с.*
- *взвешенной сумма нейрона*

⇒ *ключевое отношение**:
- *функция активации**
- *выходное значение'*

Как уже было описано в *Главе 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*, *решатель задач* занимается обработкой фрагментов базы знаний. На операционном уровне обработка сводится к добавлению, поиску, редактированию и удалению sc-узлов и sc-коннекторов *базы знаний*. На семантическом же уровне такая операция является *действием, выполняемым в памяти субъекта действия*, где, в общем случае, субъектом является *ostis-система*, а *база знаний* — ею памятью. *действие* определяется как процесс воздействия одной сущности (или некоторого множества сущностей) на другую сущность (или на некоторое множество других сущностей) в соответствии с некоторой целью (см. *Главу 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*).

Схожие задачи объединены в классы, для которых заданы обобщенные формулировки задач. Для *и.н.с.* выделены следующие классы задач:

- *задача классификации*. Задача построения классификатора, то есть отображения $\tilde{c} : X \to C$, где $X \in \mathbb{R}^m$ — признаковое пространство входного образа, $C = C_1, C_2, ...C_k$ — конечное и обычно небольшое множество меток классов.
- *задача регрессии*. Задача построения оценочной функции по примерам $(x_i, f(x_i))$, где $f(x)$ — неизвестная функция. *оценочная функция** — отображение вида $\tilde{f} : X \to \mathbb{R}$, где $X \in \mathbb{R}^m$ — признаковое пространство входных данных.
- *задача кластеризации*. Задача построения функции $a : X \to Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$ в соответствие с определенной метрикой расстояния $\rho(x, x')$, где *X* — множество объектов, *Y* — множество номеров (имен, меток) кластеров, $x, x' \in X$.
- *задача понижения размерности признакового пространства*. Задача построения функции $h : X \to Y$, сохраняющей заданные соотношения между точками множеств X и Y, где $X \subset \mathbb{R}^p$, $Y = h(X) \subset \mathbb{R}^q$, $q < p$.
- *задача управления*. Задача построения модели-регулятора состояния сложного динамического объекта.
- *задача фильтрации*. Задача построения модели, которая производит очистку исходного сигнала, содержащего некоторый шум и уменьшает влияние случайных ошибок в сигнале.
- *задача детекции*. Является частным случаем задачи классификации и задачи регрессии. Задача построения модели, осуществляющей обнаружение объектов определенных типов на фото- и видеоизображениях.
- *задача с ассоциативной памятью*. Задача построения модели, позволяющей выполнить реконструкцию исходного образа на основании сохраненных ранее образов.

Для классов задач формулируются классы методов их решения. *метод решения задач* определяется как *программа*, которая может быть как процедурной, так и декларативной. В свою очередь, *класс методов решения задач* определяется как множество всевозможных *методов решения задач*, имеющих общий язык представления этих методов. *язык представления методов* позволяет описывать *синтаксическую*, *денотационную* и *операционную семантику* этого *метода*.

Предлагается рассматривать *и.н.с.* как класс методов решения задач со своим языком представления. Таким образом, искусственная нейронная сеть - это ***нейросетевой метод решения задач***. В соответствии с *Технологией OSTIS*, спецификация класса методов решения задач сводится к спецификации соответствующего *языка представления методов*, то есть к описанию его синтаксической, денотационной и операционной семантики.

Для достижения семантической совместимости с другими *методами решения задач Технологии OSTIS*, предлагается описывать нейросетевые методы внутри семантической памяти, соответственно, *Синтаксис Языка представления нейросетевых методов решения задач в базах знаний* является *Синтаксисом SC-кода*, использующимся в *Технологии OSTIS* для представления знаний.

Таким образом, чтобы добавить в арсенал *Технологии OSTIS нейросетевые методы решения задач* и тем самым расширить круг задач, решаемых ostis-системами, необходимо описать денотационную и операционную семантику ***Языка представления нейросетевого метода решения задач в базах знаний***.

Денотационная семантика Языка представления нейросетевого метода решения задач в базах знаний описывается в рамках предметной области и соответствующей ей онтологии нейросетевого метода.

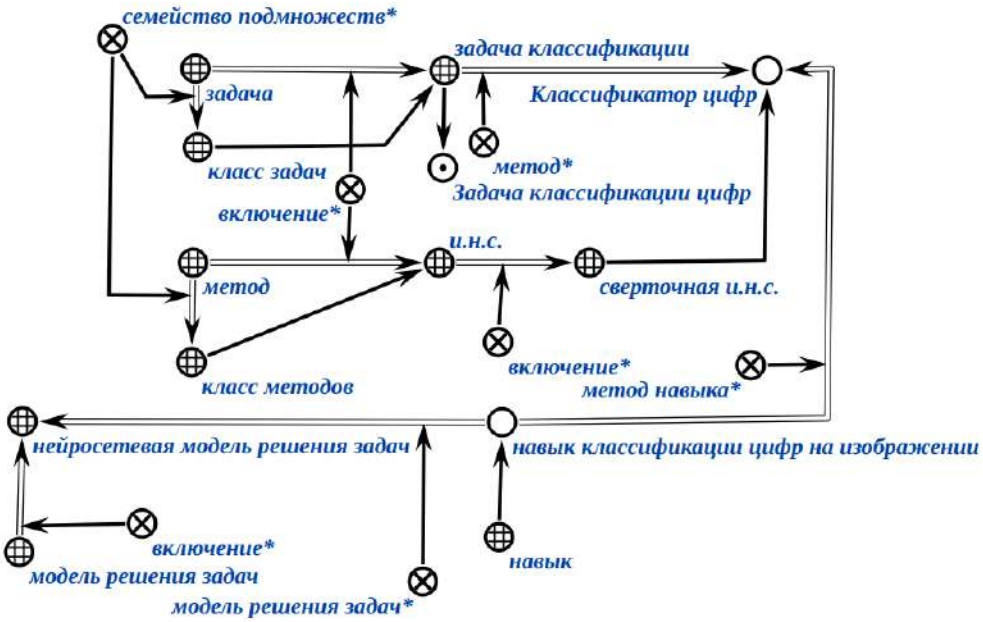
Операционной семантикой любого *языка представления методов решения задач* является спецификация семейства агентов, обеспечивающих интерпретацию любого метода, принадлежащего соответствующему классу методов. Это семейство является интерпретатором соответствующего метода решения задач. В рамках технологии

OSTIS такой интерпретатор называется *моделью решения задач*. Так как в рамках *Технологии OSTIS* используется многоагентный подход, то разработка нейросетевой модели решения задач сводится к разработке агентно-ориентированной модели интерпретации *и.н.с.*

Понятие *навыка* описывает метод, интерпретация которого полностью может быть осуществлена данной кибернетической системой, в памяти которой хранится указанный метод. Таким образом, формируя спецификацию в ostis-системе для нейросетевого метода решения задач и нейросетевой модели решения задач можно говорить о наличии у такой системы ***навыка решения задач с помощью и.н.с.***.

На рисунке *SCg-текст. Фрагмент теоретико-множественной онтологии и.н.с.* представлен фрагмент теоретико-множественной онтологии *и.н.с.*, описывающий связь таких понятий и узлов, как:
- класс задач, решаемых с помощью и.н.с.(для примера, взят класс задач классификации);
- класс нейросетевых методов решения задач;
- нейросетевая модель решения задач;
- навык решения задач с помощью и.н.с.;
- конкретные задачи и методы их решения (для примера взята конкретная обученная сверточная и.н.с.).

SCg-текст. Фрагмент теоретико-множественной онтологии и.н.с.
=



Использование *и.н.с.* как *метода решения задач* подразумевает использование уже спроектированной и обученной *и.н.с.* Однако наличие языка описания нейросетевого метода решения задач в памяти *ostis-системы* открывает дорогу для автоматизации самих процессов проектирования и обучения *и.н.с.* Такая автоматизация представляется отдельными классами задач и соответствующими навыками их решения. Подход к такой автоматизации описан в *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*.

Пункт 3.6.1.1. Денотационная семантика моделей искусственных нейронных сетей, используемых в ostis-системах

Как уже было сказано, *Денотационная семантика Языка представления нейросетевых методов в базах знаний* описывается в рамках предметной области и соответствующей ей онтологии нейросетевого метода

Максимальным классом объектов исследования предметной области искусственных нейронных сетей является *искусственная нейронная сеть*.

искусственная нейронная сеть
:= [и.н.с.]
:= [нейросетевой метод]

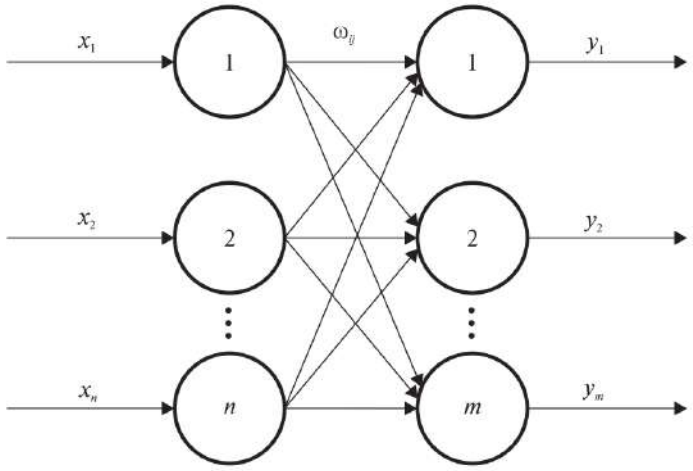
:= [множество искусственных нейронных сетей]
:= [нейронная сеть]
⇒ *пояснение**:
[Совокупность нейронных элементов и связей между ними (см. *Головко В.А..НейроТОД-2017кн*).]

Искусственная нейронная сеть состоит из ***формальных нейронов***, которые связаны между собой посредством ***синаптических связей***. Нейроны организованы в ***слои***. Каждый нейрон слоя принимает сигналы со входящих в него синаптических связей, обрабатывает их единым образом с помощью заданной ему или всему слою ***функции активации*** и передает результат на выходящие из него синаптические связи.

Архитектурой *и.н.с.* будем называть совокупность информации о структуре ее слоев, формальных нейронов, синаптических связей и функций активаций. То есть то, что можно обучить и использовать для решения задач.

Пример архитектуры *и.н.с.* представлен на рисунке *Рисунок. Пример архитектуры и.н.с.*.

Рисунок. Пример архитектуры и.н.с.
=



В соответствии с тем, какая у *и.н.с.* архитектура, можно выделить следующую иерархию классов *и.н.с.* Рассмотрим эту иерархию в *SCn-коде*.

искусственная нейронная сеть
:= [нейросетевой метод]
⇐ *включение**:
метод
⇒ *разбиение**:
Типология и.н.с. по признаку направленности связей^
= {• *и.н.с. с прямыми связями*
⇒ *декомпозиция**:
{• *персептрон*
⇒ *декомпозиция**:
{• *персептрон Розенблатта*
• *автоэнкодерная и.н.с.*
}
• *машина опорных векторов*
• *и.н.с. сеть радиально-базисных функций*
• *сверточная и.н.с.*
}
• *и.н.с. с обратными связями*
⇒ *декомпозиция**:
{• *и.н.с. Хопфилда*
• *и.н.с. Хэмминга*
}
• *рекуррентная искусственная нейронная сеть*
⇒ *декомпозиция**:

{• *и.н.с. Джордана*
• *и.н.с. Элмана*
• *мультирекуррентная и.н.с.*
• *LSTM-элемент*
• *GRU-элемент*
}
}
⇒ *разбиение**:
Типология и.н.с. по признаку полноты связей^
= {• *полносвязная и.н.с.*
• *слабосвязная и.н.с.*
}

Рассмотрим архитектурные компоненты подробнее.

формальный нейрон
:= [искусственный нейрон]
:= [нейрон]
:= [ф.н.]
:= [нейронный элемент]
:= [множество нейронов искусственных нейронных сетей]
:= [математическая модель биологического нейрона]
⊂ *искусственная нейронная сеть*
⇒ *пояснение**:
[Основной элемент *и.н.с.*, применяющий свою *функцию активации* (см. *Головко В.А..НейроТОД-2017кн*) к сумме произведений входных сигналов на весовые коэффициенты:

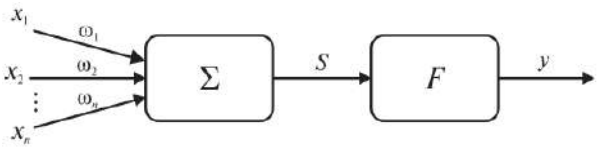
$$y = F\left(\sum_{i=1}^{n} w_i x_i - T\right) = F(WX - T)$$

где $X = (x_1, x_2, ..., x_n)^T$ — вектор входного сигнала; $W - (w_1, w_2, ..., w_n)$ — вектор весовых коэффициентов; T — пороговое значение; F — *функция активации*.]

Отдельный *формальный нейрон* является *искусственной нейронной сетью* с одним нейроном в единственном слое. *формальные нейроны* могут быть классифицированы следующим образом:
- Полносвязный *формальный нейрон* — нейрон, у которого есть полный набор связей с нейронами предшествующего слоя. Отдельный обрабатывающий элемент *и.н.с.*, выполняющий функциональное преобразование взвешенной суммы элементов вектора входных значений с помощью *функции активации*.
- Сверточный *формальный нейрон* — отдельный обрабатывающий элемент *и.н.с.*, выполняющий функциональное преобразование результата операции свертки матрицы входных значений с помощью *функции активации*. Сверточный *формальный нейрон* может быть представлен полносвязным *формальным нейроном*.
- Рекуррентный *формальный нейрон* — нейрон, имеющий обратную связь с самим собой или с другими нейронами *и.н.с.*.

Схематически *формальный нейрон* можно представить в виде следующей модели (рисунок *SCg-текст. Формальный нейрон*).

SCg-текст. Формальный нейрон
=



Определим понятия *синаптической связи* и *слоя и.н.с.*:

синаптическая связь
:= [синапс]
⊂ *ориентированная пара*

⇒ *пояснение**:
[ориентированная пара, первым компонентом которой является нейрон, из которого исходит сигнал, а вторым компонентом — нейрон, который принимает этот сигнал]

слой и.н.с.
≔ [слой]
≔ [слой искусственной нейронной сети]
≔ [множество слоев искусственных нейронных сетей]
⊂ *искусственная нейронная сеть*
⇒ *пояснение**:
[множество нейронных элементов, на которые в каждый такт времени параллельно поступает информация от других нейронных элементов сети (см. *Головко В.А..НейроТОД-2017кн*)]
⇒ *пояснение**:
[множество формальных нейронов, осуществляющих параллельную независимую обработку вектора или матрицы входных значений]

Отдельный слой является искусственной нейронной сетью с одним слоем. Следует отметить принципиальную важность этого замечания. Один *слой и.н.с.* уже является нейронной сетью, поскольку над ним можно производить все основные операции, которые производятся над "большой" *и.н.с.* (его можно обучить и использовать для решения определенной задачи).

слои и.н.с. могут быть классифицированы следующим образом (по признаку операции, осуществляемой слоем):
- *полносвязный слой и.н.с.* — слой, в котором каждый нейрон является полносвязным;
- *сверточный слой и.н.с.* — слой, в котором каждый нейрон является сверточным;
- *слой и.н.с.* нелинейного преобразования — слой, осуществляющий нелинейное преобразование входных данных;
- *dropout слой и.н.с.* — слой, реализующий технику регуляризации dropout;
- *pooling слой и.н.с.* — подвыборочный слой;
- *слой и.н.с.* батч-нормализации.

Как правило, слой нелинейного преобразования выделяется в отдельный слой только в программных реализациях. Фактически он рассматривается как финальный этап расчета выходной активности любого нейрона — применение *функции активации*.

dropout-слой функционирует только во время обучения *и.н.с.*. Поскольку полносвязные слои имеют большое количество настраиваемых параметров, они подвержены эффекту *переобучения*. Один из способов устранить такой негативный эффект — выполнить частичный отсев результатов на выходе полносвязного слоя. На этапе обучения техника dropout позволяет отбросить выходную активность некоторых нейронов с определенной, заданной вероятностью. Выходная активность "отброшенных" нейронов полагается равной нулю.

Назначение подвыборочного слоя — в осуществлении уменьшения размерности входных данных.

Нужно отметить, что данный перечень неполный — разновидности *слоев и.н.с.* появляются практически в каждой заслуживающей внимания публикации по нейросетевым алгоритмам и на текущий момент их существует достаточно много, однако, как правило, при построении более традиционных архитектур ограничиваются только приведенными вариантами слоев.

слои и.н.с. также могут быть классифицированы по исполняемой роли в рамках архитектуры (место в последовательности *слоев и.н.с.*).

Так, например, слой, расположенный первым, называется распределяющим. Слои, расположенные далее, за исключением последнего, называются обрабатывающими. Наконец, последний слой носит название выходного *слоя и.н.с.*

Последний архитектурный компонент *и.н.с.* — это функция активации:

функция активации*
≔ [функция активации нейрона*]
∈ *неролевое отношение*
∈ *бинарное отношение*
⇒ *пояснение**:
[неролевое отношение, связывающее *формальный нейрон* с функцией, результат применения которой к ***взвешенной сумме нейрона*** определяет его ***выходное значение***.]
⇒ *первый домен**:
формальный нейрон
⇒ *второй домен**:

функция

Перечислим некоторые, наиболее известные и применяемые типы функций активации:

- линейная функция
  - ⇒ *формула**:
    [
    $$y = kS$$
    где k — коэффициент наклона прямой, S — в.с.;]
- пороговая функция
  - ⇒ *формула**:
    [
    $$y = sign(S) = \begin{cases} 1, S > 0, \\ 0, S \leq 0 \end{cases}$$
    ;]
- сигмоидная функция
  - ⇒ *формула**:
    [
    $$y = \frac{1}{1 + e^{-cS}}$$
    где $c > 0$ — коэффициент, характеризующий ширину сигмоидной функции по оси абсцисс, S — в.с.;]
- функция гиперболического тангенса
  - ⇒ *формула**:
    [
    $$y = \frac{e^{cS} - e^{-cS}}{e^{cs} + e^{-cS}}$$
    где $c > 0$ — коэффициент, характеризующий ширину сигмоидной функции по оси абсцисс, S — в.с.;]
- функция softmax
  - ⇒ *формула**:
    [
    $$y_j = softmax(S_j) = \frac{e^{S_j}}{\sum_j e^{S_j}}$$
    где S_j — в.с. j-го выходного нейрона;]
- функция ReLU
  - ⇒ *формула**:
    [
    $$y = F(S) = \begin{cases} S, S > 0, \\ kS, S \leq 0 \end{cases}$$
    где $k = 0$ или принимает небольшое значение, например, 0.01 или 0.001.]

В рамках *предметной области* формализована иерархия параметров *и.н.с.*

параметр и.н.с.
⊂ *параметр*
⇒ *разбиение**:
= {• *настраиваемый параметр и.н.с.*
⇒ *декомпозиция**:
{• *весовой коэффициент синаптической связи*
• *пороговое значение*
• *ядро свертки*
}
• *архитектурный параметр и.н.с.*
⇒ *декомпозиция**:
{• *количество слоев*
• *количество нейронов*
• *количество синаптических связей*
}
}

Так же в *Предметную область нейросетевых методов* добавлены понятия для описания метрик эффективности *нейросетевых методов*. Данные метрики учитываются *решателем задач* при принятии решения об использовании того или иного *нейросетевого метода*.

Метрики могут быть классифицированы по типу решаемой задачи.

метрика оценки качества и.н.с.
⇒ *разбиение**:
Типология метрик по признаку решаемой задачи^
= {• *классификационные метрики*
⇒ *декомпозиция**:
{• *точность и.н.с.*
• *полнота и.н.с.*
• *F1-метрика*
}
• *регрессионные метрики*
⇒ *декомпозиция**:
{• *MAE*
• *MAPE*
• *RMSE*
}
}

точность и.н.с.
:= [precision]
:= [доля верно идентифицированных положительных исходов в общем числе исходов, которые были идентифицированы как положительные]
⇒ *формула**:
[
$$PRE = \frac{TP}{TP + FP}$$
где *TP* и *FP* — число истинно-положительных и ложно-положительных предсказаний нейронной сети соответственно]

полнота и.н.с.
:= [recall]
:= [доля верно идентифицированных положительных исходов в общем числе положительных исходов]
⇒ *формула**:
[
$$REC = \frac{TP}{TP + FN}$$
где *TP* и *FN* — число истинно-положительных и ложно-отрицательных предсказаний нейронной сети соответственно]

F1-метрика
⇒ *формула**:
[
$$F1 = 2 * \frac{PRE * REC}{PRE + REC}$$
где *PRE* и *REC* — точность и полнота и.и.с. соответственно]

MAE
:= [mean absolute error]
⇒ *формула**:
[$\frac{1}{N}\sum_{i=1}^{N}|y^i_{etalon} - y^i_{predicted}|$,
y^i_{etalon} — эталонное значение,
$y^i_{predicted}$ — значение, полученное и.н.с.,
N — объем обучающей выборки]

MAPE
:= [mean absolute percentage error]

⇒ *формула**:
[$\frac{1}{N}\sum_{i=1}^{N}\frac{|y^i_{etalon}-y^i_{predicted}|}{y^i_{etalon}} * 100\%$,
y^i_{etalon} — эталонное значение,
$y^i_{predicted}$ — значение, полученное и.н.с.,
N — объем обучающей выборки]

RMSE
:= [root mean squared error]
⇒ *формула**:
[$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y^i_{etalon} - y^i_{predicted})^2}$,
y^i_{etalon} — эталонное значение,
$y^i_{predicted}$ — значение, полученное и.н.с.,
N — объем обучающей выборки]

С помощью выделенных понятий становится возможна формализация в *базе знаний* архитектуры конкретных *и.н.с.* В качестве примера, на рисунке *SCg-текст. Пример формализации архитектуры искусственной нейронной сети в базе знаний* представлен пример формализации полносвязной двухслойной *и.н.с.* с двумя нейронами на входном слое и одном нейроне на обрабатывающем слоев.

Следует отметить, что в практике авторов еще не было необходимости явно представлять и.н.с., как это показано на рисунке *SCg-текст. Пример формализации архитектуры искусственной нейронной сети в базе знаний*. Чаще всего, представление и.н.с. сводилось к представлению ее операционной семантики в виде SCP-программы, как это будет показано далее.

Пункт 3.6.1.2. Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах

Операционная семантика Языка представления нейросетевого метода в базах знаний задается *многоагентный подход* к интерпретации *искусственных нейронных сетей* и спецификацией соответствующих действий.

Нейросетевой метод описан в виде программы на некотором *языке программирования*, который может быть как внешним по отношению к *ostis-системе*, так и внутренним (на данный момент, *Язык SCP*). Каждому такому *языку программирования* соответствует некоторая дочерняя *предметная область Предметная область нейросетевых методов* (см. *Kovalev M.V.Conve aIoANN-2022art*).

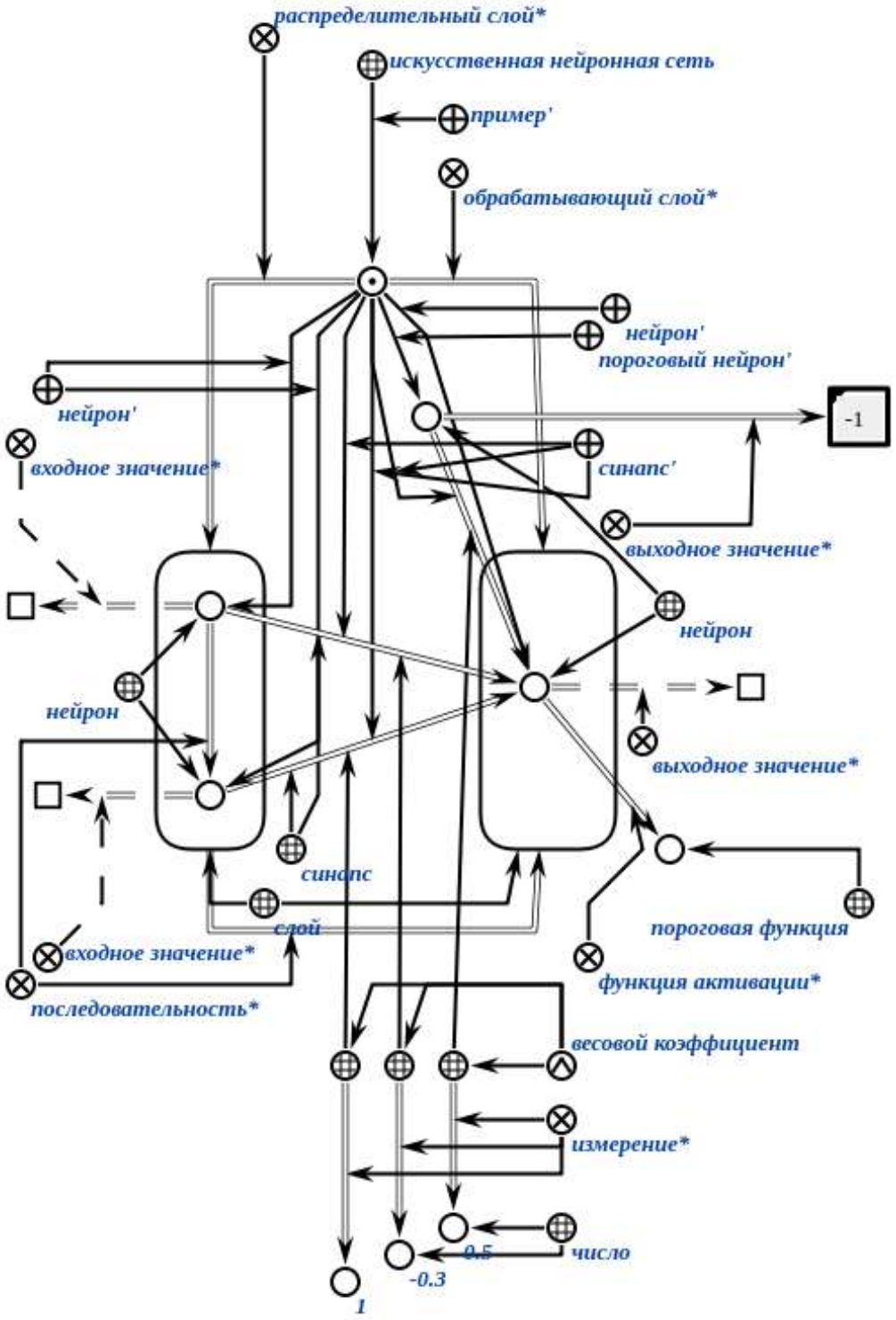
Предметная область нейросетевых методов
:= [Предметная область искусственных нейронных сетей]
⇒ *дочерняя предметная область**:
{• *Предметная область нейросетевых методов SCP*
• *Предметная область нейросетевых методов Python*
• *Предметная область нейросетевых методов C++*
}

В случае описания *нейросетевого метода* на внешнем языке, такой метод описывается в соответствующей предметной области, в рамках которой также специфицируется действие интерпретации данного метода. Данному действию соответствует агент, реализованный на соответствующем *языке программирования*.

Однако для достижения конвергенции и интеграции необходимо описывать нейросетевые методы на внутреннем языке ostis-системы, которым является *Язык SCP*. Интерпретация *scp-программы* сводится к агентно-ориентированной обработке действий в sc-памяти. Этими действиями являются *scp-операторы*.

действие интерпретации слоя и.н.с.
⇒ *декомпозиция**:
{• *действие вычисления взвешенной суммы всех нейронов слоя*
• *действие вычисления функции активации всех нейронов слоя*
• *действие интерпретации сверточного слоя*
• *действие интерпретации пулинг слоя*
}

SCg-текст. Пример формализации архитектуры искусственной нейронной сети в базе знаний
=



Для описания спецификации указанных действий необходимо ввести понятия *ориентированного множества чисел* и *матрицы*, с помощью которых задаются входные значения *и.н.с.*, выходные значения *и.н.с.*, матрицы весовых коэффициентов и прочее.

Каждый элемент ориентированного множества чисел является некоторым числом. Числа могут быть представлены в виде sc-узлов, либо с помощью строкового представления всего множества, для чего используется специальное отношение *строковое представление ормножества чисел**, которое введено в целях оптимизации некоторых вариантов реализации агента, интерпретирующего действие, использующее понятие ориентированного множества чисел.

ориентированное множество чисел
:= [ормножество чисел]
⇐ *включение**:
число
⇐ *включение**:
ориентированное множество
⇐ *первый домен**:
*строковое представление ормножества чисел**

матрица является *ориентированным множеством ориентированных множеств* чисел равной мощности.

1. Действие вычисления взвешенной суммы всех нейронов слоя

Аргументы(*объекты'*) этого действия задаются следующими отношениями:

входной вектор'
⇒ *первый домен**:
действие интерпретации и.н.с.
⇒ *второй домен**:
ориентированное множество чисел

матрица весовых коэффициентов нейронов слоя'
⇒ *первый домен**:
действие по обработке и.н.с.
⇒ *второй домен**:
матрица

Результатом действия(*результат'*) является ориентированное множество чисел, являющихся взвешенной суммой нейронов соответствующего слоя.

Пример спецификации действия вычисления взвешенной суммы всех нейронов слоя для слоя с двумя нейронами и входным вектором размерностью 2 приведен на рисунке *SCg-текст. Пример действия вычисления взвешенной суммы всех нейронов слоя*.

2. Действие вычисления функции активации всех нейронов слоя

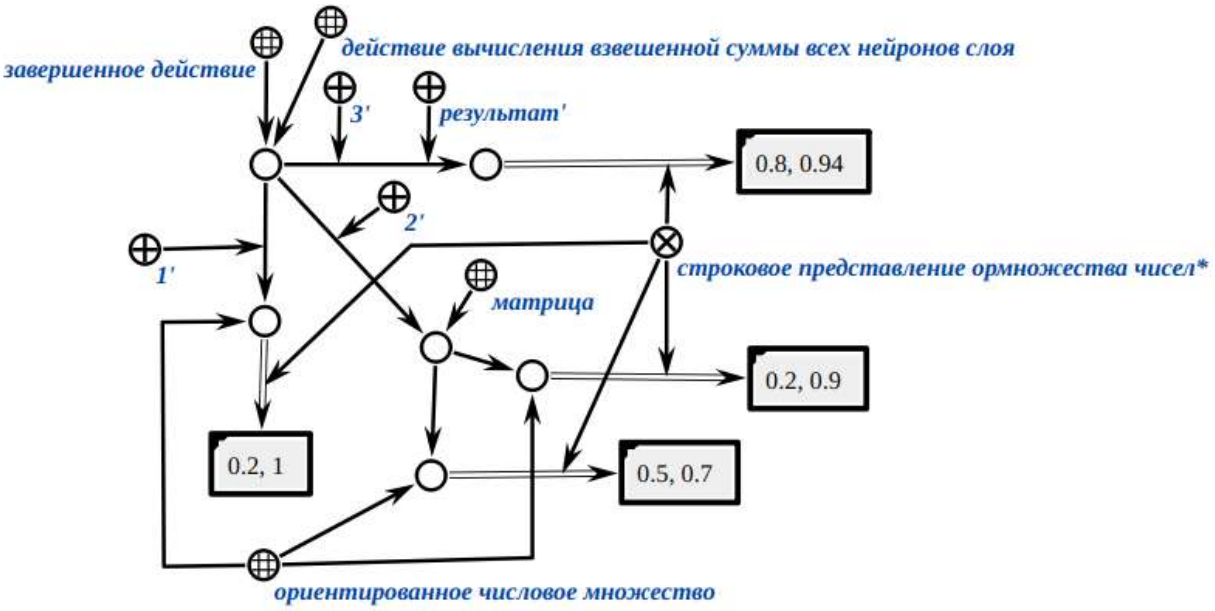
Аргументы этого действия задаются следующими отношениями:

вектор взвешенных сумм нейронов слоя'
⇒ *первый домен**:
действие по обработке и.н.с.
⇒ *второй домен**:
ориентированное множество чисел

вектор порогов нейронов слоя'
⇒ *первый домен**:
действие по обработке и.н.с.
⇒ *второй домен**:
ориентированное множество чисел

функция активации'
⇒ *первый домен**:
действие по обработке и.н.с.
⇒ *второй домен**:
функция

SCg-текст. Пример действия вычисления взвешенной суммы всех нейронов слоя
=



Результатом действия является ориентированное множество чисел, являющихся выходными значениями нейронов слоя.

3. Действие интерпретации сверточного слоя

Аргументы этого действия задаются следующими отношениями:

входная матрица'
⇒ *первый домен**:
действие интерпретации и.н.с.
⇒ *второй домен**:
матрица

ядро свертки'
⇒ *первый домен**:
действие интерпретации сверточного слоя
⇒ *второй домен**:
матрица

шаг свертки'
⇒ *первый домен**:
действие интерпретации сверточного слоя
⇒ *второй домен**:
число

Результатом действия является матрица, полученная в результате свертки входной матрицы с ядром свертки.

4. Действие интерпретации пулинг слоя

Аргументы этого действия задаются следующими отношениями:

входная матрица'
⇒ *первый домен**:
действие интерпретации и.н.с.
⇒ *второй домен**:
матрица

размер окна пулинга'
⇒ *первый домен**:
действие интерпретации пулинг слоя

⇒ *второй домен**:
матрица

размер окна пулинга'
⇒ *первый домен**:
действие интерпретации пулинг слоя
⇒ *второй домен**:
матрица

шаг окна пулинга'
⇒ *первый домен**:
действие интерпретации пулинг слоя
⇒ *второй домен**:
число

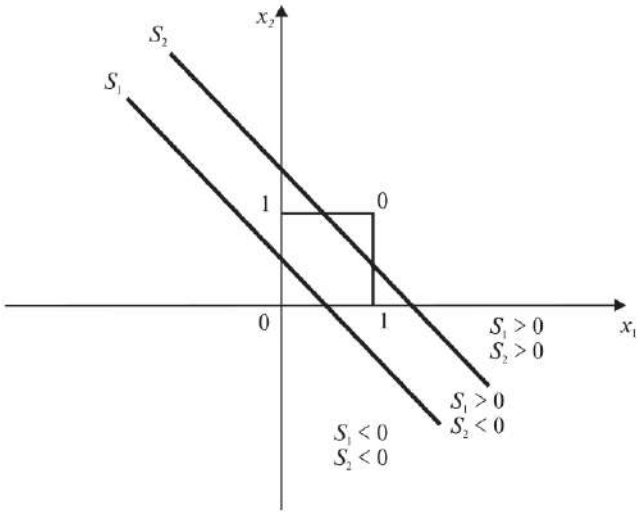
Результатом действия является матрица, полученная в результате пулинга входной матрицы.

При необходимости задавать различные аргументы для нейронов одного и того же слоя, можно специфицировать соответствующие действия, однако на данный момент этого не было произведено из-за слабой изученности подобного рода *нейросетевых моделей решения задач*.

Спецификация агентов, соответствующих указанным действиям, задает агентно-ориентированную модель интерпретации искусственных нейронных сетей. Реализация этой модели будет называться интерпретатором искусственных нейронных сетей

Рассмотрим пример описания на *нейросетевого метода*, решающего задачу, которая формулируется следующим образом: вычислить результат логической операции “ИСКЛЮЧАЮЩЕЕ ИЛИ” для значений двух логических переменных. На рисунке *Рисунок. Решение задачи "ИСКЛЮЧАЮЩЕЕ ИЛИ"* представлено решение этой задачи с помощью сигнальной функции.

Рисунок. Решение задачи "ИСКЛЮЧАЮЩЕЕ ИЛИ"
=



В работе *Головко В.А..НейроТОД-2017кн* описан однослойный персептрон, решающий поставленную задачу. Персептрон состоит из двух входных нейронов и одного выходного, с заданным порогом в 0,5 и сигнальной функцией активации:

$$F(S) = \begin{cases} 1, 0 < S < 0, \\ 0, else \end{cases}$$

Весовые коэффициенты синапсов входного слоя равны 1. На рисунке *Рисунок. Схема однослойного персептрона, решающего задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ"* представлена схема персептрона.

Данному персептрону соответствует метод, представленный в базе знаний ostis-системы на описанном в этой главе языке представления нейросетевых методов SCP. Данный метод представлен на рисунке *Рисунок. Метод, решающий задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ", представленный с помощью языка представления нейросетевых методов SCP*.

Рисунок. Схема однослойного персептрона, решающего задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ"
=

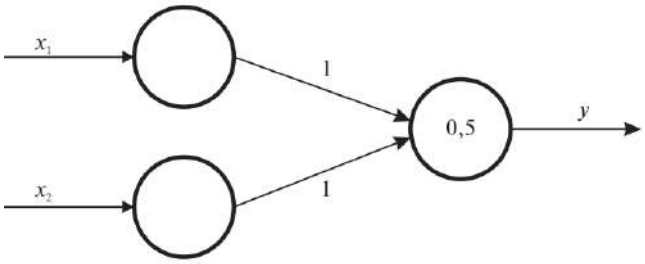


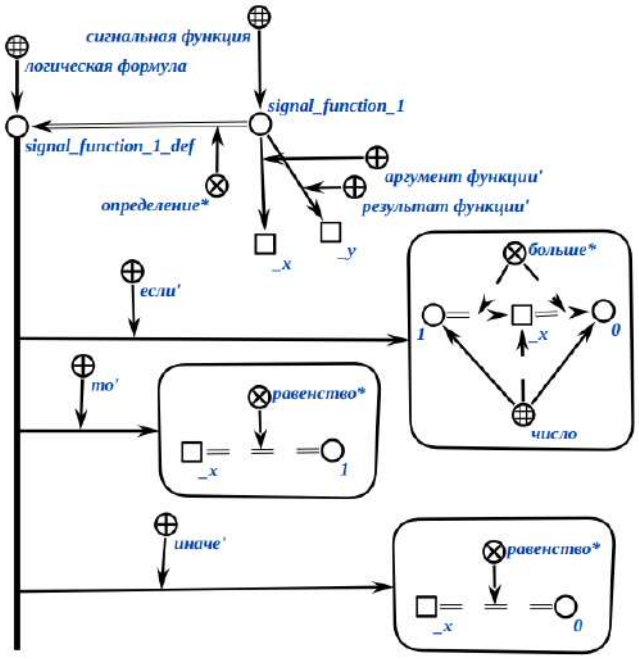
Рисунок. Метод, решающий задачу "ИСКЛЮЧАЮЩЕЕ ИЛИ", представленный с помощью языка представления нейросетевых методов SCP
=

```
proc_exclusive_or_ann
<- scp_method;
<- perceptron;
-> rrel_key_sc_element: _process1;;

proc_exclusive_or_ann = [*
_process1
_<- scp_process;
_-> rrel_1:: rrel_in:: _input_vector;
_-> rrel_2:: rrel_out:: _output_vector;

_<= nrel_decomposition_of_action:: _... (*

        _-> rrel_1:: _..operator1
        (*
                _<- action_calculate_weighted_sum_of_all_neurons_of_layer;;
                _-> rrel_1:: rrel_fixed:: rrel_scp_var:: rrel_input_vector:: _input_vector;;
                _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_synopsis_weight_matrix:: ...
                        (*
                                <- matrix;;
                                -> rrel_1: ...
                                (*
                                        <- number_oriented_set;;
                                        => nrel_oriented_set_string_representation: [1, 1];;
                                *);;
                        *);;
                _-> rrel_3:: rrel_assign:: rrel_scp_var:: rrel_result:: _weighted_sum_vector;

                _=> nrel_goto:: _..operator2;;
        *);;

        _-> _..operator2
        (*
                _<- action_calculate_activation_function_of_all_neurons_of_layer;;
                _-> rrel_1:: rrel_fixed:: rrel_scp_var:: _weighted_sum_vector;;
                _-> rrel_2:: rrel_fixed:: rrel_scp_const:: rrel_threshold_set:: ...
                        (*
                                <- number_oriented_set;;
                                => nrel_oriented_set_string_representation: [0.5];;
                        *);;
                _-> rrel_3:: rrel_fixed:: rrel_scp_const:: rrel_activation_fun:: signal_fun;;
                        (*
                                <- signal_activation_function;;
                                => nrel_definition: signal_function_1_def;;
                        *);;
                _-> rrel_4:: rrel_assign:: rrel_scp_var:: _output_vector;;

                _=> nrel_goto:: _..operator3;;
        *);;

        _-> _..operator3 (* <- return;; *);;
*);;
*];;
```

Описание метода состоит из последовательности двух обобщенных спецификаций действий — действия вычисления взвешенной суммы всех нейронов слоя и действия вычисления функции активации для всех нейронов слоя.

Сигнальная функция активации, использующаяся в персептроне, в памяти ostis-системы определяется логической формулой, представленной на рисунке *SCg-текст. Представление сигнальной функции активации в памяти ostis-системы*.

SCg-текст. Представление сигнальной функции активации в памяти ostis-системы
=



Любой агент, интерпретирующий действия с заданными с помощью отношения *функция активации'* аргументами, должен использовать интерпретатор математических функций. использующихся в качестве функций активации.

§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем

⇒ *ключевое понятие**:

- *действие трансляции условия задачи*
- *действие классификации задачи*
- *действие поиска подходящей обучающей выборки*
- *действие формирования требований к обучающей выборке*
- *действие очистки выборки*
- *действие выявления содержательных признаков*
- *действие трансформации выборки*
- *действие разбиения выборки*
- *действие выбора класса нейросетевых методов*
- *действие формирования спецификации входов и выходов и.н.с.*
- *действие выбора метода оптимизации*
- *действие выбора минимизируемой функции ошибки*
- *действие начальной инициализации и.н.с.*
- *действие выбора гиперпараметров и.н.с.*
- *метод обучения с учителем*
- *метод обучения без учителя*
- *действие обучения и.н.с.*

Наличия *Языка представления нейросетевых методов в базах знаний* и его интерпретатора позволяет обеспечить интерпретацию *нейросетевого метода* в памяти *ostis-системы*. Наличие в единой памяти не только экземпляров

методов, но и понятий, их описывающих, создает основу для автоматизации процесса построения нейросетевых методов. В памяти *ostis-системы* хранятся знания о том, методы какого класса могут решить задачу заданного класса, но экземпляров класса этого метода может не быть представлено в системе. На этот случай система должна иметь возможность сообщить пользователю о возможности решения, для которого, однако, необходимо погрузить в систему определенный метод. Так как система хранит в единой памяти задачу и требования к методу ее решения, появляется возможность спроектировать необходимый метод. Для этого необходимо наличие среды проектирования методов соответствующих классов. В случае *нейросетевого метода*, речь идет об интеллектуальной среде построения *нейросетевых методов*.

В основе интеллектуальной среды построения *нейросетевых методов* лежат соответствующие другу другу иерархии действий, задач и методов построения *и.н.с.* Наличие такой иерархии позволит описать язык представления методов построения *и.н.с.* и разработать интерпретатор этого языка.

Построение иерархии соответствующих действий построения *и.н.с.* следует начать с изучения этапов проектирования и обучения *и.н.с.*, которые, в общем случае, выполняют все разработчики и.н.с.:

1. Постановка задачи

Постановка задачи включает в себя описание входных данных (изображения/видео, временные ряды, текст), выходных данных и требований к методу решения (скорость, затраты по памяти и так далее). Также описывается дополнительная информация, которая может помочь в построении метода решения задачи (к примеру, спецификация обучающей выборки, если таковая имеется). Обычно, на данном этапе разработчик и.н.с. определяет класс задачи, формирует требования к обучающей выборке, если она не предоставлена.

Выполнение данного этапа средой проектирования *и.н.с.* подразумевает выполнение следующих действий:

- ***действие трансляции условия задачи***. Действие транслирует заданное с помощью *интерфейса ostis-системы* (к примеру, естественно-языкового интерфейса) описание задачи в память ostis-системы. Действие необходимо в случае, когда условие задачи задается пользователем. Необходимо понимать, что описание задачи поступает в базу знаний не только от *пользовательского интерфейса*. К примеру, задача может быть сформулирована самой системой в ходе ее жизнедеятельности. Данное действие является общим для всех ostis-систем, поэтому его рассмотрение выходит за рамки рассмотрения процесса построения интеллектуальной среды проектирования *и.н.с.*
- ***действие классификации задачи***. Действие определяет класс задачи (задача регрессии, детекции, кластеризации и так далее), исходя из описания задачи в базе знаний.
- ***действие поиска подходящей обучающей выборки***. В базе знаний может храниться набор спецификаций выборок, к которым у ostis-системы есть доступ. Действие производит поиск выборок, которые могут быть использованы в качестве обучающей выборки.
- ***действие формирования требований к обучающей выборке***. Если обучающая выборка не была предоставлена и не была найдена, то необходимо сформировать описание требований к обучающей выборке, которое можно будет транслировать на язык пользовательского интерфейса и запросить необходимую выборку у пользователя.

2. Предобработка выборки: очистка

На этом этапе обнаруживаются признаки, которые имеют в общем случае некорректные значения (например, для каких-то образов значение признака может иметь неопределенное значение, либо значение, не совпадающее по типу, либо аномально большое или очень маленькое значение, которое встречается в редком числе случаев). Для признаков, имеющих неопределенное значение, может быть применены различные методы устранения, например, такие значения могут быть заменены средним значением этого признака, рассчитанным по всем образам (для непоследовательных данных), либо они могут быть заменены средним значением по соседним образам (в случае временных рядов), либо каким-то фиксированным значением. Радикальная мера решения проблемы — удаление образов, имеющих неопределенные значения признаков из выборки. Однако его лучше применять, если образов с отсутствующими значениями признаков немного. Для выбросов и аномалий применяются схожие стратегии (но только в том случае, если задача не состоит в прогнозировании этих аномалий).

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия очистки выборки***, которое выполняется в случае обработки выборки, которая ранее не была представлена в памяти системы (к примеру, была получена от пользователя). Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий очистки данных и реализации методов применения этих стратегий.

3. Предобработка выборки: выявление содержательных признаков

Осуществляется инжиниринг признаков, состоящий в отборе признаков, влияющих на результат работы модели, несодержательные признаки, которые никак не коррелируют с выходом модели, удаляются. Цель этого этапа — уменьшение размерности пространства признаков для снижения влияния эффекта переобучения на модель.

Для снижения размерности признакового пространства может применяться методы отбора признаков и выделения признаков.

При отборе признаков, осуществляется формирование подмножества из исходных признаков (алгоритм последовательного обратного отбора, рекурсивный алгоритм обратного устранения признаков, алгоритмы с использованием случайных лесов).

При выделении признаков из набора признаков извлекается информация для построения нового подпространства признаков (алгоритмы с использованием автоэнкодера).

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия выявления содержательных признаков***. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации стратегий уменьшения размерности признакового пространства и реализации методов применения этих стратегий.

4. Предобработка выборки: трансформация

На этом этапе осуществляется подготовка данных к обучению. Здесь следует уделить особое внимание наличию категориальных признаков, чаще всего заданных строковыми типами. Эти признаки могут быть номинальными и порядковыми. Для кодирования порядковых признаков чаще всего применяют последовательный числовой код (1, 2, 3,...). Для кодирования номинальных такое решение неверно, так как эти признаки равноправны и не могут сравниваться по числовому коду (например, пол — 0/1). Для номинальных признаков применяется способ прямого кодирования, заключающийся в создании и использовании фиктивных признаков по количеству значений исходного. Например, признак пол (мужской, женский) преобразуется в два новых признака мужской и женский с соответствующими значениями для имеющихся образов.

Масштабирование признаков предполагает приведение значений признаков к одному общему интервалу — это особенно актуально для признаков, имеющих несоразмерные выборочные средние значения по всем образам — например, один признак в среднем имеет значение 10.000, а другой 12. Это может проявится в выполнении минимизации только по признаку с наибольшими значениями и плохой сходимости метода обучения. Чаще всего масштабирование соответствует выполнению нормализации на отрезок (min-max нормализация):

$$x^i_{norm} = \frac{x^i - x_{min}}{x_{max} - x_{min}}$$

где x^i — значение признака для отдельно взятого образа *i*, x_{min} — наименьшее значение для признака, x_{max} — наибольшее значение для признака.

Другой вариант масштабирования — применение стандартизации признаков:

$$x^i_{std} = \frac{x^i - \mu(x)}{\sigma(x)},$$

$\mu(x)$ — выборочное среднее отдельного признака, $\sigma(x)$ — стандартное отклонение.

Стандартизация сохраняет полезную информацию о выбросах в исходных данных и делает алгоритм обучения менее чувствительным к ним.

Дискретизация применяется для перехода от вещественного признака к порядковому за счет кодирования интервалов одним значением (например, если признак отражает возраст человека, то может быть произведена дискретизация значений с выделением определенных возрастных групп, где каждая группа будет кодироваться одним целым числом).

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия трансформации выборки***. Реализация интерпретатора (агента) данного действия требует описания в памяти классификации методов масштабирования признаков и реализации методов применения этих стратегий.

5. Разбиение выборки на обучающую, валидационную и тестовую (контрольную)

Производится разбиение всей выборки данных, на обучающую, тестовую и, в некоторых случаях, валидационную.

Валидационная выборка используется для оценки влияния изменения гиперпараметров на результат обучения и может применяться как дополнительный инструмент для этого наравне с сеточным поиском.

Разбиение проводится в соотношении 3:1:1, в процентах (60/20/20), если валидационная выборка не используется, то 80/20.

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия разбиения выборки***.

Все предыдущие этапы применялись к выборке, последующие этапы относятся к используемым моделям и.н.с.

6. Выбор класса нейросетевых методов в соответствии со сформулированной задачей

На этом этапе осуществляется выбор основной архитектуры и.н.с., которая будет использоваться при обучении. Однако, нужно отметить, что этот выбор относительно условный, то есть исследователь не ограничен использова-

нием только одного типа и.н.с. для решения задачи (как, например, сверточной сети для изображений, поскольку изображения можно обрабатывать и обычным многослойным персептроном). Речь скорее идет именно о рекомендованной архитектуре, но это не исключает использование любых других вариантов архитектур и их сочетаний в рамках одной модели).

Примерами таких рекомендаций являются:
- изображения/видео — сверхточные нейронные сети;
- временные ряды — многослойные персептроны или рекуррентные сети;
- текстовая информация — многослойные персептроны или рекуррентные сети;
- наборы характеристик некоторых объектов (например, спецификации автомобилей) — многослойный персептрон.

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия выбора класса нейросетевых методов***.

7. Формирование спецификации на входные и выходные данные

Выполняются дополнительные преобразования данных, связанные с изменением структур хранения (например, преобразование многомерного массива в одномерный, конвертация типов)

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия формирования спецификации входов и выходов и.н.с.***.

8. Выбор метода оптимизации

В рамках ПрО и.н.с. описаны следующие методы оптимизации:
- стохастический градиентный спуск (stochastic gradient descent — SGD);
- метод Нестерова;
- адаптивный градиент (adaptive gradient — AdaGrad);
- адаптивная оценка момента (adaptive moment estimation — Adam);
- среднеквадратическое распространение (root mean square propagation — RMSProp).

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия выбора метода оптимизации***.

9. Выбор минимизируемой функции ошибки

На этом этапе задается функция ошибок, которая будет минимизироваться. К примеру, MSE лучше подходит для задач регрессии и для кластеризации, CE — для классификационных задач.

Данные функции определяются следующим образом:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widetilde{Y}_i)^2$$

где *n* — размер обучающей выборки, Y_i — эталонное значение функции, $\widetilde{Y}_i$ — результат, полученный НС

$$CE = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \log(\widetilde{Y}_i) + (1 - Y_i)\log(1 - \widetilde{Y}_i))$$

(случай 2-классовой классификации)

$$CE = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{M} Y_i^c \log \widetilde{Y}_i^c$$

(случай многоклассовой классификации)

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия выбора минимизируемой функции ошибки***.

10. Начальная инициализация параметров нейронной сети

Наиболее часто используемые варианты инициализации весовых коэффициентов и порогов нейронной сети включают:
- Инициализация значениями из равномерного распределения на каком-то небольшом интервале, например, [-0.1, 0.1].
- Инициализация значениями из стандартного нормального распределения.
- Инициализация по методу Ксавье (см. *Glorot X..Under tDoTDFN-2010art*).

Применяется для предотвращения резкого уменьшения или увеличения значений выхода нейронных элементов после применения функции активации при прямом прохождении образа через глубокую нейронную сеть. Фактически инициализация этим методом осуществляется посредством выбора значений из равномерного распределения на отрезке $[-\sqrt{6}/\sqrt{n_i + n_{i+1}}, \sqrt{6}/\sqrt{n_i + n_{i+1}}]$, где n_i — это число входящих связей в данный слой, а n_i — число исходящих связей из данного слоя. Таким образом, инициализация этим методом проводится для разных слоев нейронной сети из разных отрезков.

- Инициализация, полученная из предобученной модели.

  Вариант инициализации, который предполагает использование в качестве "стартовой" модели предобученной модели, взятой из некоторого репозитория предобученных моделей, обученную самим исследователем или в процессе работы интеллектуальной системы.
- Инициализация по методу Кайминга (см. *He K..DelviDiRSHP-2015art*).

  Данный метод инициализации применяется для решения проблемы "затухающего" градиента и "взрывающегося" градиента. Производится посредством выбора значений из равномерного распределения на отрезке $[-\sqrt{2}/\sqrt{(1+a^2)fan}, \sqrt{2}/\sqrt{(1+a^2)fan}]$, где a — угол наклона к оси абсцисс для отрицательной части области определения функции активации типа ReLU (для обычной ReLU функции этот параметр равен 0), fan — параметр режима работы, который для фазы прямого распространения равен количеству входящих связей (для устранения эффекта "взрывающегося" градиента), а для фазы обратного распространения — количеству выходящих (для устранения эффекта "затухающего" градиента).

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия начальной инициализации и.н.с.***.

11. Выбора гиперпараметров и.н.с.

На практике некоторые гиперпараметры (такие как количество слоев, их типы, количество нейронов в слое) часто определяются экспериментально, в процессе итеративного поиска лучшего варианта решения задачи. Хотя способы частично автоматизировать этот процесс существуют, они все же рассчитаны на наличие некоторых предусловий проведения эксперимента, в частности интервалов изменения параметра (например, скорости обучения).

К гиперпараметрам, подбираемым на этом этапе, относятся:
- параметры обучения *и.н.с.* (скорость обучения, моментный параметр, размер мини-батча);
- архитектура модели *и.н.с.*, опирающаяся на ранее сформулированные спецификации входных и выходных данных (например, количество нейронов в определенном слое (слоях) или конфигурации целых слоев).

Нахождение оптимальных гиперпараметров может быть получено, например, использованием метода сеточного поиска, который позволяет проверить значения гиперпараметров, взятые с определенным шагом или из определенного интервала (кортежа). С помощью этого метода выбирается оптимальный набор гиперпараметров, который дает лучшие результаты, он используется для последующего дообучения. Или же, если полученные результаты являются приемлемыми, то процесс дальнейшего обучения вообще не проводится. Следует отметить затратность данного метода, так как фактически осуществляется перебор различных значений параметров обучения. Для снижения объема работы применяется метод случайного поиска.

Для оптимизации архитектуры определяются типы слоев нейронной сети, количество нейронных элементов в каждом слое, их характеристики — функция активации, для сверточных элементов — размер ядра, а также параметры padding и шаг свертки (stride). Здесь же может осуществляться оценка не только пользовательского варианта сети, но и предобученной архитектуры. Основное правило при выборе — количество параметров модели не должно превышать размер обучающей выборки. Для предобученных архитектур это ограничение снимается.

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия выбора гипперпараметров и.н.с.***. Действие использует классификацию и спецификации гиперпараметров *и.н.с.*

12. Обучение модели на обучающей выборке

Производится обучение модели до достижения выбранной точности (оценивается на тестовой выборке) или по другим заданным критериям (достижение заданного количества эпох обучения, неизменность точности на протяжении заданного количества эпох, падение точности на валидационной выборке и так далее).

Приведем классификацию алгоритмов обучения:

метод обучения и.н.с.
- ⊂ *метод*
- ⊃ *метод обучения с учителем*
  - ⇒ *explanation**:
    [***метод обучения с учителем*** — метод обучения с использованием заданных целевых переменных.]
  - ⊃ *метод обратного распространения ошибки*
    - ≔ [м.о.р.о.]
    - ⇒ *explanation**:

[м.о.р.о. использует заданный метод оптимизации и заданную функцию потерь для реализации фазы обратного распространения ошибки и изменения настраиваемых параметров и.н.с. Одним из самых распространенных методов оптимизации является метод стохастического градиентного спуска.]

⇒ *explanation**:

[Следует также отметить, что несмотря на то, что метод отнесен к методам обучения с учителем, в случае использования м.о.р.о. для обучения автокодировщиков в классических публикациях он рассматривается как метод обучения без учителя, поскольку в данном случае размеченные данные отсутствуют.]

⊃ *метод обучения без учителя*

⇒ *explanation**:

[***метод обучения без учителя*** — метод обучения без использования заданных целевых переменных(в режиме самоорганизации)]

⇒ *explanation**:

[В ходе выполнения алгоритма метода обучения без учителя выявляются полезные структурные свойства набора. Неформально его понимают как метод для извлечения информации из распределения, выборка для которого не была вручную аннотирована человеком (см. *Гудфеллоу Я..ГлубоО-2017кн*).]

В интеллектуальной среде проектирования данный этап соответствует выполнению ***действия обучения и.н.с.***. Действие обучения *и.н.с.* — действие, в ходе которого реализуется определенный метод обучения *и.н.с.* с заданными параметрами обучения *и.н.с.*, методом оптимизации и функцией потерь.

При обучении возможно возникновение следующих проблем:

- *переобучение* — проблема, возникающая при обучении *и.н.с.*, заключающаяся в том, что сеть хорошо адаптируется к паттернам входной активности из обучающей выборки, при этом теряя способность к обобщению. Переобучение возникает из-за применения неоправданно сложной модели при обучении *и.н.с.* Это происходит, когда количество настраиваемых параметров *и.н.с.* намного больше размера обучающей выборки. Возможные варианты решения проблемы заключаются в упрощении модели, увеличении выборки, использовании регуляризации (параметр регуляризации, техника dropout и так далее).
  Обнаружение переобученности сложнее, чем недообученности. Как правило, для этого применяется кросс-валидация на валидационной выборке, позволяющая оценить момент завершения процесса обучения. Идеальным вариантом является достижение баланса между переобученностью и недообученностью.
- *недообучение* — проблема, возникающая при обучении *и.н.с.*, заключающаяся в том, что сеть дает одинаково плохие результаты на обучающей и контрольной выборках. Чаще всего такого рода проблема возникает при недостаточном времени, затраченном на обучение модели. Однако это может быть вызвано и слишком простой архитектурой модели либо малым размером обучающей выборки. Соответственно решение, которое может быть принято ML-инженером, заключается в устранении этих недостатков: увеличение времени обучения, использование модели с большим числом настраиваемых параметров, увеличение размера обучающей выборки, а также уменьшение регуляризации и более тщательный отбор признаков для обучающих примеров.

методом обучения и.н.с. называется процесс итеративного поиска оптимальных значений настраиваемых параметров *и.н.с.*, минимизирующих некоторую заданную функцию потерь.

Стоит отметить, что хотя целью применения метода обучения является минимизация функции потерь, "полезность" полученной после обучения модели можно оценить только по достигнутому уровню ее обобщающей способности.

Методы обучения могут быть поделены на две большие группы — ***методы обучения с учителем*** и ***методы обучения без учителя*** (контролируемый и неконтролируемый методы обучения).

метод обучения с учителем — метод обучения с использованием заданных целевых переменных.

Одним из методов обучения с учителем является метод обратного распространения ошибки.

Приведем его описание в виде алгоритма:

Data: X — данные, E_t — желаемый отклик (метки), E_m — желаемая ошибка (в соответствии с выбранной функцией потерь)
Result: обученная нейронная сеть *Net*
инициализация весов *W* и порогов *T*;
repeat
  foreach $x \in X$, $e \in E_t$ **do**
    фаза прямого распространения сигнала: вычисляются активации для всех слоев и.н.с.;
    фаза обратного распространения ошибки: вычисляются ошибки для последнего слоя и всех предшествующих слоев;
    изменение настраиваемых параметров и.н.с. в соответствии с вычисленными ошибками;
  end
  вычисление общей ошибки E на данной эпохе;
until $E < E_m$;

метод обратного распространения ошибки использует заданный метод оптимизации и заданную функцию потерь для реализации фазы обратного распространения ошибки и изменения настраиваемых параметров и.н.с. Одним из самых распространенных методов оптимизации является метод стохастического градиентного спуска. Приведенный метод используется для реализации последовательного варианта обучения.

Следует также отметить, что несмотря на то, что метод отнесен к методам обучения с учителем, в случае его использования для обучения автокодировщиков в классических публикациях он рассматривается как метод обучения без учителя, поскольку в данном случае размеченные данные отсутствуют.

метод обучения без учителя — метод обучения без использования заданных целевых переменных (в режиме самоорганизации)

В ходе выполнения алгоритма метода обучения без учителя выявляются полезные структурные свойства набора. Неформально его понимают как метод для извлечения информации из распределения, выборка для которого не была вручную аннотирована человеком (см. *Гудфеллоу Я..ГлубоО-2017кн*). Метод обучения без учителя может рассматриваться как вспомогательный метод для начальной инициализации настраиваемых параметров и.н.с. В этом случае он является методом предобучения.

Среди методов, применяемых для оптимизации целевой функции можно выделить следующие:
- SGD (стохастический градиентный спуск). В данном методе корректировка настраиваемых параметров и.н.с. выполняется в направлении максимального уменьшения функции стоимости, то есть в направлении, противоположном вектору градиента функции потерь (см. *Хайкин С.НейроСПК-2006кн*).
- Метод Нестерова. Обучение методом стохастического градиентного спуска не редко происходит очень медленно. Импульсный метод позволяет ускорить обучение, особенно в условиях высокой кривизны, небольших, но устойчивых градиентов или зашумленных градиентов. В импульсном методе вычисляется экспоненциально затухающее скользящее среднее прошлых градиентов и продолжается движение в этом направлении. Метод Нестерова является вариантом импульсного алгоритма, в котором градиент вычисляется после применения текущей скорости (см. *Гудфеллоу Я..ГлубоО-2017кн*).
- AdaGrad: данный метод по отдельности адаптирует скорости обучения всех настраиваемых параметров и.н.с., умножая их на коэффициент, обратно пропорциональный квадратному корню из суммы всех прошлых значений квадрата градиента (см. *Duchi J..AdaptSMfOL-2011art*).
- RMSProp. Данный метод является модификацией AdaGrad, которая позволяет улучшить его поведение в невыпуклом случае путем изменения способа агрегирования градиента на экспоненциально взвешенное скользящее среднее. Использование экспоненциально взвешенного скользящего среднего гарантирует повышение скорости сходимости после обнаружения выпуклой впадины, как если бы внутри этой впадины алгоритм AdaGrad был инициализирован заново (см. *Гудфеллоу Я..ГлубоО-2017кн*);
- Adam. Данный метод можно рассматривать как комбинацию RMSProp и AdaGrad (см. *Kingma D..Adam aMfSO-2014art*). Помимо усредненного первого момента, данный метод использует усредненное значение вторых моментов градиентов.

Отметим, что успешность применения методов оптимизации зависит главным образом от знакомства пользователя с соответствующим алгоритмом (см. *Гудфеллоу Я..ГлубоО-2017кн*).

Еще одним важным компонентом, влияющим на процесс обучения, является используемая функция потерь.

функция потерь — функция, используемая для вычисления ошибки, рассчитываемой как разница между фактическим эталонным значением и прогнозируемым значением, получаемым *и.н.с.*

Среди функций потерь, используемые в качестве целевых функций для применяемого метода оптимизации, можно выделить:
- MSE — средняя квадратичная ошибка

$$MSE = \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{m}(y_i^l - e_i^l)^2$$

где y_i^l — прогноз модели, e_i^l — ожидаемый (эталонный) результат, *m* — размерность выходного вектора, *L* — объем обучающей выборки;

- BCE — бинарная кросс-энтропия (binary cross-entropy)

$$BCE = -\sum_{l=1}^{L}(e^l \log(y^l) + (1 - e^l)\log(1 - y^l))$$

где y^l — прогноз модели, e^l — ожидаемый (эталонный) результат: *0* или *1*, *L* — объем обучающей выборки;

- MCE — мультиклассовая кросс-энтропия (multiclass cross-entropy)

$$MCE = -\sum_{l=1}^{L}\sum_{i=1}^{m} e_i^l \log(y_i^l)$$

где y_i^l — прогноз модели, e_i^l — ожидаемый (эталонный результат), *m* — размерность выходного вектора.

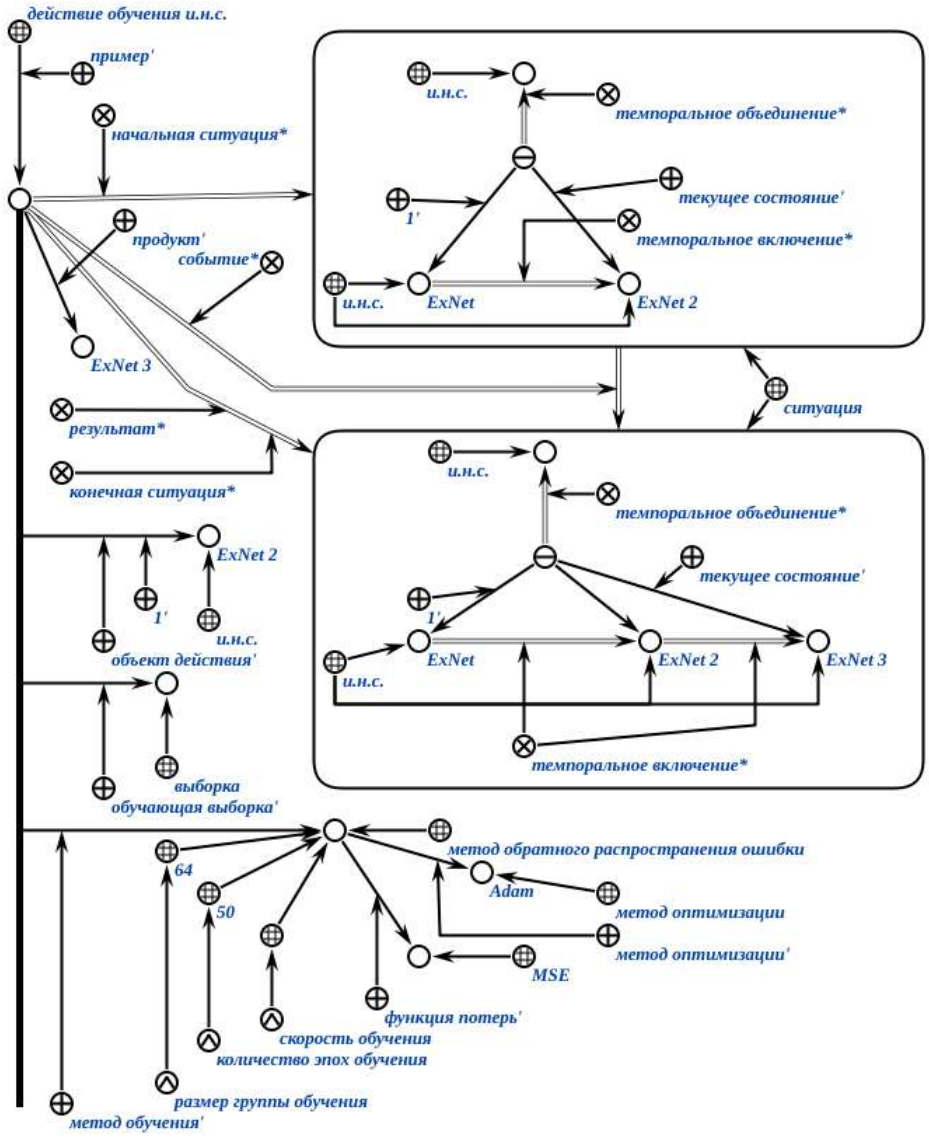
Отметим, что для бинарной кросс-энтропии в выходном слое *и.н.с.* будет находиться один нейрон, а для для мультиклассовой кросс-энтропии количество нейронов в выходном *слое и.н.с.* совпадает с количеством классов.

Для решения задачи классификации рекомендуется использовать бинарную или мультиклассовую кросс-энтропийную функцию потерь, для решения задачи регрессии рекомендуется использовать среднюю квадратичную ошибку.

Действие обучения *и.н.с.* можно проиллюстрировать следующим изображением *SCg-текст. Действие обучения и.н.с.*.

SCg-текст. Действие обучения и.н.с.

=



13. Оценка эффективности и.н.с

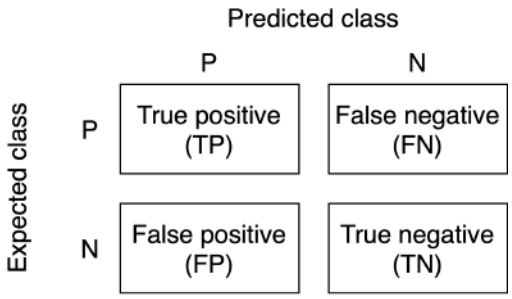
После выполнения обучения осуществляется оценка полученной модели с помощью метрик оценки качества.

Далее результат оценки может быть визуализирован с помощью матрицы ошибок (confusion matrix) и ROC-кривой.

Матрица ошибок представляется собой матрицу (см. Рисунок. Матрица ошибок), в которую помещены сведения о числе истинно-положительных, истинно-отрицательных, ложно-положительных и ложно-отрицательных предсказаниях классификатора.

Рисунок. Матрица ошибок

=



ROC-кривая (receiver operating characteristic) — это график, в котором, основываясь на заданном пороге решения классификатора, рассчитываются доли ложноположительных и истинно положительных исходов. Основываясь на ROC-кривой, высчитывается AUC-показатель (площадь под кривой), которая используется в качестве характеристики качества модели.

В интеллектуальной среде проектирования данный этап соответствует выполнению *действия оценки эффективности и.н.с.*.

Рассмотрим пример выполнения описанных этапов разработчиком для конкретной задачи — *классификации цифр из выборки рукописных цифр MNIST*:

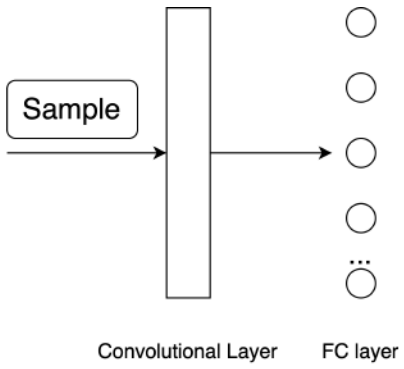
- Исходными данными задачи является: выборка из 70.000 изображений, предварительно разделенная на обучающую (60.000 изображений) и контрольную (10.000 изображений) выборки. Каждое изображение представлено двумерным массивом 28X28 чисел из интервала [0, 255], числа представляют определенный оттенок серого цвета. Помимо этого каждому изображению соответствует метка класса, соответствующая конкретной цифре от 0 до 9.

  Ставится задача: *обучить модель, которая будет принимать на вход двумерный массив данных и возвращать метку класса, соответствующей распознанной цифре*.

  Таким образом, тип решаемой задачи — **классификационная**, природа данных задачи — **изображения**.
- В рассматриваемой выборке отсутствуют аномалии, ошибочные данные, признаки с отсутствующими значениями.
- В рассматриваемой задаче отсутствуют несодержательные признаки.
- В качестве метода предобработки данных используем масштабирование признаков, а именно нормализацию на отрезок [0, 1].
- Выполним разбиение обучающей части данных на обучающую и валидационную выборки в соотношении 4:1 (48.000 в обучающей и 12.000 в валидационной).
- Так как выборка включает в себя изображения, будем использовать сверточную нейронную сеть.
- Не требуется.
- В качестве оптимизационного алгоритма будем использовать метод стохастического градиентного спуска (SGD).
- Так как решается задача классификации, выберем в качестве минимизируемой функции кросс-энтропийную функцию потерь.
- В качестве начальной инициализации будем использовать инициализацию по методу Кайминга.
- На предыдущих этапах было определено, что для решения задачи будет использоваться сверточная нейронная сеть. При использовании one-hot кодирования в последнем полносвязном слое будет 10 нейронов по числу классов в задаче.

  Для упрощения будем использовать архитектуру, изображенную на *Рисунок. Архитектура и.н.с., решающая задачу классификации цифр*, не содержащую промежуточные слои.

Рисунок. Архитектура и.н.с., решающая задачу классификации цифр
=



Для нахождения оптимального набора гиперпараметров будем применять метод случайного поиска.

Перечислим кортежи, из которых будут сэмплироваться гиперпараметры:

- Скорость обучения — (0.9, 0.1, 0.01, 0.001);
- Количество нейронов в сверточном слое — (5, 10, 15, 20);
- Размер ядра свертки — (3, 5, 7, 9);
- Моментный параметр — (0, 0.5, 0.9);

- Размер мини-батча — (16, 32, 64, 128).

После определения данных параметров и оценки эффективности работы алгоритма, получим следующую таблицу:

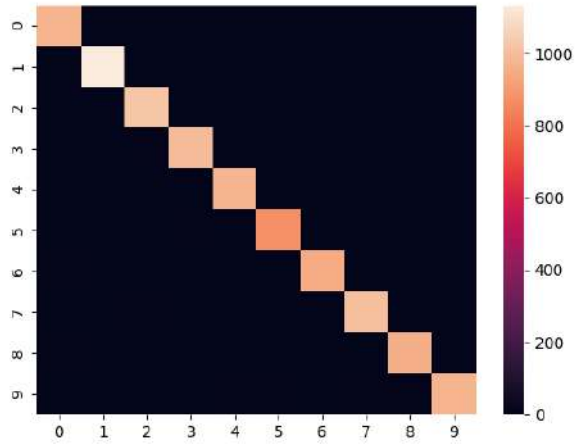
Таблица. Результаты решения задачи
(используемые сокращения: mbs — mini-batch size, ks — kernel size, lr — learning rate, cnc — convolutional neurons count, acc — accuracy, it — iterations count)
=

#	mbs	ks	lr	momentum	cnc	acc	it
1	128	3	0.001	0.5	10	0.9033	10
2	64	9	0.9	0	15	0.1039	1
3	32	3	0.01	0.5	20	0.9741	10
4	32	7	0.01	0.5	15	0.9794	10
5	16	9	0.001	0.5	20	0.9189	2
6	64	3	0.1	0.5	10	0.9736	10
7	64	7	0.001	0.9	15	0.9007	1
8	32	9	0.1	0.5	5	0.9806	10
9	128	5	0.1	0.5	20	0.98	10
10	32	9	0.01	0.9	5	0.9806	10
11	128	3	0.001	0.9	10	0.893	1
12	32	5	0.9	0.9	20	0.1008	1
13	16	9	0.9	0.5	20	0.0976	1
14	32	7	0.9	0.9	15	0.0932	1
15	128	5	0.01	0.5	20	0.9197	2
16	16	3	0.001	0.5	10	0.904	1
17	16	9	0.001	0	20	0.8866	1
18	128	9	0.1	0.5	5	0.9793	10
19	128	3	0.001	0	10	0.6697	1
20	16	3	0.1	0	15	0.9729	4
21	32	7	0.9	0.5	15	0.1048	1
22	128	7	0.9	0	15	0.1113	1
23	64	9	0.01	0.5	10	0.9482	2
24	16	7	0.9	0	20	0.0985	1
25	16	3	0.1	0.5	5	0.9558	2
26	64	7	0.01	0.9	15	0.9839	10
27	16	7	0.1	0	10	0.9836	10
28	16	5	0.01	0	20	0.9608	2
29	16	5	0.01	0.9	20	0.9847	10
30	32	5	0.01	0.5	15	0.9532	2

Можно заметить, что лучший результат (acc = 0.9839) по обобщающей способности на валидационной выборке был получен при следующих параметрах: mbs = 64, ks = 7, lr = 0.01, momentum = 0.9, cnc = 15.

- В качестве критерия останова нами был выбран самый простой критерий по достижению заданного количества эпох обучения. Дообучение не проводилось, для оценки обобщающей способности использовалась модель, полученная после выполнения процедуры подбора гиперпараметров. Обобщающая способность на тестовой выборке составила **0.9853**, то есть **98.53%**.
- Построив матрицу ошибок на основании обученной модели и тестовой выборки, получим результат, проиллюстрированный на рис. Рисунок. Матрица ошибок для задачи MNIST

Рисунок. Матрица ошибок для задачи MNIST
=



Мы получили матрицу с явно выраженным диагональным преобладанием, таким образом полученная модель делает относительно небольшое число ошибок.

Исходя из анализа этапов построения и.н.с., которые выполняют разработчики, можно вывести следующую классификацию действий по построению и.н.с.:

действие по построению и.н.с.
⇒ *декомпозиция**:
{• *действие по обработке выборки*
⇒ *декомпозиция**:
{• *действие поиска подходящей обучающей выборки*
• *действие формирования требований к обучающей выборке*
• *действие очистки выборки*
• *действие выявления содержательных признаков*
• *действие трансформации выборки*
• *действие разбиения выборки*
}
• *действие по проектированию и.н.с.*
⇒ *декомпозиция**:
{• *действие выбора класса нейросетевых методов*
• *действие формирования спецификации входов и выходов и.н.с.*
}
• *действие обучения и.н.с.*
⇒ *декомпозиция**:
{• *действие выбора метода оптимизации*
• *действие выбора минимизируемой функции ошибки*
• *действие начальной инициализации и.н.с.*
• *действие выбора гиперпараметров и.н.с.*
• *действие обучения и.н.с.*
• *действие оценки эффективности и.н.с.*
}
}

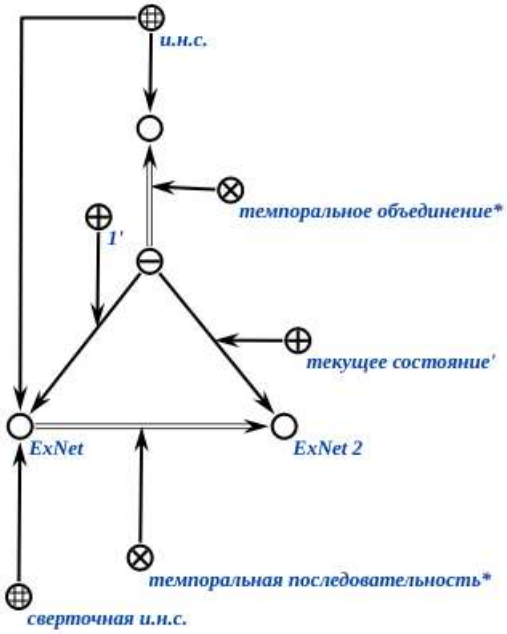
Реализация интерпретатора описанных в данной главе действий по построению *и.н.с.* и описания в базе знаний экспертных знаний разработчиков *и.н.с.* (а значит реализация интеллектуальной среды проектирования *и.н.с.*) позволит автоматически, исходя из описания задачи, генерировать нейросетевые методы в памяти *ostis-системы*, что является одним из ключевых направлений дальнейшего развития конвергенции и интеграции и.н.с. с базами знаний.

Так как в результате действий по построению *и.н.с.* объект этих действий, конкретная *и.н.с.*, может существенно меняться (меняется конфигурация сети, ее весовые коэффициенты), то *и.н.с.* представляется в базе знаний как темпоральное объединение всех ее версий. Каждая версия является *и.н.с.* и темпоральной сущностью. На множестве

этих темпоральных сущностей задается темпоральная последовательность с указанием первой и последней версии. Для каждой версии описываются специфичные знания.

Общие для всех версий знания описываются для *и.н.с.*, являющейся темпоральным объединением всех версий (рисунок SCg-текст. Темпоральность нейронной сети)

SCg-текст. Темпоральность нейронной сети
=



Далее более подробно рассмотрим действие по обучению *и.н.с.* и сделаем обзор основных методов, применяемых для обучения.

Заключение к Главе 3.6.

В главе описан подход к *интеграции и конвергенции искусственных нейронных сетей с базами знаний* в *интеллектуальных компьютерных системах нового поколения* с помощью представления и интерпретации *искусственной нейронной сети* в *базе знаний*.

Описаны *Синтаксис, Денотационная и Операционная семантика Языка представления нейросетевых методов в базах знаний*, который позволяет представить и интерпретировать в памяти интеллектуальной системы любую *и.н.с.* Наличие такого языка порождает семантическую совместимость нейросетевого метода с другими методами, представленными в памяти системы, что позволяет анализировать саму *и.н.с.* и этапы ее работы любыми другими методами системы.

Так же наличие языка представления нейросетевых методов позволяет описывать в памяти системы экспертные знания разработчиков *и.н.с.* В главе приведены этапы построения *и.н.с.*, которые выполняют разработчики *и.н.с.* На основании этих этапов, с целью проектирования интеллектуальной среды построения *нейросетевых методов*, в *базе знаний* были классифицированы и описаны действия по построению *и.н.с.*

Проектирования и реализация интеллектуальной среды построения *и.н.с.* в *базе знаний* системы является одним из двух основных направлений дальнейшего развития работу по конвергенции и интеграции и.н.с. с базами знаний.

Вторым основным направлением является разработка подхода к обработке фрагментов *базы знаний* с помощью *и.н.с.*, для чего необходимо разработать универсальный алгоритм взаимно-однозначного соответствия фрагментов базы знаний и входных векторов *и.н.с.* Язык представления знаний способен представить любое знание. Наличие в системе нейросетевого метода, способного принимать на вход фрагменты знаний, позволит решить новые, слабо изученные классы задач.

Часть 4.

Онтологические модели интерфейсов интеллектуальных компьютерных систем нового поколения

⇒ *аннотация**:

[Принципы организации и структура *интерфейсов интеллектуальных компьютерных систем нового поколения*. Подход к реализации *естественно-языковых интерфейсов интеллектуальных компьютерных систем нового поколения*, *аудио* и *речевых интерфейсов интеллектуальных компьютерных систем нового поколения*. Построение и использование трехмерного представления в различных задачах прикладных *интеллектуальных компьютерных систем*, а также соответствующих систем позиционирования и ориентации в пространстве, описание такого представления, а также принципов его построения.**]**

⇒ *подраздел**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*

Глава 4.1.
Общие принципы организации интерфейсов ostis-систем

⇒ *автор**:
- *Садовский М. Е.*

⇒ *аннотация**:
[В главе рассмотрены принципы организации *интерфейсов интеллектуальных компьютерных систем нового поколения*. Ключевыми свойствами *интерфейсов интеллектуальных компьютерных систем нового поколения* являются адаптивность и мультимодальность, что обеспечивает переход от парадигмы грамотного пользователя к парадигме равноправного сотрудничества пользователя с интеллектуальной системой, что позволяет повысить эффективность человеко-машинного взаимодействия. В главе рассматривается подход к обеспечению указанных свойств на основе онтологической модели интерфейса и онтологической модели процесса проектирования интерфейсов.]

⇒ *подраздел**:
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*
- *§ 4.1.3. Интерфейсные действия пользователей ostis-систем*
- *§ 4.1.4. Сообщения, входящие в ostis-систему и выходящие из нее*
- *§ 4.1.5. Действия и внутренние агенты пользовательского интерфейса ostis-системы*

⇒ *ключевое понятие**:
- *интерфейс*
- *физический интерфейс*
- *программный интерфейс*
- *пользовательский интерфейс*
- *интерфейс ostis-систем*
- *адаптивный интерфейс*
- *интеллектуальный интерфейс*
- *мультимодальный интерфейс*
- *компонент пользовательского интерфейса*
- *интерфейсное действие пользователя*
- *сообщение*
- *решатель задач пользовательского интерфейса ostis-систем*
- *интерпретатор sc-моделей пользовательских интерфейсов*
- *интерпретатор пользовательских действий*

⇒ *ключевое отношение**:
- *инициируемое пользовательским интерфейсом действие**

⇒ *ключевое знание**:
- *Предметная область пользовательских интерфейсов*
- *Предметная область компонентов интерфейса*

⇒ *библиографическая ссылка**:
- *Фомина Т.А..ПроекАИИС-2020cm*
- *Hussain J..ModelBAUIBoCaUEE-2018art*
- *Валеев С.С..ПострАИвСР-2018cm*
- *Montero M..Adapt aAWBES-2005bk*
- *Schlungbaum E.IndivUIaMU-1997art*
- *Brdnik S..IntelUIaTEaS-2022art*
- *Volkel S..What iIiIUI-2020art*
- *Ehlert P.IntelUIIaS-2003bk*
- *Sadouski M.SemanDoaAUI-2022art*
- *Hitz M..UsingAOftAGoUIfDBA-2016art*
- *Liu B..DerivUIfOaMBA-2005art*
- *Gaulke W..UsingPOtLM-2015art*
- *Грибова В.В..АвтомРПИсД-2011cm*

- *Грибова В.В..ГенерКПИУ-2005ст*
- *Abrams M..UIMLaA-1999art*
- *Paterno F..MARIAaUDMALL-2009art*
- *Limbourg Q..UsiXM aUIDLSMLoI-2004art*
- *Puerta A..BeyonDMfAUIG-1994art*
- *Heckmann D..tUserMaCOGUMO-2001art*
- *Paulheim H..UI20AFOoUIaI-2013art*
- *Грибова В.В..Онтол дРиГАП-2022ст*
- *Kong J..Desig oHCAM-2011art*
- *Boriskin A.S..OntolBDoISU-2017art*
- *Корончик Д.Н.СеманТКПП-2011ст*
- *Корончик Д.Н.УнифиСМПИ-2013ст*
- *Sadouski M.OntolAttBoSM-2021art*
- *Sadouski M.tSrtuct oNICSI-2022art*
- *Голенков В.В..СтандОТОП-2021кн*
- *Myers B.aBriefHoHCIT-2001art*

Введение в Главу 4.1.

Организация взаимодействия пользователей с компьютерными системами (в том числе и с интеллектуальными компьютерными системами) оказывает существенное влияние на эффективность автоматизации человеческой деятельности, пользовательский опыт и уровень удовлетворенности пользователей.

Одним из ключевых свойств интеллектуальных компьютерных систем нового поколения является их **интероперабельность** — способность к эффективному взаимодействию. Такие системы являются автономными и самодостаточными субъектами деятельности наравне с человеком. Однако, в основе современной организации взаимодействия пользователя с компьютерной системой лежит парадигма грамотного пользователя, который знает, как управлять системой и несет полную ответственность за качество взаимодействия с ней. Многообразие форм и видов интерфейсов приводит к необходимости пользователя адаптироваться к каждой конкретной системе, обучаться принципам взаимодействия с ней для решения необходимых ему задач.

На современном этапе развития Искусственного интеллекта для повышения эффективности взаимодействия необходим переход от парадигмы грамотного управления используемым инструментом к парадигме **равноправного сотрудничества, партнерскому взаимодействию** интеллектуальной компьютерной системы со своим пользователем. Дружественность пользовательского интерфейса должна заключаться в адаптивности системы к особенностям и квалификации пользователя, исключении любых проблем для пользователя в процессе диалога с интеллектуальной компьютерной системой, в перманентной заботе о совершенствовании коммуникационных навыков пользователя. Следовательно, необходимо отойти от привычной адаптации пользователя к системе (путем обучения ее использованию) в сторону адаптации самого интерфейса под цели, задачи и характеристики конкретного пользователя в режиме реального времени (см. *Фомина Т.А..ПроекАИИС-2020ст*).

§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов

интерфейс — совокупность технических, программных и методических (протоколов, правил, соглашений) средств, обеспечивающих обмен информацией между пользователем и устройствами и программами, а также между устройствами и другими устройствами и программами. В широком смысле слова, это способ (стандарт) взаимодействия между объектами. Интерфейс в техническом смысле слова задает параметры, процедуры и характеристики взаимодействия объектов.

интерфейсы бывают разных видов. Они отличаются по характеру систем, которые взаимодействуют между собой; реализацией и функциями.

Вне зависимости от типа интерфейса, взаимодействие компьютерной системы с окружающей средой происходит при помощи *сенсоров* и *эффекторов*. Ключевая задача *интерфейса* — обеспечение эффективного взаимодействия с внешними субъектами (пользователями, другими *ostis-системами*, другими традиционными компьютерными системами).

Принято выделять следующие виды *интерфейсов*:
- *физический интерфейс*;
- *программный интерфейс*;

- *пользовательский интерфейс*.

физический интерфейс позволяет преобразовать сигналы и передать их от одного компонента оборудования к другому и определяется набором электрических связей и характеристиками сигналов.

программный интерфейс предназначен для обмена информацией между компонентами вычислительной системы и задает набор необходимых процедур, их параметров и способов обращения.

пользовательский интерфейс — один из наиболее важных компонентов компьютерной системы. Представляет собой совокупность аппаратных и программных средств, обеспечивающих обмен информацией между пользователем и компьютерной системой.

В контексте данной главы основное внимание будет уделено *пользовательским интерфейсам*, другие виды интерфейсов являются объектами будущих исследований.

Ключевыми проблемами современных пользовательских интерфейсов являются:
- необходимость пользователя обучаться принципам взаимодействия с каждой конкретной системой;
- отсутствие партнерского взаимодействия между пользователем и системой (система является объектом управления со стороны пользователя), что приводит к необходимости пользователя быть постоянным инициатором взаимодействия;
- отсутствие адаптации системы к особенностям каждого конкретного пользователя и окружающей среды для максимально комфортного взаимодействия пользователя с системой.

интерфейс интеллектуальных компьютерных систем нового поколения должен обеспечивать взаимодействие с пользователем на равноправной основе, уметь адаптироваться к его особенностям, а также **воспринимать различные типы** ввода информации. Для организации такого взаимодействия используются термины *адаптивного*, *интеллектуального* и *мультимодального интерфейса*.

адаптивный интерфейс — пользовательский интерфейс, который изменяется на основе потребностей пользователя или контекста использования.

Как правило, контекст использования состоит из знаний о пользователе, платформе и среде, как показано на рисунке *Рисунок. Контекст использования системы* (см. *Hussain J..ModelBAUIBoCaUEE-2018art*).

Рисунок. Контекст использования системы
=

Настройка функциональных возможностей и параметров интерфейса может осуществляться либо вручную самим пользователем, либо автоматически системой на основании имеющейся информации о пользователе. Таким образом, следует различать адаптивные и адаптируемые системы, эти термины не являются синонимами, хотя в литературе довольно часто можно встретить подмену данных понятий (см. *Валеев С.С..ПострАИвСР-2018ст*).

В адаптируемых системах любая адаптация является предопределенной и может изменяться пользователями перед запуском системы. В адаптивных же системах, напротив, любая адаптация является динамической, то есть происходит в то же время, когда пользователь взаимодействует с системой, и зависит от поведения пользователя. Но система также может быть адаптируемой и адаптивной одновременно (см. *Montero M..Adapt aAWBES-2005bk*). Недостаток ручного редактирования интерфейса заключается в необходимости пользователя быть достаточно хорошо знакомым, как с самой системой, так и со средствами, позволяющими изменять ее *интерфейс*.

Также в литературе можно встретить термин *“адаптированный интерфейс”*.

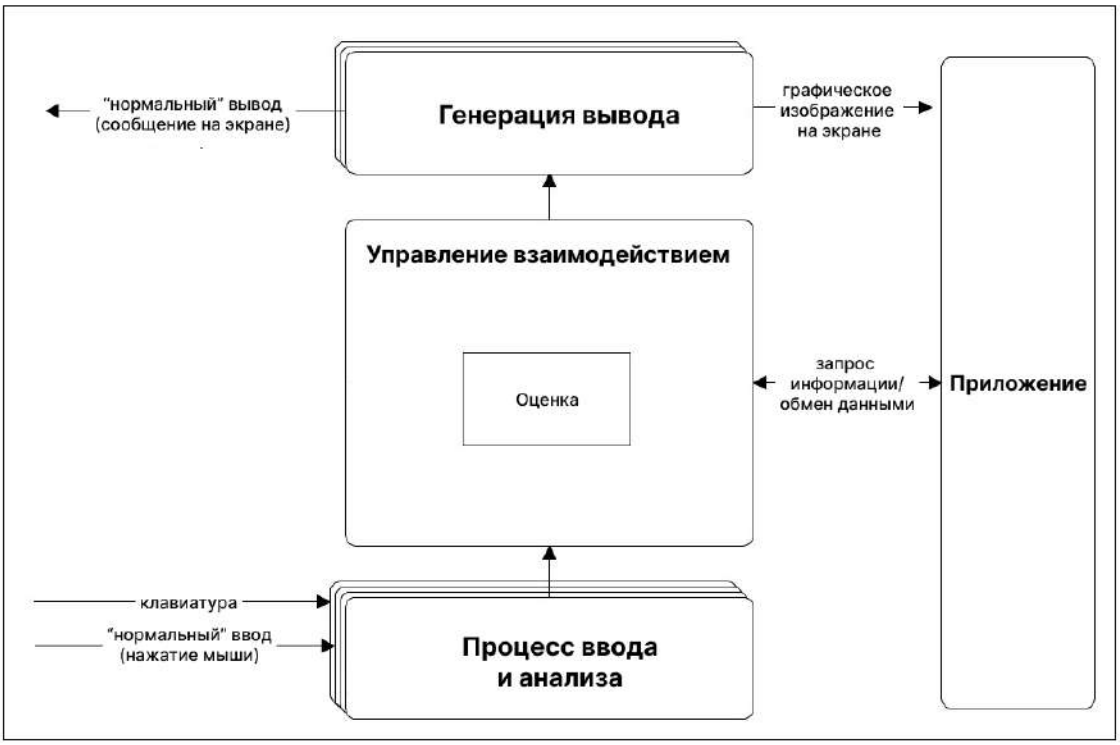
адаптированный интерфейс — это *пользовательский интерфейс*, который адаптирован к конечному пользователю при проектировании и не изменяется во время эксплуатации системы (см. *Schlungbaum E.IndivUIaMU-1997art*).

интеллектуальный интерфейс — пользовательский интерфейс, который может предположить дальнейшие действия пользователей и представить информацию на основе этого предположения. Можно заметить, что понятия интеллектуальный и адаптивный интерфейс имеют отличия. Однако, в различных статьях эти понятия рассматриваются как синонимы (см. *Brdnik S..IntelUIaTEaS-2022art*, *Volkel S..What iIiIUI-2020art*, *Ehlert P.IntelUIIaS-2003bk*).

мультимодальный интерфейс — пользовательский интерфейс, предназначенный для обработки двух или более комбинированных режимов пользовательского ввода, таких как речь, перо, касание, ручные жесты и взгляд, скоординированным образом с выводом мультимедийной системы.

Взаимодействие с большей частью традиционных компьютерных систем происходит с помощью клавиатуры и мыши (тачпада, стилуса). Пользовательский интерфейс таких систем, как правило, не хранит информацию о модели пользователя, историю его действий и так далее. Традиционный пользовательский интерфейс также не содержит модуль адаптации. На рисунке *Рисунок. Структура традиционного пользовательского интерфейса* представлена структура традиционного пользовательского интерфейса (см. *Sadouski M.SemanDoaAUI-2022art*).

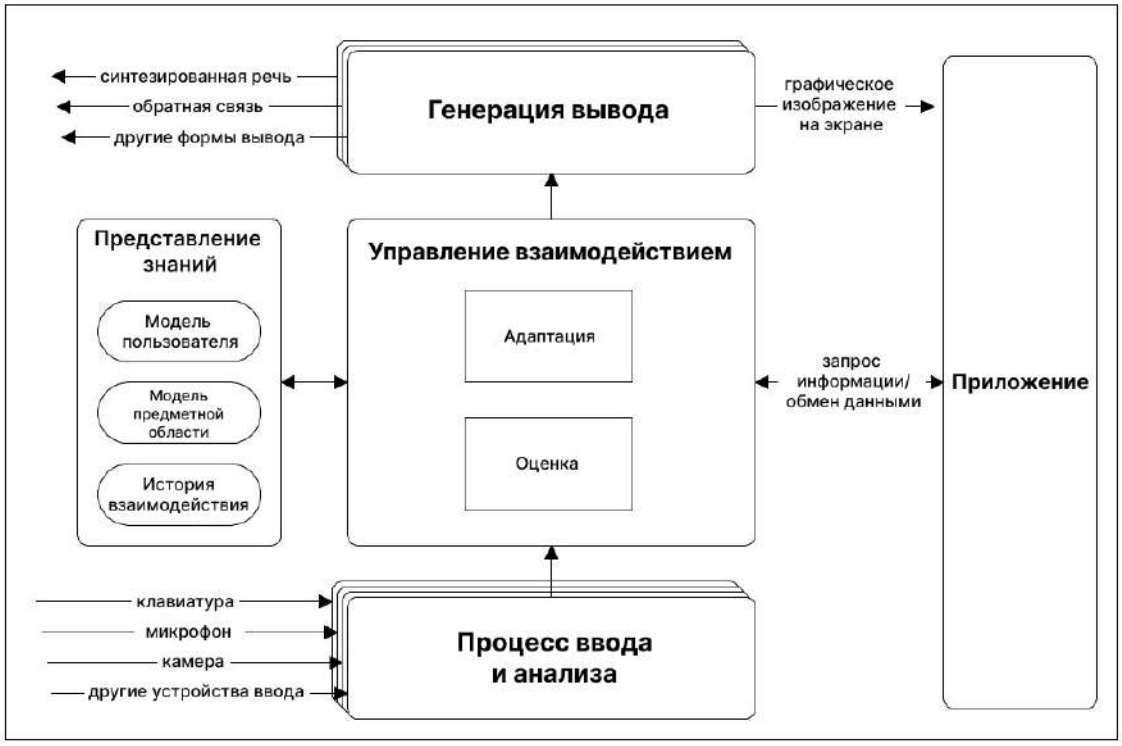
Рисунок. Структура традиционного пользовательского интерфейса
=



Общая архитектура адаптивного интеллектуального мультимодального пользовательского интерфейса в свою очередь, как правило, выглядит так, как представлено на рисунке *Рисунок. Структура адаптивного интеллектуального мультимодального пользовательского интерфейса*.

Рисунок. Структура адаптивного интеллектуального мультимодального пользовательского интерфейса

=



Среди современных средств создания адаптивных пользовательских интерфейсов можно выделить следующие средства, представленные на рисунке *Рисунок. Существующие средства создания адаптивных пользовательских интерфейсов* (см. *Hussain J..ModelBAUIBoCaUEE-2018art*).

Рисунок. Существующие средства создания адаптивных пользовательских интерфейсов
=

Легенда ● Полностью ◑ Частично ○ Нет ◌ Не определено	Мультимодальный источник данных	Направленная и косвенная адаптация	Подход к моделированию	Контекст			Вспомогательный инструмент	Аспекты адаптации			Отзыв пользователя при адаптации
				Пользователь	Платформа	Окружающая среда		Отображение	Навигация	Контент	
3-Layer Architecture	◑	◑	◑	●	●	●		◌	◌	◌	◑
CAMELEON-RT	●	●	◌	◑	◑	◑	●	◌	◌	◌	○
CEDAR	●	●	●	●	●	●	●	◑	●	●	●
Malai	◑	●	◑	○	●	○	●	●	●	◑	○
TRIPLET	◑	◑	●	●	●	●	◌	●	●	●	◑
Egoki system	◑	◑	●	●	●	○	○	●	●	●	◑
SUPPLE	◑	●	●	●	●	○	●	●	◑	○	○
MyUI	●	◑	●	●	●	●	●	●	●	◑	◑
Roam framework	◌	◑	●	○	●	○	◌	●	○	○	○
XMobile	◌	◑	●	○	●	○	◑	●	◑	○	◌
AUI-UXA	●	●	●	●	●	●	●	●	●	●	●

Вне зависимости от средств создания *адаптивных интеллектуальных мультимодальных пользовательских интерфейсов* такие системы должны эффективно хранить и обрабатывать знания о пользователе, взаимодействии с ним и другую необходимую информацию. Большинство таких систем используют онтологическую модель для хранения информации для адаптации пользовательского интерфейса (см. *Hitz M..UsingAOftAGoUIfDBA-2016art* , *Liu B..DerivUIfOaMBA-2005art*, *Gaulke W..UsingPOtLM-2015art*, *Грибова В.В..АвтомРПИсД-2011ст*, *Грибова В.В..ГенерКПИУ-2005ст*), а также декларативные языки описания *пользовательского интерфейса* (см. *Abrams M..UIMLaA-1999art* , *Paterno F..MARIAaUDMALL-2009art* , *Limbourg Q..UsiXM aUIDLSMLoI-2004art* , *Puerta A..BeyonDMfAUIG-1994art*). Именно онтологический подход позволяет:

- создать наиболее полное унифицированное описание различных аспектов пользовательского интерфейса;
- легко интегрировать различные аспекты *пользовательского интерфейса*;
- упростить повторное использование модели интерфейса.

В рамках онтологического подхода принято выделять онтологии и предметные области. База знаний адаптивного интеллектуального мультимодального интерфейса должна включать как минимум следующие предметные области:

- *Предметную область и онтологию модели пользователя*;
- *Предметную область и онтологию компонентов интерфейса*;
- *Предметную область и онтологию интерфейсных действий*;
- *Предметную область и онтологию контекста использования*.

Среди существующих *онтологий модели пользователя* можно выделить онтологию GUMO (см. *Heckmann D..tUserMaCOGUMO-2001art*), в рамках которой выделяют:

- физиологическое состояние — может измениться за секунды;
- психическое состояние — может измениться за минуты;
- эмоциональное состояние — может измениться за часы;
- характер — может измениться за месяцы;
- личность — может измениться за годы;
- демография — обычно не может измениться.

В работе *Paulheim H..UI20AFOoUIaI-2013art* рассматривается *онтология компонентов интерфейса*, на верхнем уровне которой рассматриваются следующие типы компонентов:

- компонент пользовательского интерфейса для отображения;
- декоративный компонент пользовательского интерфейса;

- интерактивный компонент пользовательского интерфейса;
- компонент ввода данных;
- компонент для манипуляции отображением;
- компонент для запуска операций;
- контейнер;
- окно;
- модальное окно;
- немодальное окно.

В данную онтологию также можно включить классы свойств компонента, которые определяют оформление внешнего вида интерфейсных элементов, начиная от простых, таких как шрифт, цвет, размер элементов, до составных, содержащих наборы интерфейсных решений (см. *Грибова В.В..Онтол дРиГАП-2022ст*).

Классификация *интерфейсных действий* представлена в *Paulheim H..UI20AFOoUIaI-2013art* и содержит следующие основные классы:
- действие мышью;
- действие речью;
- действие осязания;
- действие прикосновения.

Онтология контекста использования рассматривается в работе *Kong J..Desig oHCAM-2011art* и описывает:
- Статус пользователей:
  - Движение (стояние, сидение, ходьба);
  - Возможность слушать (да, нет);
  - Возможность печатать (да, нет);
  - Возможность говорить (да, нет);
  - Возможность читать (да, нет);
- Естественная среда:
  - Освещение (яркий, умеренно освещенный, темный);
  - Шум (шумный, тихий);
  - Ветер (сильный, слабый, безветрие);
  - Погода (солнечно, облачно, дождливо);
  - Температура (жарко, тепло, холодно);
  - Местоположение (в офисе, в аэропорту, на улице, в библиотеке, дома, в торговом центре);
- Особенности устройства:
  - Размер экрана (большой, маленький);
  - Тип экрана (монохромный, цветной);
  - Клавиатура (большая, маленькая, виртуальная).

Для управления взаимодействием пользователя с системой принято использовать интеллектуальные агенты.

интеллектуальный агент способен выполнять гибкое автономное действие для достижения своих целей. Согласно данному определению, гибкость означает:
- Реактивность: интеллектуальные агенты способны воспринимать свою среду и реагировать в своевременном режиме на изменения в ней, чтобы удовлетворить свои цели проектирования;
- Проактивность: интеллектуальные агенты способны проявлять целенаправленное поведение, инициируя действия для достижения своих целей проектирования;
- Социальная способность: интеллектуальные агенты способны взаимодействовать с другими агентами (и, возможно, людьми) с целью удовлетворения своих целей проектирования.

интеллектуальные агенты направлены на единственную цель, но обладают большим знанием о рассуждении в пределах своей деятельности. Умение использовать другие ресурсы (других агентов), предпочтения пользователя или клиента и другие способности являются признаками интеллектуального агента.

В результате проведенного анализа можно сделать следующие выводы:
- Для перехода к *парадигме равноправного сотрудничества пользователя и системы* интерфейсы должны быть адаптивными, интеллектуальными и мультимодальными. Существующие решения позволяют проектировать такие интерфейсы, однако имеют ряд недостатков, которые будут представлены далее.
- Структура *интеллектуальных интерфейсов* включает базу знаний, модуль управления взаимодействием пользователя с системой.
- При проектировании базы знаний активно применяется онтологический подход и уже реализованы некоторые онтологии, которые используются при проектировании интеллектуальных интерфейсов.
- Модуль управления взаимодействием пользователя с системой, как правило, реализуется на основе многоагентного подхода.

Среди недостатков существующих решений можно выделить:
- Существующие решения, как правило, предусматривают вопросно-ответный принцип взаимодействия.
- Актуальной остается проблема совместимости интеллектуального интерфейса с интеллектуальной системой, для которой он создается, в силу различий используемых средств и методов при проектировании и реализации.
- Актуальной остается проблема совместимости компонентов интеллектуального интерфейса (база знаний и модуль управления взаимодействием) между собой.

Для устранения недостатков существующих решений предлагается использовать онтологический подход на основе семантической модели при проектировании и реализации *адаптивного интеллектуального мультимодального пользовательского интерфейса*, принципы которого будут рассмотрены далее.

§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем

Для организации *интерфейса ostis-систем* предлагаются следующие принципы:
- Под *пользовательским интерфейсом ostis-системы* понимается специализированная *ostis-система*, ориентированная на решение интерфейсных задач, и имеющая в своем составе базу знаний и решатель задач пользовательского интерфейса ostis-системы (см. *Boriskin A.S..OntolBDoISU-2017art*). Таким образом обеспечивается совместимость *ostis-системы* и ее *пользовательского интерфейса*.
- Для решения задачи построения пользовательского интерфейса в базе знаний *пользовательского интерфейса ostis-системы* содержится sc-модель *компонентов пользовательского интерфейса*, *интерфейсных действий пользователей*, а также классификация *пользовательских интерфейсов* в целом.
- Решатель задач *пользовательского интерфейса ostis-системы* основан на многоагентном подходе, а сами агенты имеют возможность инициирования действий и сообщений пользователю и другим агентам.
- При проектировании *пользовательского интерфейса ostis-системы* используется компонентный подход, который предполагает представление всего интерфейса приложения в виде отдельных специфицированных компонентов, которые могут разрабатываться и совершенствоваться независимо (см. *Корончик Д.Н.СеманТКПП-2011ст*, *Корончик Д.Н.УнифиСМПИ-2013ст*).
- Каждый компонент *пользовательского интерфейса ostis-системы* является внешним отображением определенного элемента из базы знаний, что позволяет использовать их в качестве аргументов пользовательских команд и правильно трактовать прагматику и семантику объектов интерфейсной деятельности, легко изменять интерфейс системы даже во время ее эксплуатации, позволяет пользователю задавать системе вопросы касательно любого из компонентов интерфейса.
- Модель *пользовательского интерфейса ostis-системы* строится независимо от реализации платформы интерпретации такой модели, что позволяет легко переносить разработанную модель на разные платформы.
- Описание модели базы знаний и решателя задач *пользовательского интерфейса ostis-системы* предлагается осуществлять на основе универсального унифицированного языка представления знаний, что обеспечит совместимость между этими компонентами.

пользовательский интерфейс ostis-системы является адаптивным интеллектуальным мультимодальным пользовательским интерфейсом, структура которого представлена на рисунке *Рисунок. Структура пользовательского интерфейса ostis-системы* (см. *Sadouski M.OntolAttBoSM-2021art*, *Sadouski M.tSrtuct oNICSI-2022art*).

Рисунок. Структура пользовательского интерфейса ostis-системы
=

Предметная область пользовательских интерфейсов представляет собой формализованную типологию пользовательских интерфейсов. Пример фрагмента данной предметной области в базе знаний пользовательского интерфейса будет выглядеть следующим образом.

интерфейс
:= [совокупность технических, программных и методических (протоколов, правил, соглашений) средств, обеспечивающих обмен информацией между пользователем и устройствами и программами, а также между устройствами и другими устройствами и программами. В широком смысле слова, это способ (стандарт) взаимодействия между объектами. Интерфейс в техническом смысле слова задает параметры, процедуры и характеристики взаимодействия объектов]

интерфейс
⇒ *разбиение**:
{• *пользовательский интерфейс*
:= [один из наиболее важных компонентов компьютерной системы. Представляет собой совокупность аппаратных и программных средств, обеспечивающих обмен информацией между пользователем и компьютерной системой.]
• *программный интерфейс*
:= [система унифицированных связей, предназначенных для обмена информацией между компонентами вычислительной системы. Программный интерфейс задает набор необходимых процедур, их параметров и способов обращения.]
• *физический интерфейс*
:= [устройство, преобразующее сигналы и передающее их от одного компонента оборудования к другому. Физический интерфейс определяется набором электрических связей и характеристиками сигналов.]
}

адаптивный интерфейс
:= [*пользовательский интерфейс*, который изменяется на основе потребностей пользователя или контекста]

интеллектуальный интерфейс
:= [*пользовательский интерфейс*, который может предположить дальнейшие действия пользователей и представить информацию на основе этого предположения]

мультимодальный интерфейс
:= [*пользовательский интерфейс*, предназначенный для обработки двух или более комбинированных режимов пользовательского ввода, таких как речь, перо, касание, ручные жесты и взгляд, скоординированным образом с выводом мультимедийной системы]

пользовательский интерфейс
⊃ *командный пользовательский интерфейс*
:= [*пользовательский интерфейс*, при котором обмен информацией между компьютерной системой и пользователем осуществляется путем написания текстовых инструкций или команд]
⊃ *WIMP-интерфейс*
:= [Window, Image, Menu, Pointer — интерфейс]
:= [Окно, Образ, Меню, Указатель — интерфейс]
:= [*пользовательский интерфейс*, при котором обмен информацией между компьютерной системой и пользователем осуществляется в форме диалога при помощи окон, меню и других элементов управления]
⊃ *пользовательский интерфейс ostis-системы*
⊃ *SILK-интерфейс*
:= [Speech, Image, Language, Knowledge — интерфейс]
:= [Речь, Образ, Язык, Знание — интерфейс]
:= [*пользовательский интерфейс*, наиболее приближенный к естественной для человека форме общения. Компьютерная система находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд преобразуется в понятную человеку форму, например, в естественно-языковую форму или изображение.]
⊃ *естественно-языковой интерфейс*
:= [SILK-интерфейс, обмен информацией между компьютерной системой и пользователем в котором происходит за счет диалога. Диалог ведется на одном из естественных языков]
⊃ *речевой интерфейс*
:= [SILK-интерфейс, обмен информацией в котором происходит за счет диалога, в процессе которого компьютерная система и пользователь общаются с помощью речи. Данный вид интерфейса наиболее приближен к естественному общению между людьми]

Предметная область пользователя, *Предметная область контекста использования*, формализована в соответствии с материалами, рассмотренными в *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*.

Предметная область компонентов интерфейса содержит классификацию компонентов пользовательского интерфейса, пример приведен далее (см. *Голенков В.В..СтандОТОП-2021кн*).

компонент пользовательского интерфейса
:= [знак сущности базы знаний, имеющий определенную форму внешнего представления на экране]
⇒ *разбиение**:
{• *атомарный компонент пользовательского интерфейса*
:= [компонент *пользовательского интерфейса*, не содержащий в своем составе других *компонентов пользовательского интерфейса*]
• *неатомарный компонент пользовательского интерфейса*
:= [*компонент пользовательского интерфейса*, состоящий из других *компонентов пользовательского интерфейса*]
}

визуальная часть пользовательского интерфейса ostis-системы
:= [часть базы знаний пользовательского интерфейса ostis-системы, содержащая необходимые для отображения пользовательского интерфейса компоненты]
⊂ *неатомарный компонент пользовательского интерфейса*

Компоненты пользовательского интерфейса могут быть отнесены к одной из двух категорий: *компонент пользовательского интерфейса для отображения*, *интерактивный компонент пользовательского интерфейса*.

Полная классификация компонентов пользовательского интерфейса приведена далее:

компонент пользовательского интерфейса
- ⊃ *интерактивный компонент пользовательского интерфейса*
  - ⊃ *компонент ввода данных*
    - ⊃ *компонент ввода данных с прямой ответной реакцией*
      - ⊃ *область рисования*
      - ⊃ *ползунок*
      - ⊃ *компонент ввода текста с прямой ответной реакцией*
        - ⊃ *однострочное текстовое поле*
        - ⊃ *многострочное текстовое поле*
      - ⊃ *компонент выбора*
      - ⇒ *разбиение**:
        Типология компонентов выбора по количеству элементов
        - = {• *компонент выбора одного значения*
          • *компонент выбора нескольких значений*
          }
        - ⊃ *выбираемый элемент*
        - ⊃ *радиокнопка*
        - ⊃ *переключатель*
        - ⊃ *флаговая кнопка*
    - ⊃ *компонент ввода данных без прямой ответной реакции*
      - ⊃ *кнопка-счетчик*
      - ⊃ *компонент ввода движений*
      - ⊃ *компонент речевого ввода*
  - ⊃ *компонент для представления и взаимодействия с пользователем*
    - ⊃ *активирующий компонент*
    - ⊃ *компонент непрерывной манипуляции*
      - ⊃ *компонент редактирования размера*
      - ⊃ *полоса прокрутки*
  - ⊃ *компонент запроса действий*
    - ⊃ *компонент выбора команд*
      - ⊃ *пункт меню*
      - ⊃ *кнопка*
    - ⊃ *компонент ввода команд*
- ⊃ *компонент пользовательского интерфейса для отображения*
  - ⊃ *компонент вывода*
    - ⊃ *компонент вывода видео*
    - ⊃ *компонент вывода звука*
    - ⊃ *компонент вывода изображения*
    - ⊃ *компонент вывода графической информации*
      - ⊃ *индикатор выполнения*
      - ⊃ *диаграмма*
      - ⊃ *карта*
    - ⊃ *компонент вывода текста*
      - ⊃ *сообщение*
      - ⊃ *заголовок*
      - ⊃ *параграф*
  - ⊃ *декоративный компонент пользовательского интерфейса*
    - ⊃ *пустое пространство*
    - ⊃ *разделитель*
  - ⊃ *контейнер*
    - ⊃ *списковый контейнер*
    - ⊃ *древовидный контейнер*
    - ⊃ *узловой контейнер*
    - ⊃ *таблично-строковый контейнер*
    - ⊃ *таблично-клеточный контейнер*
    - ⊃ *панель вкладок*
    - ⊃ *панель вращения*
    - ⊃ *меню*
    - ⊃ *строка меню*
    - ⊃ *панель инструментов*

- ⊃ *строка состояния*
- ⊃ *панель прокрутки*
- ⊃ *окно*
  - ⊃ *модальное окно*
  - ⊃ *немодальное окно*

В рамках *Предметной области методик проектирования интерфейсов* предлагается формализовать различные существующие методы проектирования интерфейсов, такие как:
- проектирование пользовательских интерфейсов на основе онтологий(концепция разработки пользовательских интерфейсов на основе онтологий);
- методика эргономического проектирования;
- методика целеориентированного проектирования.

В рамках *Предметной области средств проектирования интерфейсов* предлагается формализовать существующие средства проектирования интерфейсов (см. *Myers B.aBriefHoHCIT-2001art*), такие как:
- средства для поддержки создания интерфейса написанием кода;
- интерактивные инструментальные средства;
- средства, основанные на создании интерфейса путем связывания отдельно созданных компонентов.

В рамках *Предметной области логических правил адаптации интерфейса* предлагается формализовать типологию логических правил, на основе которых будет происходить адаптация интерфейса к пользователю.

Таким образом, предложена структура *пользовательского интерфейса ostis-системы*, которая включает *базу знаний пользовательского интерфейса ostis-системы* и *решатель задач пользовательского интерфейса ostis-системы*. Рассмотрены *предметные области* в рамках *базы знаний пользовательского интерфейса ostis-системы*. Далее будут подробно рассмотрены классификация *интерфейсных действий пользователей ostis-систем*, *сообщений* и структура *решателя задач пользовательского интерфейса ostis-системы*.

§ 4.1.3. Интерфейсные действия пользователей ostis-систем

Действие, выполняемое пользователем над некоторым *компонентом пользовательского интерфейса*, называется *интерфейсным действием*. Для связи данного действия с *компонентом пользовательского интерфейса* и необходимым к выполнению *внутренним действием системы* используется отношение *инициируемое пользовательским интерфейсом действие**.

Классификация интерфейсных действий:

интерфейсное действие пользователя
- ⊃ *действие мышью*
  - ⊃ *прокрутка мышью*
    - ⊃ *прокрутка мышью вверх*
    - ⊃ *прокрутка мышью вниз*
  - ⊃ *наведение мышью*
  - ⊃ *отпускание мышью*
  - ⊃ *нажатие мыши*
    - ⊃ *одиночное нажатие мыши*
    - ⊃ *двойное нажатие мыши*
  - ⊃ *жест мышью*
  - ⊃ *отведение мышью*
  - ⊃ *перетаскивание мышью*
- ⊃ *действие голосом*
- ⊃ *действие клавиатурой*
  - ⊃ *нажатие функциональной клавиши*
  - ⊃ *нажатие клавиши набора текста*
- ⊃ *действие осязанием*
- ⊃ *действие сенсором*
  - ⊃ *нажатие сенсора*
    - ⊃ *одиночное нажатие сенсора*
    - ⊃ *двойное нажатие сенсора*
  - ⊃ *жест по сенсору*
    - ⊃ *жест по сенсору одним пальцем*

- ⊃ *жест по сенсору несколькими пальцами*
- ⊃ *отпускание сенсором*
- ⊃ *перетаскивание сенсором*
- ⊃ *действие пером*
  - ⊃ *нажатие функциональной клавиши пером*
  - ⊃ *рисование пером*
  - ⊃ *написание текста пером*

прокрутка мышью — интерфейсное действие пользователя, соответствующее прокрутке содержимого некоторого компонента пользовательского интерфейса при помощи мыши.

наведение мышью — интерфейсное действие пользователя, соответствующее появлению курсора мыши в рамках компонента пользовательского интерфейса.

отпускание мышью — интерфейсное действие пользователя, соответствующее отпусканию некоторого компонента пользовательского интерфейса в рамках другого компонента пользовательского интерфейса при помощи мыши.

нажатие мыши — интерфейсное действие пользователя, соответствующее выполнению нажатия мыши в рамках некоторого компонента пользовательского интерфейса.

отведение мышью — интерфейсное действие пользователя, соответствующее выходу курсора мыши за рамки компонента пользовательского интерфейса.

перетаскивание мышью — интерфейсное действие пользователя, соответствующее перетаскиванию компонента пользовательского интерфейса при помощи мыши.

нажатие сенсора — интерфейсное действие пользователя, соответствующее выполнению нажатия сенсора в рамках некоторого компонента пользовательского интерфейса.

жест по сенсору — интерфейсное действие пользователя, соответствующее выполнению некоторого жеста, выполняемого при помощи движения пальцев на экране сенсора.

отпускание сенсором — интерфейсное действие пользователя, соответствующее отпусканию некоторого компонента пользовательского интерфейса в рамках другого компонента пользовательского интерфейса при помощи сенсора.

перетаскивание сенсором — интерфейсное действие пользователя, соответствующее перетаскиванию компонента пользовательского интерфейса при помощи сенсора.

действие пером — интерфейсное действие пользователя, осуществляемое при помощи пера на графическом планшете.

класс интерфейсных действий пользователя — множество, элементами которого являются классы *интерфейсных действий пользователя*.

При взаимодействии пользователя с *компонентом пользовательского интерфейса* могут быть произведены различные *интерфейсные действия*. В зависимости от выполненного *интерфейсного действия* и компонента, над которым оно было выполнено, происходит инициирование некоторого *внутреннего действия системы*. Для задания такого инициируемого при взаимодействии с пользовательским интерфейсом действия и используется указанное отношение. Первым компонентом связки отношения *инициируемое пользовательским интерфейсом действие** является связка, элементами которой являются элемент множества компонентов пользовательского интерфейса и элемент множества *класс интерфейсных действий пользователя*. Вторым компонентом является элемент множества *класс внутренних действий системы*.

Таким образом, рассмотрена классификация *пользовательских действий пользователей ostis-системы*, которые могут выполняться в рамках *пользовательского интерфейса ostis-системы*.

§ 4.1.4. Сообщения, входящие в ostis-систему и выходящие из нее

сообщение — дискретная информационная конструкция, используемая в процессе передачи от отправителя к получателю.

В качестве *отправителя сообщения* может выступать как пользователь системы, так и сама система. В случае ostis-системы сообщение может быть эффекторным либо рецепторным.

эффекторное сообщение ostis-системы — сообщение ostis-системы, формируемое самой ostis-системой при возникновении некоторых ситуаций. К ситуациям, инициирующим возникновение эффекторных сообщений, можно отнести:

- ситуации, возникающие при анализе деятельности самого пользователя. Например, задание аргументов, не соответствующих типу инициируемого действия или появление подсказок при использовании компонентов пользовательского интерфейса;
- ситуации, возникающие при анализе синтаксиса текстов внешних языков. Например, неполнота сформированного предложения на внешнем языке или использование конструкций, нехарактерных или некорректно использованных в контексте отдельно взятого внешнего языка/

рецепторное сообщение ostis-системы — сообщение ostis-системы, являющееся реакцией на императивное сообщение (сообщение, побуждающее к какому-либо действию). Возможными реакциями ostis-системы на императивное сообщение пользователя являются:

- указание факта завершения выполнения некоторой задачи, что, например, характерно для поведенческих действий;
- получение ответа на поставленную задачу, формируемого либо в результате анализа базы знаний пользовательского интерфейса, либо в результате анализа предметной части базы знаний самой ostis-системы.

сообщение пользователя ostis-системы может быть сформировано как на внешнем языке (языке, внешнем по отношению к ostis-системе, который не используется для коммуникации внутри системы), так и на внутреннем языке (SC-коде).

Любое сообщение может быть атомарным (не содержащем в своем составе другие сообщения) либо неатомарным (сообщение, в состав которого входят другие сообщения).

Типология сообщений представлена в следующем фрагменте:

сообщение
⇒ *разбиение**:
{• *сообщение пользователя системы*
⊂ *сообщение пользователя ostis-системы*
• *сообщение системы*
}
⇒ *разбиение**:
{• *атомарное сообщение*
• *атомарное сообщение*
}
⇒ *разбиение**:
{• *сообщение на естественном языке*
• *сообщение на искусственном языке*
}
⊂ *графическое сообщение*
:= [сообщение, содержащее графическую информацию]
⊂ *видео-сообщение*
:= [сообщение, содержащее видео-информацию]
⊂ *аудио-сообщение*
:= [сообщение, представленное в звуковом формате]
⊂ *обонятельное сообщение*
:= [сообщение, содержащее информацию о запахах]
⊂ *текстовое сообщение*
:= [сообщение, содержащее текстовую информацию]
⊂ *сообщение, требующее трансляции*
:= [сообщение, которое необходимо сформировать системой для дальнейшей передачи его пользователю]
⊂ *протранслированное сообщение*
:= [сообщение, которое было сформировано системой для дальнейшей передачи его пользователю]

Таким образом, рассмотрена типология *сообщений*, которые являются основой взаимодействия пользователя с *интерфейсом* системы.

§ 4.1.5. Действия и внутренние агенты пользовательского интерфейса ostis-системы

решатель задач пользовательского интерфейса ostis-систем состоит из некоторого коллектива *sc-агентов*, обеспечивающих работу пользователя с *компонентами пользовательского интерфейса ostis-системы*.

При использовании *sc-агентов* стоит помнить различия в семантической и прагматической составляющей любого *компонента пользовательского интерфейса*. Семантическая составляющая заключается в определении того, знаком какой сущности является отображаемый на экране компонент. Прагматическая составляющая рассматривает прикладной аспект (аспект применения) отображаемого на экране компонента.

На уровне *sc-памяти* имеет значение только семантическая составляющая, однако данный факт не влияет на процесс эксплуатации системы пользователем, поскольку обе составляющие отражают разные стороны одного и того же знака некоторой сущности. Например, за каждой кнопкой скрывается знак некоторого *класса действия*, инициируемого при нажатии на кнопку. Таким образом, *интерфейсное действие пользователя*, как правило, инициирует некоторое *внутреннее действие системы*.

внутреннее действие системы
⊃ *внутреннее действие ostis-системы*

внутреннее действие ostis-системы
≔ [действие в sc-памяти]
≔ [действие, выполняемое в sc-памяти]

Каждое *внутреннее действие ostis-системы* обозначает некоторое преобразование, выполняемое некоторым *sc-агентом* (или коллективом *sc-агентов*) и ориентированное на преобразование *sc-памяти*.

действие в sc-памяти
⊃ *действие в sc-памяти, инициируемое вопросом*
⊃ *действие редактирования базы знаний ostis-системы*
⊃ *действие установки режима ostis-системы*
⊃ *действие редактирования файла, хранимого в sc-памяти*
⊃ *действие интерпретации программы, хранимой в sc-памяти*

С точки зрения обработки модели *базы знаний пользовательского интерфейса ostis-систем* должны быть решены следующие задачи:
- обработка пользовательских действий;
- интерпретация модели *базы знаний пользовательского интерфейса ostis-системы* (построение *пользовательского интерфейса*);

Таким образом, в *решателе задач пользовательского интерфейса ostis-систем* можно выделить *интерпретатор sc-моделей пользовательских интерфейсов* и *интерпретатор пользовательских действий*.

интерпретатор sc-моделей пользовательских интерфейсов в качестве входного параметра принимает экземпляр *компонента пользовательского интерфейса* для отображения. При этом компонент может быть как атомарным, так и неатомарным (например, компонент главного окна приложения). Результатом работы интерпретатора является графическое представление указанного компонента с учетом используемой реализации *платформы интерпретации семантических моделей ostis-систем*.

Алгоритм работы данного *интерпретатора* следующий (см. *Sadouski M.SemanDoaAUI-2022art*):
- проверяется тип входного *компонента пользовательского интерфейса* (атомарный или неатомарный);
- если *компонент пользовательского интерфейса* является атомарным, то отобразить его графическое представление на основании указанных для него свойств. В случае, если данный компонент не входит в *декомпозицию* любого другого *компонента пользовательского интерфейса* — завершить выполнение. Иначе определить компонент, в *декомпозицию* которого входит рассматриваемый *компонент пользовательского интерфейса*, применить его свойства для текущего атомарного компонента и начать обработку найденного неатомарного компонента, перейдя к первому пункту;
- если *компонент пользовательского интерфейса* является неатомарным, то проверить, были ли отображены компоненты, на которые он был декомпозирован. Если да, то завершить выполнение, иначе определить еще не отображенный компонент из декомпозиции обрабатываемого неатомарного компонента и начать обработку найденного компонента, перейдя к первому пункту.

интерпретатор пользовательских действий является *неатомарным sc-агентом*, который включает в себя множество *sc-агентов*, каждый из которых обрабатывает *интерфейсные действия пользователя* определенного класса (например, *абстрактный sc-агент обработки действия нажатия мыши*, *абстрактный sc-агент обработки дей-*

ствия отпускания мыши и так далее). Интерпретатор реагирует на появление в базе знаний системы экземпляра *интерфейсного действия пользователя*, находит связанный с ним класс *внутреннего действия* и генерирует экземпляр данного *внутреннего действия* для последующей обработки.

Таким образом, предложена структура *решателя задач пользовательского интерфейса ostis-систем*, которая позволяет обеспечить работу пользователя с *компонентами пользовательского интерфейса ostis-системы*.

Заключение к Главе 4.1.

В главе были рассмотрены принципы организации **партнерского взаимодействия пользователя с интеллектуальной системой** и принципы построения *интерфейсов интеллектуальных компьютерных систем нового поколения*, обеспечивающих переход к парадигме равноправного сотрудничества.

В результате проведенного анализа были сделаны следующие выводы:

- Для перехода к парадигме равноправного сотрудничества пользователя и системы интерфейсы должны быть адаптивными, интеллектуальными и мультимодальными. Существующие решения позволяют проектировать такие интерфейсы, однако имеют ряд недостатков.
- Структура *интеллектуальных интерфейсов* включает базу знаний, модуль управления взаимодействием пользователя с системой.
- При проектировании базы знаний активно применяется онтологический подход и уже реализованы некоторые онтологии, которые используются при проектировании *интеллектуальных интерфейсов*.
- Модуль управления взаимодействием пользователя с системой, как правило, реализуется на основе многоагентного подхода.

Среди недостатков существующих решений были выделены:

- Существующие решения, как правило, предусматривают вопросно-ответный принцип взаимодействия.
- Актуальной остается проблема совместимости интеллектуального интерфейса с интеллектуальной системой, для которой он создается, в силу различий используемых средств и методов при проектировании и реализации.
- Актуальной остается проблема совместимости компонентов интеллектуального интерфейса (база знаний и модуль управления взаимодействием) между собой.

Предложен онтологический подход на основе семантической модели к проектированию и реализации *адаптивного интеллектуального мультимодального пользовательского интерфейса* на основе *Технологии OSTIS*, который позволит обеспечить:

- совместимость интеллектуального интерфейса с интеллектуальной системой;
- совместимость компонентов интеллектуального интерфейса между собой;
- взаимодействие пользователя с системой через *интеллектуальный интерфейс* на равноправной основе.

Предложена и рассмотрена структура *пользовательского интерфейса ostis-системы* и составляющие ее компоненты.

Глава 4.2.
Естественно-языковые интерфейсы ostis-систем

⇒ *автор**:
- *Никифоров С.А.*
- *Гойло А.А.*
- *Цянь Л.*

⇒ *аннотация**:
[В данной главе рассматривается подход к реализации *естественно-языковых интерфейсов ostis-систем*, построенных по *Технологии OSTIS*, а также предлагается модель *контекста диалога*. В данном подходе все этапы анализа, включая *лексический*, *синтаксический* и *семантический анализ* могут производиться непосредственно в *базе знаний* такой системы. Такой подход позволит эффективно решать такие задачи как управление глобальным и локальным *контекстами диалога*, а также разрешение языковых явлений таких как анафоры, омонимия и эллиптические фразы.]

⇒ *подраздел**:
- *§ 4.2.1. Синтаксический анализ естественно-языковых сообщений, входящих в ostis-систему*
- *§ 4.2.2. Понимание естественно-языковых сообщений, входящих в ostis-систему*
- *§ 4.2.3. Методика и средства разработки естественно-языковых интерфейсов*

⇒ *ключевое знак**:
- *Абстрактный sc-агент лексического анализа*
- *Абстрактный sc-агент синтаксического анализа*
- *Абстрактный sc-агент понимания сообщения*

⇒ *ключевое понятие**:
- *естественно-языковой интерфейс*
- *речевой интерфейс*
- *действие. лексический анализ естественно-языкового сообщения*
- *действие. синтаксический анализ естественно-языкового сообщения*
- *действие. понимание естественно-языкового сообщения*
- *контекст*
- *контекст диалога*

⇒ *ключевое отношение**:
- *потенциально эквивалентная структура**
- *множество тематических контекстов диалога**

⇒ *ключевое знание**:
- *Структура решателя задач естественно-языкового интерфейса ostis-систем*
- *Синтаксический анализ естественно-языкового сообщения*
- *Понимание естественно-языкового сообщения*
- *Разрешение контекста*
- *Принципы построения естественно-языкового интерфейса ostis-систем для китайского языка*

⇒ *библиографическая ссылка**:
- *GlobaVAMbT-2019el*
- *Pais S..NLPBPaaSaBR-2022art*
- *Trajanov ..Surve oNLPiPMT-2022art*
- *Khurana D..NaturLPSotA-2022art*
- *Strubell E..Energ aPCfDL-2019art*
- *LargeLMaNML-2021el*
- *AmazoAOSWiA-2022el*
- *Siri-2022el*
- *GooglAYOP-2022el*
- *Cortana-2022el*
- *Hoy M.B.AlexaSCamItVA-2018art*
- *Jackendoff R..XS aSoPS-1977bk*
- *Davydenko I.T.SemanMMaToKB-2018art*

- *Shunkevich.D.V.AgentMMaToC-2018art*
- *Давыденко И.Т.МоделМиСРГБ-2017дс*

Введение в Главу 4.2.

В настоящее время существует большое количество различных *интерфейсов* компьютерных систем, что усложняет интероперабельность между такими системами и людьми в силу необходимости ознакомления с интерфейсом каждой новой системы, который зачастую может быть не интуитивно понятен.

Одной из основных особенностей *ostis-систем* должен являться *пользовательский интерфейс*, способный обеспечить эффективное взаимодействие пользователя с системой в условиях его общей профессиональной неподготовленности.

Одной из наиболее естественных и удобных форм передачи информации между людьми является речь, что обуславливает все большее распространение *естественно-языковых интерфейсов* (см. *GlobaVAMbT-2019el*). В настоящий момент времени уже ни у кого не вызывает сомнения, что данная форма взаимодействия человека и машины играет и будет играть значительную роль во взаимодействии с различными компьютерными системами.

Однако необходимо отметить, что большое многообразие *языков* (как *естественных*, так и *искусственных*) ведет к необходимости упрощения процесса создания таких *интерфейсов* для каждого отдельно взятого *языка*.

Разработка естественно-языковых интерфейсов для современных интеллектуальных систем, основанных на знаниях, в общем случае требует учета следующих двух основных аспектов:
- особенности обрабатываемого естественного языка;
- диапазон базы знаний интеллектуальных систем, то есть широта знаний в базах знаний интеллектуальных систем.

На данный момент существует большое количество методов обработки естественного языка, которые в основном можно разделить на два направления:
- методы на основе правил и лингвистических знаний;
- методы машинного обучения, основанные на математической статистике и теории информации.

В основе большинства подходов к обработке и пониманию *естественного языка* лежит машинное обучение (см. *Pais S..NLPBPaaSaBR-2022art*, *Trajanov ..Surve oNLPiPMT-2022art*). Несомненно, для большинства широко распространенных *языков* модели для обработки естественного языка работают очень хорошо и совершенствуются с каждым днем, но несмотря на успехи в данной области, данный подход имеет ряд недостатков:
- проблемы при работе с различными областями, например, значения слов или предложений могут быть различными в зависимости от *предметной области*. Таким образом, модели для NLP могут хорошо работать для отдельной *предметной области*, но не подходить для широкого применения (см. *Khurana D..NaturLPSotA-2022art*);
- создание новой модели модели требует наличия большого объема данных, а качество таких данных напрямую влияет на качество получаемой модели, что ведет к большим затратам на ее обучение (см. *Strubell E..Energ aPCfDL-2019art*, *LargeLMaNML-2021el*);
- данные модели представляют собой "черный ящик", так как данные модели не обладают средствами для обоснования своего вывода;
- каждая такая модель решает только свой узкий класс задач, отсутствует общий подход к обработке естественного языка (см. *Khurana D..NaturLPSotA-2022art*).

Данные недостатки используемых методов являются причиной части недостатков современных систем, реализующих *естественно-языковой интерфейс*, так, несмотря на то, что сейчас существует большое количество речевых ассистентов, создаваемых разными компаниями (см. *AmazoAOSWiA-2022el*, *Siri-2022el*, *GooglAYOP-2022el*, *Cortana-2022el*). Они обладают схожими недостатками, например, исключительно распределенной реализацией, в силу недостаточной для запуска ресурсоемких моделей производительности устройств конечных пользователей. Это в свою очередь ведет к проблемам с приватностью (см. *Hoy M.B.AlexaSCamItVA-2018art*).

Подмодуль понимания речи данных систем формирует конструкцию, отражающую смысл сообщения используя *фреймовую модель*. Упрощенный пример такой конструкции приведен на *Рисунок. Иллюстрация формализованного смысла сообщения*.

Рисунок. Иллюстрация формализованного смысла сообщения
=

```
{
  "text": "how many people between Tuesday and Friday",
  "intents": [
    { "name": "inquiry"}
  ],
  "entities": {
    "metric": [
      { "role": "metric", "value": "metric_visitor"}
    ],
    "datetime": [
      { "role": "datetime",  "type": "interval",
        "from": { "grain": "day", "value": "2020-05-05T00:00:00.000-07:00" },
        "to": { "grain": "day", "value": "2020-05-09T00:00:00.000-07:00" }
      }
    ]
  }
}
```

При этом для представления результатов промежуточных этапов обработки используются иные форматы, модули которые их реализуют не имеют какой-либо единой основы и взаимодействуют посредством специализированных *программных интерфейсов* между ними, что приводит к несовместимости способов представления результатов на различных этапах обработки и конечного результата обработки текстов. Данная несовместимость в свою очередь ведет к существенным накладным расходам при разработке такой системы и в особенности при ее модификации.

В качестве решения проблемы совместимости предлагается использование подхода к обработке *естественного языка* на основе его *формальной модели* в виде набора *онтологий*, сформированных с использованием универсальных средств представления знаний, что будет способствовать интероперабельности как компонента по обработке *естественного языка* в целом с другими компонентами системы, так и между составляющими самого данного компонента.

Целью главы является формирование модели *интерфейса*, в основе которой лежит подход к обработке *естественного языка* на основе *онтологий*, содержащих формальное описание *естественного языка*.

естественно-языковой интерфейс — *SILK-интерфейс* (Speech — речь, Image — образ, Language — язык, Knowledge — знание), обмен информацией между компьютерной системой и пользователем в котором происходит за счет диалога (см. *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*).

естественно-языковой интерфейс
⊃ *речевой интерфейс*

речевой интерфейс — *SILK-интерфейс*, обмен информацией в котором происходит за счет диалога, в процессе которого компьютерная система и пользователь общаются с помощью речи (см. *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*). Данный вид интерфейса наиболее приближен к естественному общению между людьми.

В предлагаемом подходе можно выделить следующие этапы обработки *естественного языка*:
- лексический анализ;
- синтаксический анализ;
- понимание сообщения.

В свою очередь, лексический анализ в включает в себя *декомпозицию* текста на токены и их сопоставление с *лексемами*.

Понимание сообщения сводится к генерации вариантов значения сообщения и выбору из них корректного на основании *контекста*, а также погружение его в данный *контекст*.

Ниже приведена структура *решателя задач естественно-языкового интерфейса*.

Решатель задач естественно-языкового интерфейса
⇒ *декомпозиция абстрактного sc-агента**:
- {• *Абстрактный sc-агент лексического анализа*
  ⇒ *декомпозиция абстрактного sc-агента**:
  - {• *Абстрактный sc-агент декомпозиции текста на токены*
    • *Абстрактный sc-агент сопоставления токенов с лексемами*
    }
- *Абстрактный sc-агент синтаксического анализа*

- *Абстрактный sc-агент понимания сообщения*

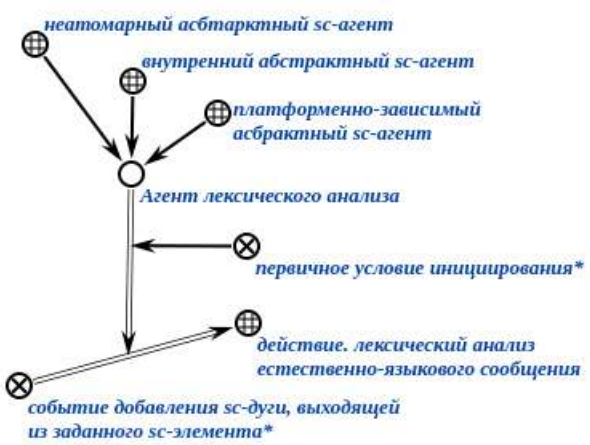
}

В свою очередь, *Абстрактный sc-агент понимания сообщения* декомпозируется на:

Агент понимания сообщения
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент генерации вариантов значения сообщения*
• *Абстрактный sc-агент выбора и обновления контекста*
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент разрешения контекста*
• *Абстрактный sc-агент выбора смысла сообщения на основе контекста*
• *Абстрактный sc-агент погружения сообщения в контекст*
}
}

Для каждого *агента* в базе знаний должна находиться спецификация, пример фрагмента такой спецификации приведен на *SCg-текст. Иллюстрация спецификации агента.*.

SCg-текст. Иллюстрация спецификации агента.
=



При анализе сообщения на *естественном языке* используются средства введенные в *Главе 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

§ 4.2.1. Синтаксический анализ естественно-языковых сообщений, входящих в ostis-систему

В данной главе мы не будем подробно описывать процесс лексического анализа и его ключевые аспекты (такие как, например, устранение омонимии) — данный вопрос требует дополнительной проработки. Вместо этого, акцент будет сделан на этапе синтаксического анализа, предварительным условием которого является уже проведенный лексический анализ текста *естественного языка*.

Лексический анализ основывается на средствах введенных в *§ 2.7.1. Формализация синтаксиса естественных языков*.

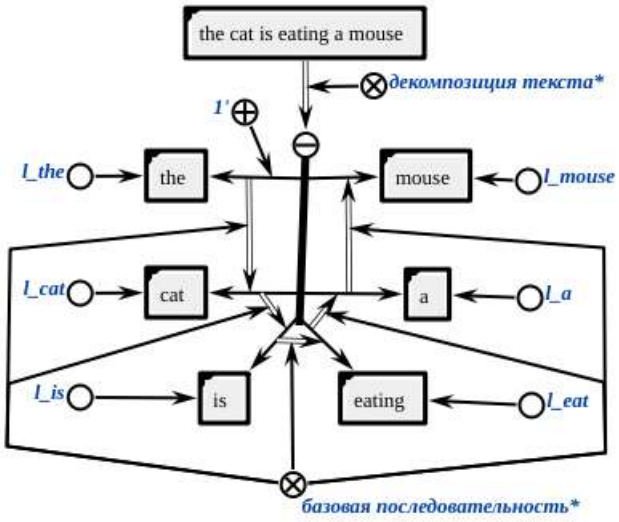
действие. лексический анализ естественно-языкового сообщения
⇒ *обобщенная декомпозиция**:
{• *действие. декомпозиция текста на токены*
• *действие. сопоставление токенов с лексемами*
}

С точки зрения *ostis-системы*, любой *естественно-языковой* текст является *файлом* (см. *§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*).

Лексический анализ представляет собой декомпозицию текста на последовательность токенов и сопоставление *лексем* с получившимися при данной декомпозиции токенами. Следует отметить, что данные токены при необходимости могут сопоставляться не с *лексемами*, а с их подмножествами, входящими в ее *морфологическую парадигму*, соответствующими определенным грамматическим категориям: падежу, числу, роду и так далее.

Результат лексического анализа представлен на *SCg-текст. Иллюстрация результата лексического анализа.*.

SCg-текст. Иллюстрация результата лексического анализа.
=



Для осуществления лексического анализа, в базе знаний системы также должен присутствовать словарь, содержащий *лексемы* и их различные формы.

Под *лексемой* понимается единица словарного состава языка, которая представляет собой множество всех форм некоторого слова. Пример спецификации *лексемы* в базе знаний приведен на *SCg-текст. Иллюстрация к спецификации лексемы в базе знаний.*, в *§ 2.7.1. Формализация синтаксиса естественных языков*.

В *§ 2.7.1. Формализация синтаксиса естественных языков* предлагалась формализация лингвистических знаний, совместимых в первую очередь с европейскими языками. В данной главе мы также рассмотрим, как могут учитываться особенности конкретных естественных языков при разработке естественно-языковых интерфейсов ostis-систем на примере китайского языка.

Традиционно в лингвистике структура слова изучается в рамках морфологии. Носителем морфологической парадигмы и ключевым элементом анализа является лексема. Однако, в китайском языке из-за письменной традиции (текст китайского языка состоит из последовательности иероглифов без пробелов), наименьшей единицей при обработке текстов считается единица сегментации. Описание единицы сегментации приводится в государственном стандарте «Стандарт сегментации слов современного китайского языка, используемый для обработки информации».

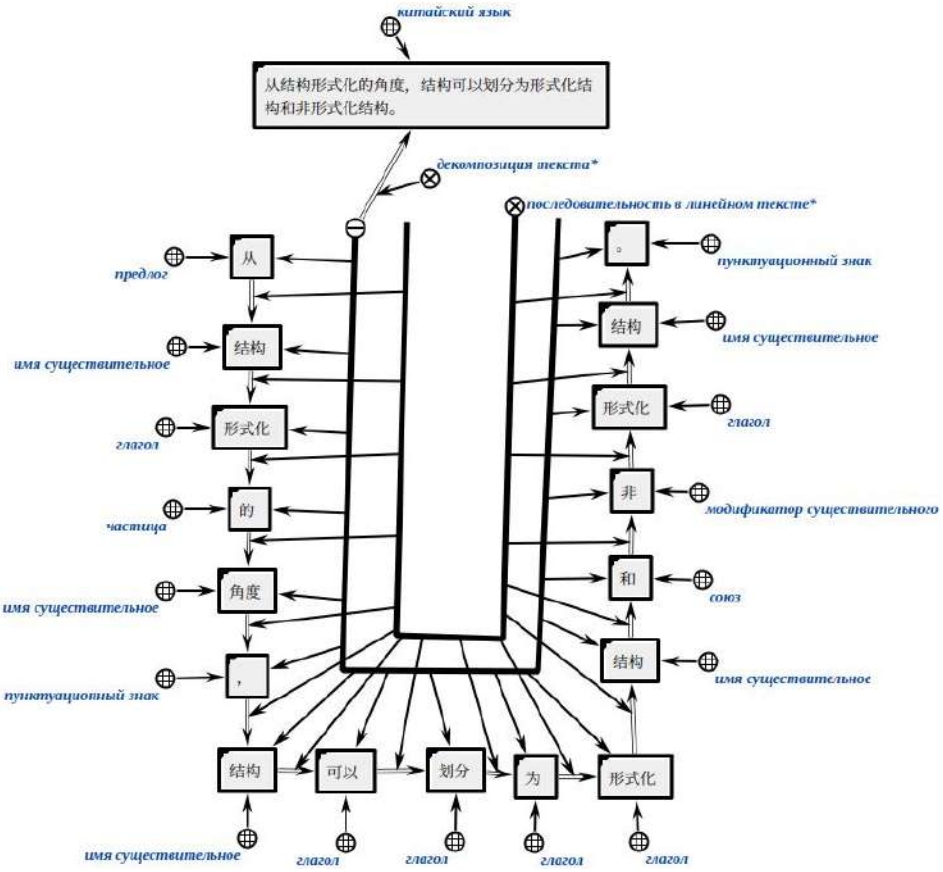
единица сегментации
:= *определение**:
[базовая единица для обработки китайского языка, имеющая определенные семантические или грамматические свойства]
⊂ *файл*

Стоит отметить, что термин единица сегментации используется при компьютерной обработке текстов китайского языка и не полностью совпадает с описанием слов в китайской лингвистике.

Кроме того, в китайском языке отсутствуют четкие показатели категорий числа, падежа и рода, в отличие от русского языка и других европейских языков. Функцию слова в китайском языке можно определить не на основании морфемного состава, а при помощи анализа связей этого слова с другими словами. В связи с этим, в процессе анализа текстов китайского языка сначала необходимо выполнить лексический анализ, разбивающий поток иероглифов в тексте китайского языка на отдельные значимые единицы сегментации.

Результат декомпозиции предложения на единицы сегментации представлен на *SCg-текст. Иллюстрация результата лексического анализа предложения на китайском языке*.

SCg-текст. Иллюстрация результата лексического анализа предложения на китайском языке
=



Агент синтаксического анализа выполняет переход от размеченного на *лексемы* текста к его *синтаксической структуре* (см. *§ 2.1.1. Формализация понятия информационной конструкции*).

При этом из-за невозможности разрешения структурной неоднозначности на этапе синтаксического анализа, его результатом в общем случае будет являться множество потенциальных синтаксических структур.

Синтаксический анализ также основывается на средствах введенных в *§ 2.7.1. Формализация синтаксиса естественных языков*, в котором приведен пример одной синтаксической структуры, представленный на *SCg-текст. Иллюстрация синтаксической структуры предложения. Первая часть* и *SCg-текст. Иллюстрация синтаксической структуры предложения. Вторая часть*.

§ 4.2.2. Понимание естественно-языковых сообщений, входящих в ostis-систему

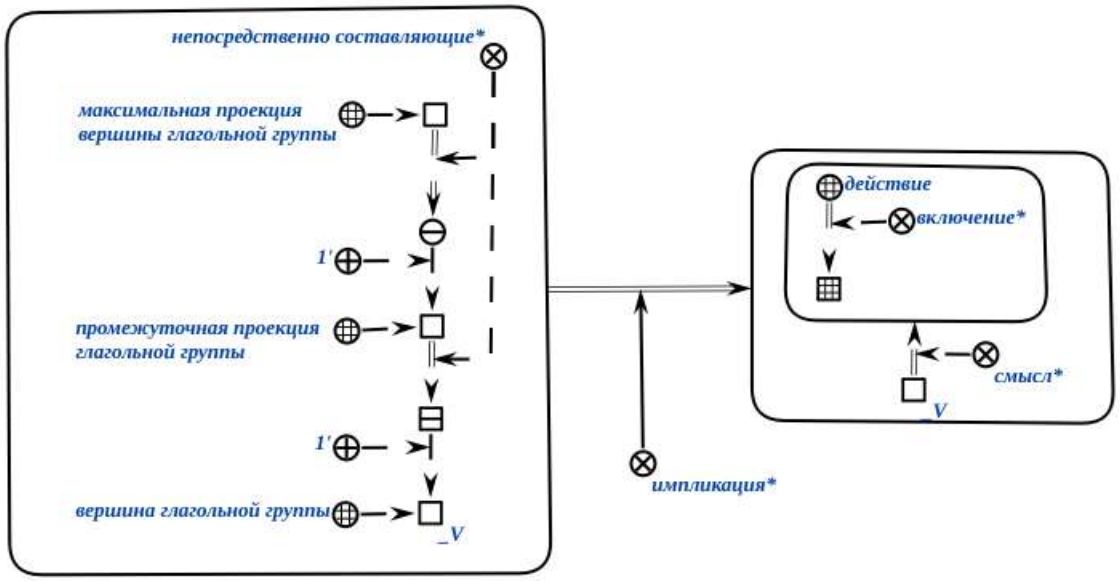
действие. понимание естественно-языкового сообщения
⇒ *обобщенная декомпозиция**:
{• *действие. генерация вариантов значения сообщения*
• *действие. выбор и обновление контекста*
⇒ *обобщенная декомпозиция**:
{• *действие. разрешение контекста*
• *действие. выбор смысла сообщения на основе контекста*
• *действие. погружение сообщения в контекст*
}
}

действие. генерация вариантов значения сообщения — *действие*, в ходе которого осуществляется формирование *строгой дизъюнкции* потенциально эквивалентных структур.

потенциально эквивалентная структура* — *бинарное ориентированное отношение*, связывающее структуру и множество структур, которые потенциально могут быть эквивалентны ей, однако для достоверного определения факта требуются дополнительные *действия*.

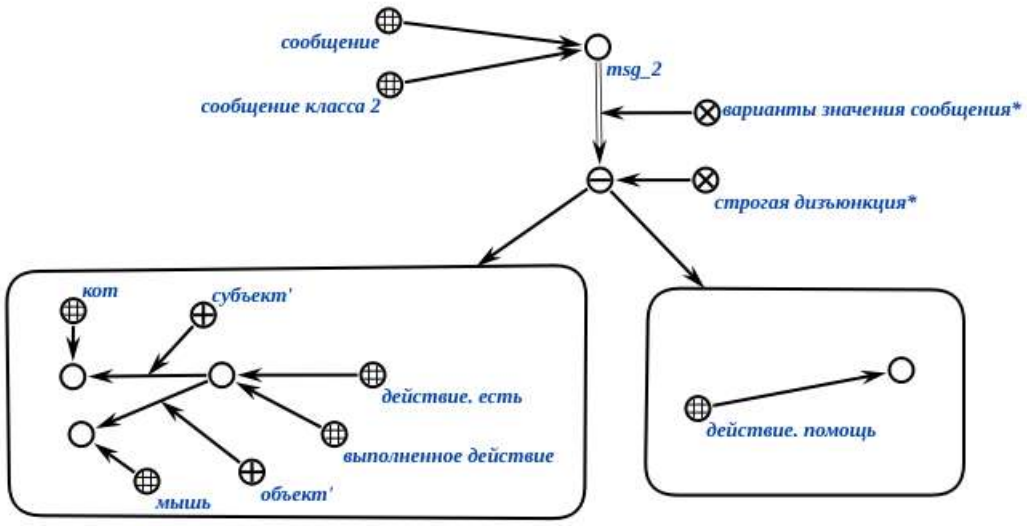
При этом, переход от результата синтаксического анализа к потенциально эквивалентным сообщению структурам осуществляется по правилам, представленным в подразделе § 2.7.2.. Пример одного из правил представлен на *SCg-текст. Иллюстрация правила перехода от синтаксической структуры к семантике*.

SCg-текст. Иллюстрация правила перехода от синтаксической структуры к семантике
=



В результате данного действия в базе знаний формируется структура, описывающая возможные варианты смысла сообщения, пример такой структуры в терминах грамматики составляющих (см. *Jackendoff R..XS aSoPS-1977bk*) приведен на *SCg-текст. Иллюстрация конструкции, описывающей потенциальные смыслы сообщения.*. Наличие нескольких таких структур объясняется тем, что в общем случае на этапе синтаксического анализа выполняется генерация нескольких вариантов синтаксической структуры. Выбор корректного значения сообщения будет осуществлен в ходе выполнения последующих действий.

SCg-текст. Иллюстрация конструкции, описывающей потенциальные смыслы сообщения.
=



Следует отметить, что при необходимости смысл сообщения может быть сгенерирован не только на основании его синтаксической структуры в терминах грамматики составляющих, но и других знаний о данном сообщении, например выделенных из текста данного сообщения троек вида субъект-отношение-объект, результата его классификации и тому подобные.

Дальнейшие этапы процесса понимания сообщения выполняются на основе контекста.

контекст — *sc-структура*, содержащая знания, которыми оперирует система в ходе одного или нескольких диалогов. В общем случае, данные знания включают в себя как предварительно занесенные в *базу знаний*, так и полученные в ходе работы с сенсоров и/или диалога.

контекст диалога
⊂ *контекст*
⇒ *разбиение**:
Типология контекстов диалога по глобальности^
= {• *тематический контекст*
• *пользовательский контекст*
• *глобальный контекст*
}

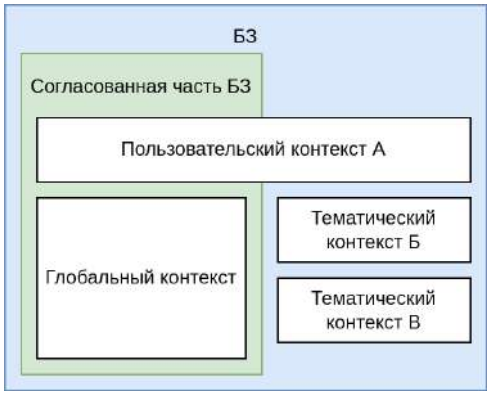
тематический контекст — *контекст диалога*, содержащий специфические для темы сведения (сведения, полученные во время ведения диалога, на определенную тематику, например, при диалоге об определенном наборе сущностей).

множество тематических контекстов диалога* — *бинарное ориентированное отношение*, диалог с ориентированным множеством его тематических контекстов.

пользовательский контекст — *контекст диалога*, содержащие специфические для пользователя сведения, которые могут быть использованы в диалоге с ним на любую тематику. В общем случае пользовательский *контекст* имеет пересечение с согласованной частью *базы знаний* (предварительно занесенная в *базу знаний* достоверная информация о пользователе, прошедшая необходимую модерацию), но не включается в нее целиком (часть, полученная в ходе диалога в которой мы не уверены). Пример соотнесения различных типов контекстов с согласованной частью базы знаний приведен на *Рисунок. Соотношение контекстов с согласованной частью баз знаний.*.

глобальный контекст — *контекст диалога*, содержащий сведения, которые могут быть необходимы при ведении диалога с любым пользователем. *глобальный контекст* — подмножество согласованной части *базы знаний*, содержащее те сведения, что допустимо использовать в диалоге. Например, в диалоге с определенным пользователем не нужно использовать:
- находящуюся в базе знаний служебную информацию, необходимую для работы системы, но не предназначенную для использования в диалоге;
- части пользовательских контекстов иных пользователей.

Рисунок. Соотношение контекстов с согласованной частью баз знаний.
=



контекст диалога
⇒ *разбиение**:
Типология контекство по сроку достоверности знаний^
= {• *неизменяемый в ходе работы системы контекст диалога*
• *изменяемый в ходе работы системы контекст диалога*

}

неизменяемый в ходе работы системы контекст диалога содержит в себе знания, необходимые для обеспечения выполнения системой своих функций, которые были заложены в нее априорно ее разработчиками и/или администраторами и не изменяются в ходе ее функционирования на постоянной основе.

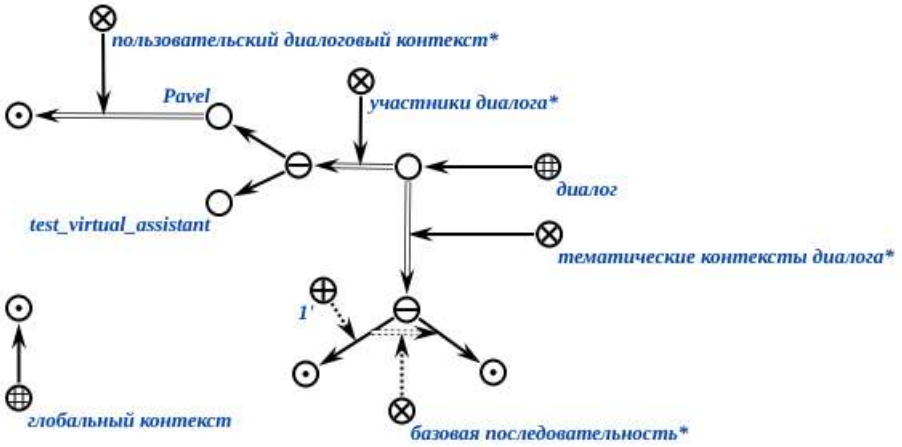
изменяемый в ходе работы системы контекст диалога содержит в себе знания, необходимые для обеспечения выполнения системой своих функций, которые были ей получены в ходе ее работы и/или достоверность которых скоротечна.

изменяемый в ходе работы системы контекст диалога
⇒ *разбиение**:
Типология изменяемых в ходе работы системы контекстов по источнику знаний^
= {• *контекст диалога, содержащий знания из внешних источников*
• *контекст диалога, содержащий знания, полученные в ходе диалога*
}
⇒ *разбиение**:
Типология изменяемых контекстов по степени их достоверности^
= {• *достоверный контекст диалога*
• *недостоверный контекст диалога*
}

Подмножество *контекста* может включаться в согласованную часть *базы знаний*, например, если речь идет о каких-то предварительно занесенных в *базу знаний* биографических сведениях — дате рождения и тому подобном.

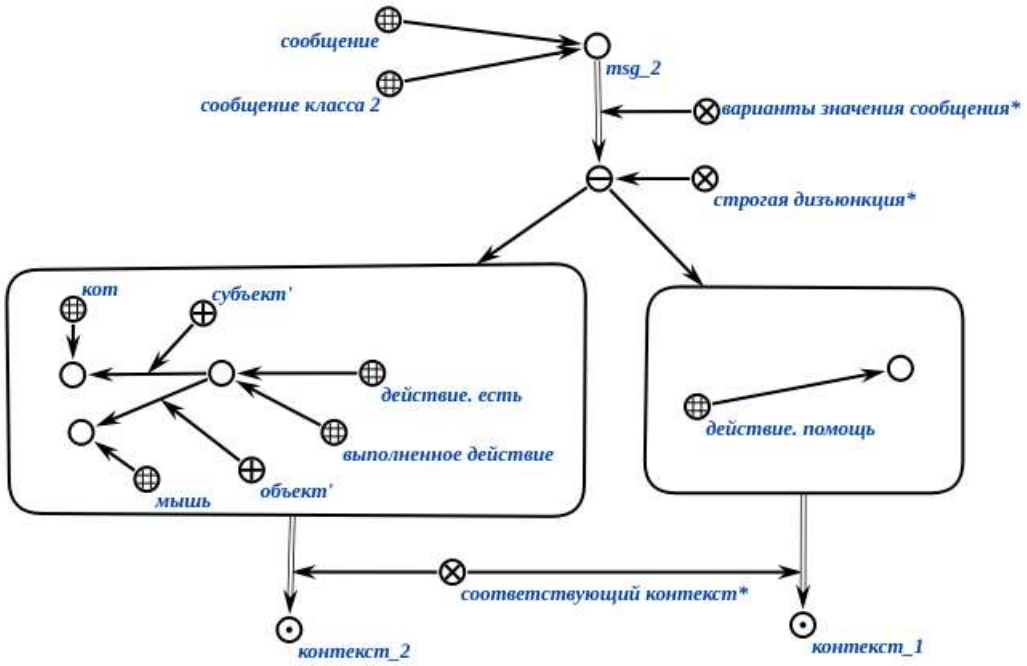
В каждый момент времени с пользователем связан 1 пользовательский диалоговый контекст (содержащий, по крайней мере известные заранее факты о нем: имя, возраст и тому подобное) и несколько тематических. Пример спецификации контекстов представлен на *SCg-текст. Иллюстрация спецификации контекстов.*.

SCg-текст. Иллюстрация спецификации контекстов.
=



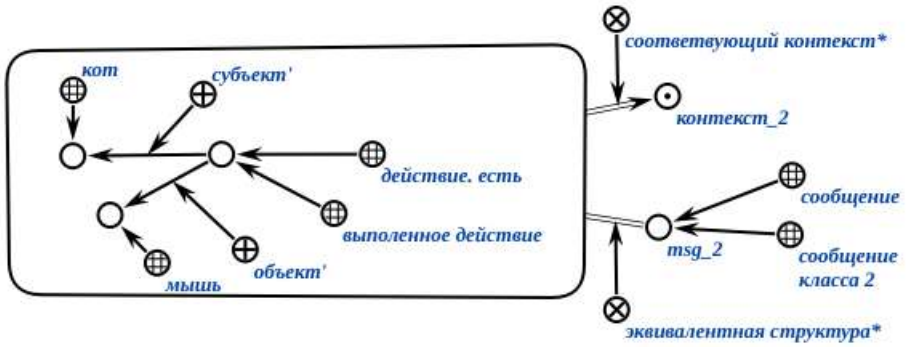
Так, ***действие. разрешение контекста*** сводится к сопоставлению каждому варианту его значения соответствующего *контекста*. Выбор производится на основании значения функции $F_{CTD}(T, C)$, где *T — вариант трансляции*, *C — тематический контекст*. Подходящим контекстом для варианта трансляции считается тот, для которого значение этой функции максимально. В случае, если подходящий контекст не найден, генерируется новый. Пример результата данного действия представлен на *SCg-текст. Иллюстрация сообщения, всем вариантам значения которого сопоставлен контекст.*.

SCg-текст. Иллюстрация сообщения, всем вариантам значения которого сопоставлен контекст.
=



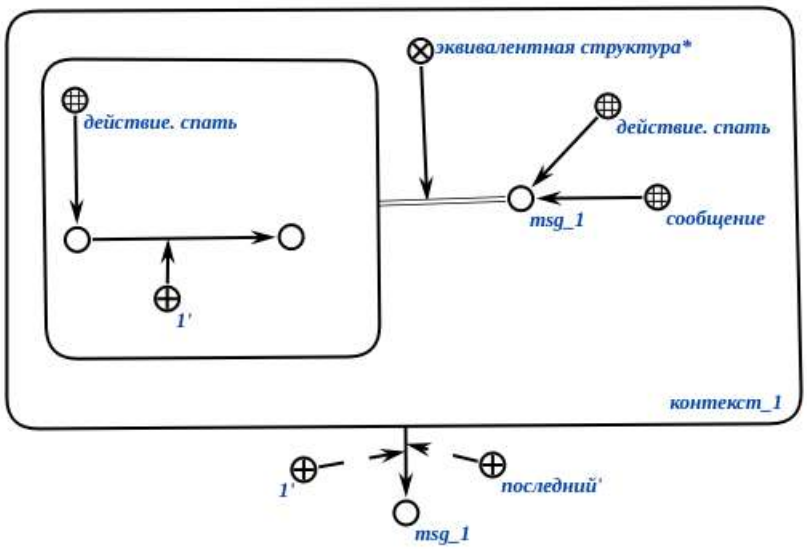
действие. выбор смысла сообщения представляет собой выбор из множества вариантов трансляции и соответствующих им *контекстов* одной пары и обозначение ее как эквивалентной сообщению конструкции. В простейшем случае, на данном этапе допустимо выполнить выбор в соответствии с рассчитанными на предыдущем этапе для пар потенциально эквивалентных структур и соответствующих им *контекстов* значениями функции $F_{CTD}(T, C)$ и выбрать пару, для которой оно максимально, однако при необходимости также возможно введение и отдельной функции. Пример результата данного действия представлен на *SCg-текст. Иллюстрация конструкции, описывающей эквивалентную сообщению структуру.*.

SCg-текст. Иллюстрация конструкции, описывающей эквивалентную сообщению структуру.
=

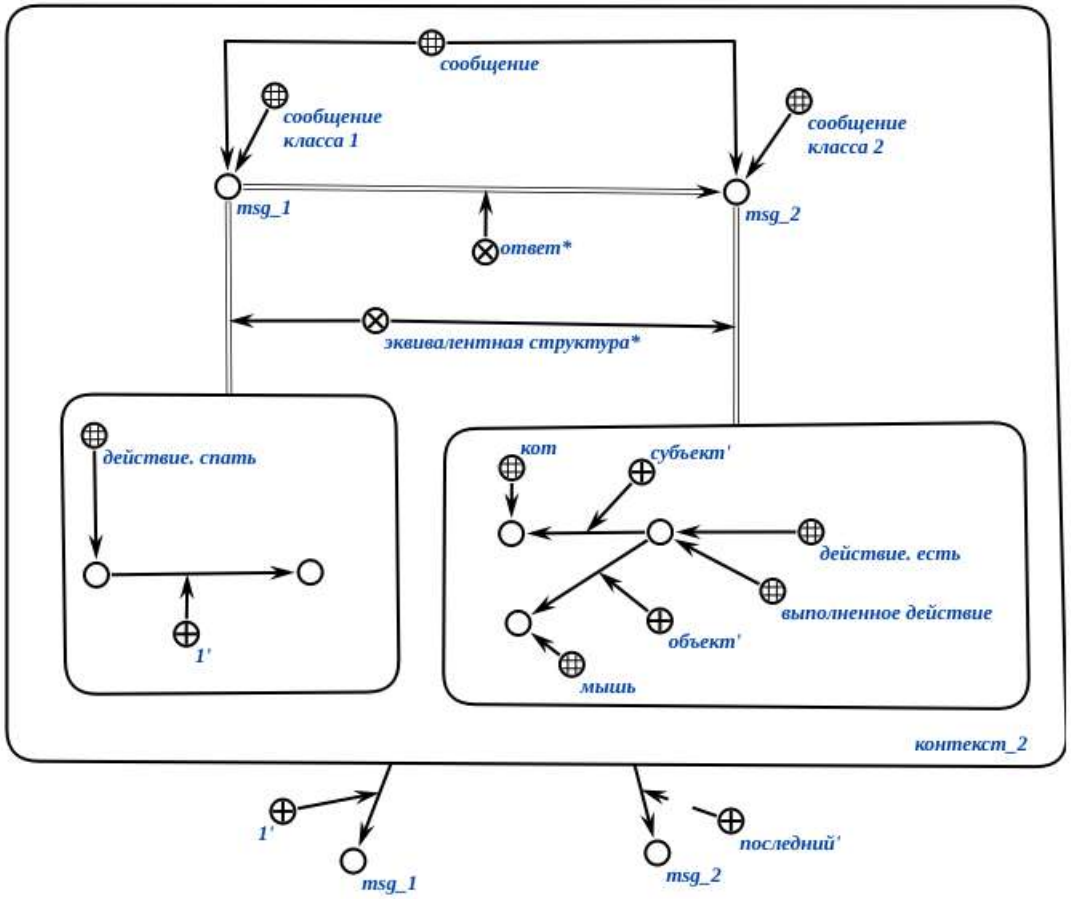


Действие. погружение сообщения в контекст представляет собой погружение полученного смысла сообщения в *контекст*. Кроме выбранного смысла сообщения, в контекст может добавляться и иная необходимая для обработки сообщения информация. Кроме того, на данном этапе на основе хранящихся в контексте сведений также должно выполняться разрешение местоимений. Примеры *контекста* до погружения в него сообщения и после погружения представлены на *SCg-текст. Иллюстрация контекста до погружения сообщения.* и *SCg-текст. Иллюстрация контекста после погружения сообщения.*.

SCg-текст. Иллюстрация контекста до погружения сообщения.
=



SCg-текст. Иллюстрация контекста после погружения сообщения.
=



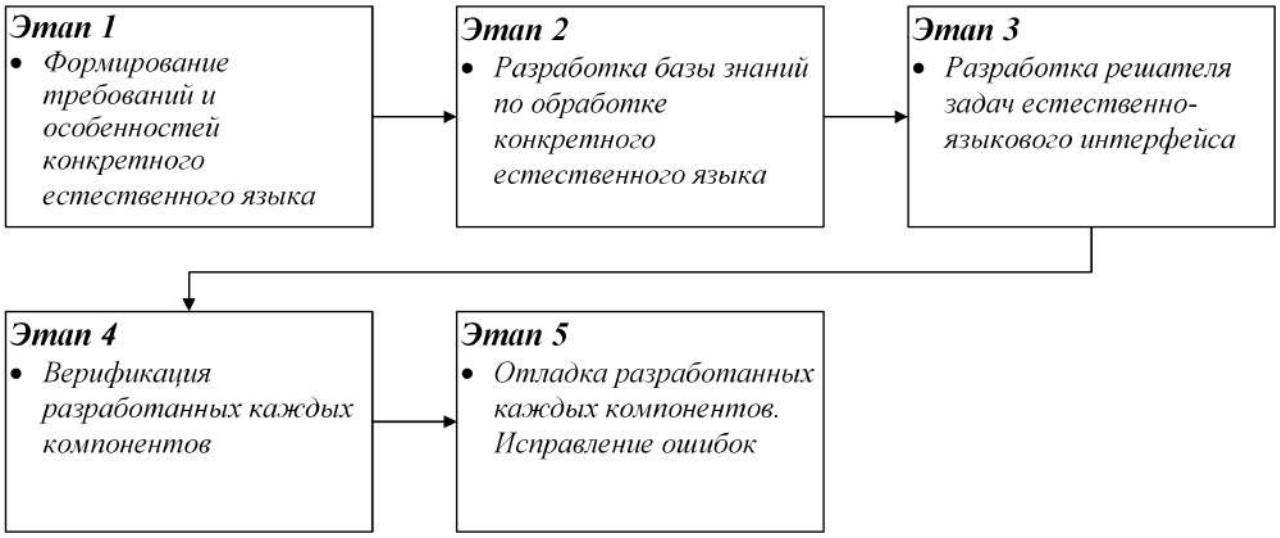
Таким образом, актуальная информация собирается в тематический контекст, объединив который с контекстом пользователя и глобальным контекстом можно получить общий контекст, на основании которого должны осуществляться требуемые действия системы, включая генерацию ответа системы.

§ 4.2.3. Методика и средства разработки естественно-языковых интерфейсов

Методика разработки естественно-языковых интерфейсов включает несколько этапов, в которых необходимо учитывать методику построения и модификации гибридных баз знаний и гибридных решателей задач (см. *Davydenko I.T.SemanMMaToKB-2018art*, см. *Shunkevich.D.V.AgentMMaToC-2018art*).

На *Рисунок. Этапы процесса разработки естественно-языкового интерфейса* представлен перечень таких этапов с указанием последовательности их выполнения.

Рисунок. Этапы процесса разработки естественно-языкового интерфейса
=



Данная методика может быть применена при разработке конкретного естественно-языкового интерфейса по конкретной предметной области.

Далее рассмотрим этапы процесса разработки естественно-языкового интерфейса ostis-систем:

Этап 1. Формирование требований с учетом особенностей конкретного естественного языка.

На данном этапе необходимо четко рассматривать особенности конкретного *естественного языка*. Затем можно разработать базу знаний по обработке конкретного *естественного языка* и соответствующие *решатели задач* для выполнения обработки. После определения конкретного *естественного языка* существует вероятность того, что в составе библиотеки компонентов уже есть реализованный вариант требуемой базы знаний и соответствующих решателей. В противном случае, тем не менее, у разработчика появляется возможность включить разработанную *базу знаний* по обработке конкретного естественного языка и соответствующие *решатели задач* в *библиотеку компонентов* для последующего использования.

Этап 2. Разработка базы знаний по обработке конкретного естественного языка.

На данном этапе при разработке базы знаний используются общие принципы согласованного построения и модификации *гибридных баз знаний* (см. *Давыденко И.Т.МоделМиСРГБ-2017дс*).

Этап 3. Разработка решателей задач естественно-языковых интерфейсов.

Для разработки *решателей задач естественно-языкового интерфейса*, направленных на приобретение фактографических знаний и генерацию текстов конкретного *естественного языка*, используются общие принципы согласованного построения и модификации гибридных *решателей задач* (см. *Shunkevich.D.V.AgentMMaToC-2018art*).

Этап 4. Верификация разработанных компонентов.

На данном этапе выполняется верификация разработанных *компонентов* (*базы знаний* по обработке конкретного естественного языка и соответствующего *решателя задач*) конкретного *естественно-языкового интерфейса*.

Этап 5. Отладка разработанных компонентов. Исправление ошибок.

Как правило, этапы 4 и 5 могут выполняться циклически до тех пор, пока разработанные компоненты не будут соответствовать предъявляемым требованиям.

Библиотека многократно используемых компонентов является важнейшим понятием в рамках *Технологии OSTIS* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*). Библиотека многократно используемых компонентов естественно-языковых интерфейсов позволяет выбрать компоненты уже разработанные компоненты и включить их в разрабатываемый *естественно-языковой интерфейс* других *ostis-систем*, то есть разработанные компоненты *естественно-языковых интерфейсов* могут быть повторно использованы при разработке естественно-языковых интерфейсов в других *ostis-системах*. Ниже предложена структура библиотеки многократно используемых компонентов *естественно-языковых интерфейсов*.

Библиотека многократно используемых компонентов естественно-языковых интерфейсов
⇒ *разбиение**:
- {• *Библиотека многократно используемых компонентов базы знаний естественно-языковых интерфейсов*
  - := [Библиотека многократно используемых компонентов лингвистической базы знаний]
  - := [Библиотека многократно используемых компонентов базы знаний по обработке естественного языка]
- • *Библиотека многократно используемых компонентов решателей задач естественно-языковых интерфейсов*
  - := [Библиотека многократно используемых компонентов решателей задач для обработки естественного языка]
- }

При необходимости, *библиотеку многократно используемых компонентов естественно-языковых интерфейсов* можно дополнять знаниями о конкретных *естественных языках*. Ниже представлен пример структуры библиотеки многократно используемых компонентов для построения китайско-языкового интерфейса.

Библиотека многократно используемых компонентов китайско-языкового интерфейса
⇒ *разбиение**:
- {• *Библиотека многократно используемых компонентов базы знаний китайско-языкового интерфейса*
  - := [Библиотека многократно используемых компонентов базы знаний по обработке китайского языка]
- • *Библиотека многократно используемых компонентов решателей задач китайско-языкового интерфейса*
  - := [Библиотека многократно используемых компонентов решателей задач для обработки китайского языка]
- }

Глава 4.3.
Аудиоинтерфейс ostis-систем

⇒ *авторы**:
- *Захарьев В.А.*
- *Азаров И.С.*
- *Вашкевич М.И.*
- *Крищенович В.А.*
- *Жаксылык К.*

⇒ *аннотация**:
[Данная глава посвящена рассмотрению вопросов создания *аудио* и *речевых интерфейсов* для *интеллектуальных компьютерных систем нового поколения*. Предлагается использование подхода на основе онтологического проектирования и формализации системы понятий из предметной области аудиоинтерфейсов, посредством *Технологии OSTIS*. Изложены основные идеи, лежащие в основе данного подхода, а также их отличительные особенности от общепринятых. Показано, что в перспективе использование данного подхода может обеспечить свойства *унификации*, *семантической совместимости* и *интероперабельности*, при разработке аудио и речевых интерфейсов, что в итоге позволит существенным образом сократить издержки при создании *интеллектуальных компьютерных систем нового поколения* для решении *комплексных задач*.]

⇒ *подраздел**:
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*
- *§ 4.3.2. Предметная область и онтология задач аудиоинтерфейса ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

⇒ *ключевое понятие**:
- *сигнал*
- *аудиосигнал*
- *речевой сигнал*
- *модель сигнала*
- *аудиоинтерфейс*
- *речевой интерфейс*

⇒ *библиографическая ссылка**:
- *Pearl C.DesigVUIPoC-2016bk*
- *Chen N..SelfSDLfSC-2021art*
- *Lu L..ConteAfACaS-2002art*
- *Fernandes E.Speec aVRM-2022el*
- *Bellegarda J..SpokeLUfNIS-2014art*
- *Lemley J..DeepLfCDaSP-2017art*
- *Hoy M.B.AlexaSCamItVA-2018art*
- *Semantic S.ScienPSRbK-2022el*
- *Popov V..Fast aLoDTTSw-2020art*
- *Povey D..tKaldiSRT-2011art*
- *Deepa P..aRepor oVRSTM-2021art*
- *VOSK-2022el*
- *Radford A..RobusSRvL-2022art*
- *Deli V..SpeecTPBoNM-2019art*
- *Голенков В.В..СтандОТОП-2021кн*
- *Давыденко И.Т.МоделМиСРГБ-2017дс*
- *Shunkevich.D.V.AgentMMaToC-2018art*
- *Захарьев В.А..Подхо кУРНнОСАА-2018ст*
- *Zahariev V.A..SemanAoVMBoaFC-2019cm*
- *Zahariev V.A..IntelVABoOST-2020art*
- *Zahariev V.A..ConveSABotFRotML-2021cm*
- *Serra X..aSystefSATS-1990art*
- *Griffin D.W..MultiEV-1988art*
- *Petrovsky A..HybriSDBoIH-2011art*

- *Azarov E..InstaHRoSUM..-2013art*
- *Laroche J..HNSSMBoaHNM-1993art*
- *McAulay R..SpeecASBoaSR-1986art*
- *Degottex G..MixedSMaiAVT-2013art*
- *Kawahara H.Explo otOAoVR-2010art*
- *Kawahara H..Devel oERTB-2009art*
- *ISOIECITCoAVO-2005art*
- *ISOIECITMAT-2020art*
- *IEEESfSGAC-2020el*
- *IECAVaRE-2015el*
- *AEST3250-2004art*

Введение в Главу 4.3.

Разговорная речь является одной из наиболее естественных и эффективных форм передачи информации между людьми. Этот факт объясняет значительный интерес исследователей к вопросам развития и применения *речевых интерфейсов* для обеспечения человеко-машинного взаимодействия в составе современных коммуникационных, мультимедийных и интеллектуальных систем (см. *Pearl C.DesigVUIPoC-2016bk*, *Chen N..SelfSDLfSC-2021art*).

Более всеобъемлющей формой обеспечения взаимодействия с пользователем и окружающей средой посредством анализа и синтеза акустических сигналов является *аудиоинтерфейс*. Данную разновидность интерфейса, выступающей родительской по отношению к речевым, можно кратко определить как аппаратно-программный комплекс осуществляющий анализ и синтез сигналов во всем доступном спектре параметров носителей акустической информации. Например, для решения задач анализа обстановки и событий происходящих в акустическом окружении системы, синтеза неречевых сигналов (звуков техногенного и природного характера, сигналов оповещения, музыки, и так далее) (см. *Lu L..ConteAfACaS-2002art*).

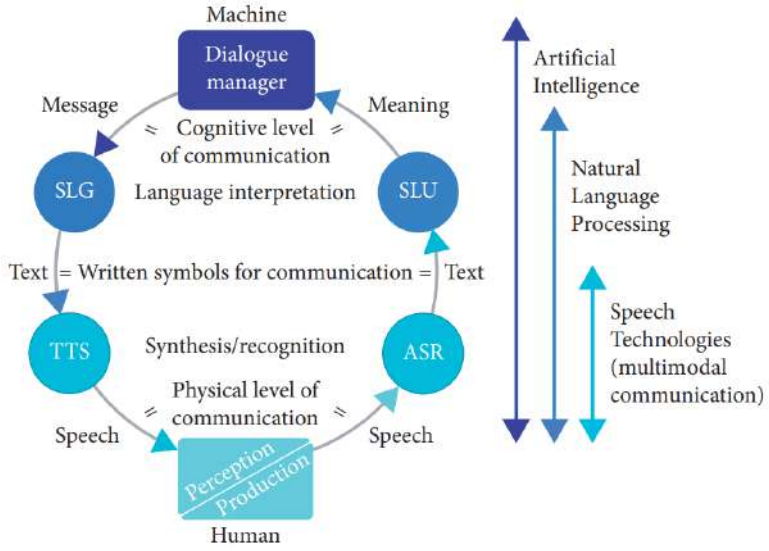
Об актуальности направления разработки аудио и речевых интерфейсов свидетельствуют следующие основные тенденции развития данного направления:

- экономические показатели и прогнозы развития рынка речевых технологий, текущие среднегодовые темпы роста которого, по оценкам экспертов, составляют порядка 22%, а совокупный объем будет равен 59,6 млрд. долл. США к 2030 (см. *Fernandes E.Speec aVRM-2022el*);
- появление широкого спектра продуктов на основе речевого интерфейса, получивших массовое распространение. В первую очередь это персональные голосовые ассистенты, такие как “Alexa” (Amazon), “Siri” (Apple), “Cortana” (MicroSoft), “Алиса” (Yandex) (см. *Bellegarda J..SpokeLUfNIS-2014art* , *Lemley J..DeepLfCDaSP-2017art*, *Hoy M.B.AlexaSCamItVA-2018art*);
- интерес со стороны научного сообщества, выражающийся в росте публикаций в этом направлении исследований на 15% за последние 5 лет (см. *Semantic S.ScienPSRbK-2022el*).

Необходимо отметить, что основная масса научных публикаций в данном направлении посвящена развитию базовых технологий, являющихся составляющими речевого интерфейса, таким как синтез речи по тексту, а также распознавание речи в текст (см. *Popov V..Fast aLoDTTSw-2020art* , *Povey D..tKaldiSRT-2011art* , *Deepa P..aRepor oVRSTM-2021art*). Последние достижения в этих направлениях связаны с бурным развитием нейросетевых моделей и вычислительных средств. Они позволили довести качественные характеристики использования речевых технологий до коммерческого уровня (см. *VOSK-2022el*, *Radford A..RobusSRvL-2022art*).

Большинство существующих систем, как правило, рассчитаны на решение определенного круга задач и сложно совместимы друг с другом. Данный факт в особенности остро проявляется при проектировании сложных систем, наподобие интеллектуальных персональных диалоговых ассистентов (см. *Рисунок. Компоненты системы человеко-машинного речевого диалога Deli V..SpeecTPBoNM-2019art*), требующих использования многообразия различных видов обрабатываемой информации и различных *моделей решения задач*. Такие системы, кроме стандартных модулей распознавания (ASR, automatic speech recognition) и синтеза (TTS, text to speech), на уровне аудиоинтерфейса также должны содержать модели, определяющие наличие/отсутствие речи в аудиосигнале в сложной акустической обстановке, классификации звуков окружающей среды, распознавание диктора и пр. Помимо этого элементы речевого интерфейса должны быть совместимы с более высокоуровневыми модулями обработки естественно-языковой информации, такими как модули понимания (SLU, spoken language understanding) и генерации речи (SLG, spoken language generation), управления диалогом (DM, dialog manager) (см. *Deli V..SpeecTPBoNM-2019art*).

Рисунок. Компоненты системы человеко-машинного речевого диалога Deli V..SpeecTPBoNM-2019art =



Все это требует разработки подходов, основанных не только на методах машинного обучения и обработке сигналов, но и на обработке естественного языка, символических методах искусственного интеллекта, онтологическом проектировании и формализации предметной области аудиоинтерфейса. Это позволит создать системы, которые обладают полным спектром знаний в формализованном виде о типах задач, которые они должны решать, и методах, доступных для их решения.

Необходимым условием для создания таких систем нового поколения, обладающих улучшенными характеристиками по критериям *интероперабельности* и *гибкости* является также тот факт, что данные системы должны быть построены на основе базовой технологии, позволяющей обеспечить такое единство формы представления информации на всех ее уровнях.

Совокупность данных факторов приводит к необходимости создания *интеллектуальных компьютерных систем нового поколения*, которые будут включать в себя модули аудио и речевого интерфейса, построенные на основе принципов интероперабельности и семантической совместимости для решения *комплексных задач*.

§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов

Для достижения поставленной цели предлагается прибегнуть к подходу на основе принципов, лежащих в основе “Стандарта открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем” или кратко “Стандарта технологии OSTIS” (см. *Голенков В.В..СтандОТОП-2021кн*).

Суть подхода заключается в рассмотрении процесса проектирования аудиоинтерфейса как интерфейсной подсистемы в рамках общего процесса разработки *интеллектуальной компьютерной системы* и построении ее формальной *логико-семантической модели*.

Для создания подобной модели интеллектуальной компьютерной системы нового поколения необходимо:

- произвести декомпозицию информационной компьютерной системы на компоненты. Качество декомпозиции при этом определяется простотой последующего синтеза общей формальной модели из формальных моделей выделенных компонентов.
- провести *конвергенцию* выделенных компонентов в целях построения совместимых (легко интегрируемых) формальных моделей этих компонентов;
- провести интеграцию построенных формальных моделей выделенных компонентов и получить общую *формальную модель*.

В качестве технологической основы для реализации предлагаемого подхода будет использоваться Технология OSTIS (см. *Голенков В.В..СтандОТОП-2021кн*), соответственно подсистема аудиоинтерфейса будет строиться как *многократно используемый компонент*, который в будущем будет при необходимости встраиваться в различные ostis-системы.

Ориентация на *Технологию OSTIS* обусловлена ее следующими основными преимуществами:

- в рамках указанной технологии предложены унифицированные средства представления различных видов знаний, в том числе — *метазнаний*, что позволяет описать всю необходимую для анализа информацию в одной базе знаний в едином ключе (см. *Давыденко И.Т.МоделМиСРГБ-2017дс*);
- используемый в рамках технологии формализм позволяет специфицировать в базе знаний не только понятия, но и любые внешние с точки зрения базы знаний файлы (например, фрагменты речевого сигнала), в том числе — синтаксическую структуру таких файлов;
- предложенный в рамках технологии подход к представлению различных видов знаний (см. *Давыденко И.Т.МоделМиСРГБ-2017дс*) и моделей их обработки (см. *Shunkevich.D.V.AgentMMaToC-2018art*) обеспечивает модифицируемость ostis-систем, то есть позволяет легко расширять функциональные возможности системы, вводя новые виды знаний (новые системы понятий) и новые модели обработки знаний;

Более подробно принципы построения комплексной технологии разработки и поддержки жизненного цикла *интеллектуальных компьютерных систем нового поколения — Технологии OSTIS* — изложены в *Главе 1.2. Интеллектуальные компьютерные системы нового поколения*.

В данной главе монографии, в отличие от предыдущих работ авторов, посвященных вопросам семантического анализа голосовых сообщений на основе формализованного контекста и создания диалоговых ассистентов на основе модели ментального лексикона или мультимодальной системы на основе нейросимволического подхода, Технология OSTIS используется для непосредственного построения онтологии подсистем аудио интерфейса (см. *Захарьев В.А..Подхо кУРНнОСАА-2018cm*, *Zahariev V.A..SemanAoVMBoaFC-2019cm*, *Zahariev V.A..IntelVABoOST-2020art*, *Zahariev V.A..ConveSABotFRotML-2021cm*).

Поскольку *аудиоинтерфейс интеллектуальных компьютерных систем нового поколения* должен иметь архитектуру, соответствующую общим правилам построения ostis-систем, можно выделить и формализовать следующие основные его части:

аудиоинтерфейс интеллектуальных компьютерных систем нового поколения

⇒ *сокращение**:
[аудиоинтерфейс *интеллектуальных компьютерных систем*]

⇒ *обобщенная декомпозиция**:
{• *база знаний подсистемы аудиоинтерфейса интеллектуальных компьютерных систем нового поколения*
• *решатель задач подсистемы аудиоинтерфейса интеллектуальных компьютерных систем нового поколения*
• *интерфейс для взаимодействия с остальными интерфейсными подсистемами ostis-системы*
}

Согласно общим принципам организации интерфейсов ostis-систем, изложенным в *Главе 4.1.*, *аудио- и речевой интерфейс* относятся к подмножеству *SILK-интерфейсов пользовательских интерфейсов интеллектуальных компьютерных систем*.

Для решения задачи построения пользовательского интерфейса в базе знаний пользовательского интерфейса ostis-системы необходимо наличие *sc-модели* компонентов *пользовательского интерфейса*, интерфейсных действий пользователей, а также классификации пользовательских интерфейсов в целом. При проектировании интерфейса используется компонентный подход, который предполагает представление всего интерфейса приложения в виде отдельных *специфицированных компонентов*, которые могут разрабатываться и совершенствоваться независимо.

Таким образом, необходимо отметить, что процесс разработки аудиоинтерфейса для *интеллектуальных компьютерных систем нового поколения*, подразумевает прежде всего создание семантически структурированных баз знаний в виде иерархической системы предметных областей и соответствующих им онтологий, специфицирующих эти предметные области. Следовательно, первым шагом для достижения поставленной цели должен являться этап выделения и формализации сущностей аудио и речевого интерфейса для погружения данной информации в базу знаний интеллектуальной компьютерной системы.

С нашей точки зрения можно провести декомпозицию предметных областей и онтологий, входящих в *базу знаний аудиоинтерфейса*, по следующим основным направлениям:

Предметная область и онтология аудиоинтерфейса интеллектуальных компьютерных систем нового поколения

⇒ *декомпозиция**:
{• *Предметная область и онтология задач аудиоинтерфейса*
• *Предметная область и онтология моделей параметрического представления сигнала*
}

Как видно, во главу онтологии положен функциональный подход к декомпозиции предметных областей, что является вполне естественным, поскольку соответствует природе задач, реализуемых аудиоинтерфейсом.

Представленные выше принципы в совокупности позволяют осуществлять конвергенцию и интеграцию компонентов как на уровне подсистемы аудиоинтерфейса, так и на уровне всей *интеллектуальной компьютерной системы нового поколения* в целом, что, в свою очередь, позволяет перевести *интеллектуальную информационную систему* в класс гибридных, *интероперабельных* и *семантически совместимых систем*.

Далее перейдем непосредственно к рассмотрению конкретных предметных областей и построению онтологии аудиоинтерфейса.

§ 4.3.2. Предметная область и онтология задач аудиоинтерфейса ostis-систем

Первым шагом на пути к построению базы знаний подсистемы аудиоинтерфейса *интеллектуальных компьютерных систем нового поколения* является формализация онтологии верхнего уровня. В основе данной онтологии предлагается положить формализованное представление основных сущностей предметной области и их свойств, а также функциональных задач, которые аудио и речевой интерфейс призваны решать.

К основным сущностям, требующим формализации и погружения в базу знаний, относятся множества понятий, представленные далее.

Одним из ключевых понятий, требующих формализации, является базовое определение самого сигнала, а также основных разновидностей сигналов, в зависимости от их природы представляющих наибольший интерес в области аудиоинтерфейсов. Чтобы сделать процесс описания средствами Технологии OSTIS более ясным, перед непосредственным переходом к нему приведем перечень основных сущностей и понятий, которые требуют формализации и погружения базу знаний:

- сигнал;
- акустический сигнал;
- аудиосигнал;
- речевой сигнал.

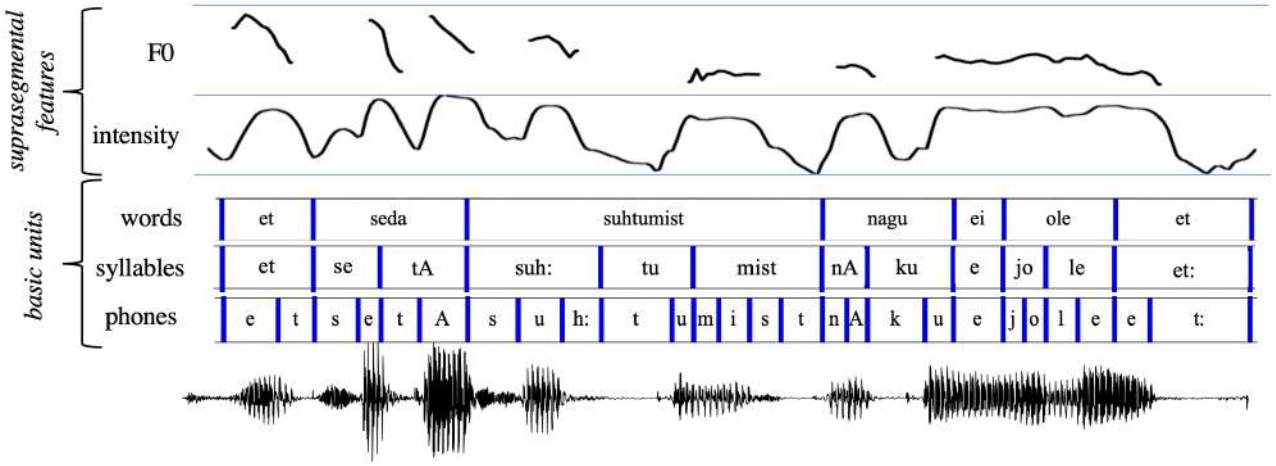
В зависимости от способа математического описания обрабатываемого сигнала в ostis-системе, можно определить следующие их классы:

- аналоговый сигнал;
- дискретный сигнал;
- цифровой сигнал;
- периодический сигнал;
- апериодический сигнал;
- гармонический сигнал;
- тональный сигнал;
- шумовой сигнал;
- импульсный сигнал;

Формализуем также основные понятия, связанные с характеристиками самого сигнала, по следующим основным его атрибутам *Рисунок. Сегментные и надсегментные характеристики речевого сигнала*:

- амплитуда сигнала;
- частота сигнала;
- фаза сигнала;
- интенсивность сигнала;
- длительность сигнала;
- мощность/энергия сигнала;
- осцилограмма сигнала;
- спектр сигнала;
- частота дискретизации сигнала;
- степень квантования сигнала;

Рисунок. Сегментные и надсегментные характеристики речевого сигнала
=



К ключевым понятиям предметной области, лежащим в семантической окрестности подпространства функционального назначения аудиоинтерфейсов и обработки аудиосигналов, относятся:
- анализ аудиосигнала;
- синтез аудиосигнала;
- кодирование аудиосигнала;
- шумоочистка аудиосигнала;
- классификация аудиосигнала;
- классификация событий (Enviromental Sound Classification, Acoustic Scenes and Events);
- детектирование аномалий (Anomalous Sound Detection);
- идентификация положения источника в пространстве (Sound Source Localization);

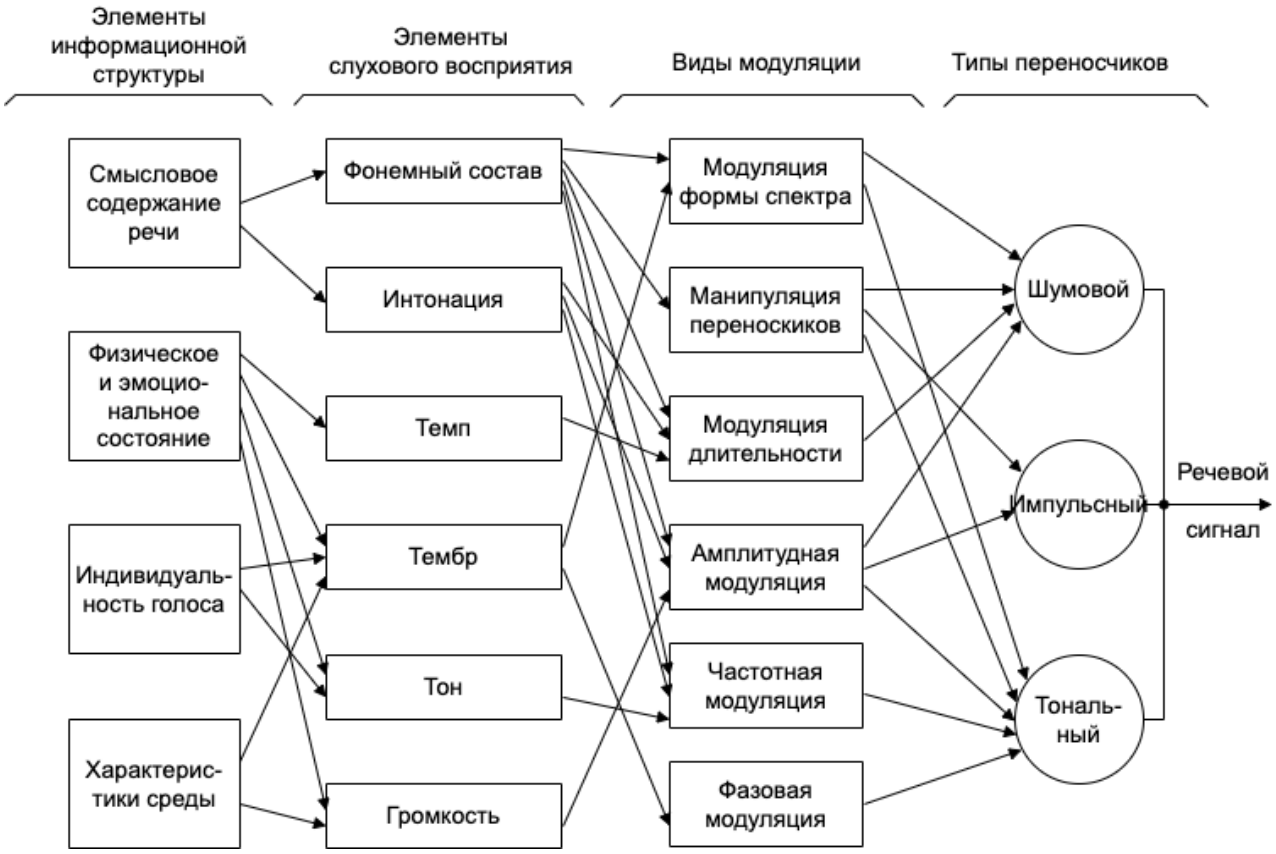
Основные понятия аудио и речевого интерфейса также тесно связаны с основными его характеристиками *Рисунок. Сегментные и надсегментные характеристики речевого сигнала* (см. *Okko R.LinguSoS-2022el*), которые можно разделить на следующие основные группы понятий:
- характеристики речевого сигнала;
- лингвистические характеристики сигнала;
- паралингвистические характеристики сигнала;
- экстралингвистические характеристики сигнала;
- сегментные характеристики речевого сигнала (Segmantal Features);
- надсегментные характеристики речевого сигнала (Suprasegmental Features);
- громкость речевого сигнала;
- тембр речевого сигнала;
- темп речевого сигнала;
- частота основного тона сигнала;
- фонемный состав речевого сигнала.

Основными понятиями предметной области, лежащими в семантической окрестности функционального назначения речевого сигнала и обработки аудиосигналов, являются следующие:
- анализ речевого сигнала;
- детектирование наличия ключевых слов (Key Words Spotting);
- активация по ключевому слову (Wake Up Word Detection);
- детектирование наличия речевого сигнала (Voice Activity Detection);
- синтез речевого сигнала;
- синтез текста в речь (Text-to-Speech Synthesis);
- эмоциональный синтез текста в речь (Emotional Text-to-Speech Syntehsis);
- синтез пения (Sing Synthesis);
- распознавание эмоций в речевом сигнале (Emotional Speech Recognition);
- распознавание диктора;
- сепарация речи (speech separation, speech diarization);
- классификация диктора (Speaker Recognition);
- верификация диктора (Speaker Verification);

Необходимо отметить, что вышеобозначенные понятия зачастую сложным и нетривиальным образом связаны между собой в процессе перехода от источников информации к непосредственным физическим параметрам. Такую сложную структуру сигнала можно представить в виде схемы его информационной структуры *Рисунок. Информационная структура речевого сигнала* (см. *Лобанов Б.М..РечевИИСУП-2006кн*). Этот факт требует от *интеллектуальных компьютерных систем нового поколения* формализации понятий для того, чтобы система могла автоматически интерпретировать взаимосвязи между данными характеристиками при работе с аудио и речевыми сигналами. И, как следствие, выдать ответ пользователю, объясняющий на основе каких характеристик система пришла к тому или иному выводу.

Поскольку для построения *интеллектуальных компьютерных систем нового поколения* в фокусе лежат именно задачи, связанные с обработкой речевых сигналов, решение которых необходимо в первую очередь для построения речевого интерфейса, акцент при формализации постараемся сделать на особенности данной предметной области.

Рисунок. Информационная структура речевого сигнала
=



Основу *sc-модели базы знаний* составляет иерархическая система предметных областей и соответствующих им онтологий. Ниже показан верхний уровень иерархии части *базы знаний*, относящейся непосредственно к аудио и речевым интерфейсам.

Приведем формализованное представление некоторых из обозначенных выше понятий:

сигнал
⇒ *определение**:
[физический процесс, несущий сообщение (информацию) о каком-либо событии, состоянии объекта наблюдения либо передающий команды управления, оповещения и так далее**]**
⊃ *акустический сигнал*
⇒ *определение**:
[сигнал, представляющий собой распространение упругих волн в газообразной, жидкой или твердой среде**]**
⊃ *аудиосигнал*
≔ **[**слышимый звуковой сигнал**]**
⇒ *определение**:

[акустический сигнал, параметры которого находятся в пределах диапазона значений доступного для восприятия органами чувств человека]
⊃ *акустический сигнал*
⇒ *примечание**:
[диапазон частот аудиосигнала лежит в интервале от 20 до 20 000 Гц.]
⊃ *речевой сигнал*
⇒ *определение**:
[аудиосигнал, который образуется в результате прохождения воздушных потоков через речевой тракт человека. В результате всевозможных акустических преобразований происходит формирование различных звуков речи]
∋ *устная речь*
∋ *речеобразование*
⇒ *примечание**:
[механизм речеобразования человека представляет собой акустическую трубу с динамически изменяющимися параметрами поперечного сечения, возбуждаемую либо квазипериодической последовательностью импульсов, генерируемых голосовыми связками, либо турбулентным потоком воздуха, проталкиваемого сквозь сужения, в разных местах речевого тракта]

В зависимости от модели представления сигнала в ostis-системе также могут быть определены следующие описания основных видов сигналов, применение которых обосновано особенностями природы анализируемого сигнала, а также решаемой задачей анализа:

математическая модель сигнала
⇐ *объединение**:
{• *аналоговый сигнал*
⇒ *определение**:
[сигнал параметры которого можно измерить в любой момент времени]
⇒ *определение**:
[сигнал, у которого каждый из представленных параметров описывается функцией времени и непрерывным множеством возможных значений]
• *дискретный сигнал*
⇒ *определение**:
[сигнал, у которого хотя бы один из представленных параметров описывается конечным множеством возможных значений]
⇐ *объединение**:
{• *дискретный по времени*
• *дискретный по амплитуде*
}
• *цифровой сигнал*
⇒ *определение**:
[сигнал, у которого каждый из представляющих параметров описывается функцией дискретного времени и конечным множеством возможных значений]
⇒ *включает**:
{• *дискретный по времени сигнал*
• *квантованный (дискретный) по амплитуде сигнал*
}
• *периодический сигнал*
• *апериодический сигнал*
• *тональный сигнал*
• *гармонический сигнал*
• *имульсный сигнал*
• *шумовой сигнал*
}

Необходимо отметить, что по причине ограничений на размер материала для характеристик аудиосигнала приведем только иерархию общей их взаимосвязи, поскольку семантика данных понятий вполне характерна и для других областей технических наук и не требует подробных примеров и пояснений.

характеристика аудиосигнала
⇐ *объединение**:
{• *амплитуда сигнала*
• *частота сигнала*

- *фаза сигнала*
- *интенсивность сигнала*
- *длительность сигнала*
- *мощность сигнала*
- *спектр сигнала*
- *осцилограмма сигнала*
  - ⇒ *определение**:
    [функция фиксирующая зависимость изменения характеристик сигнала (в первую очередь амплитуды) от времени]
- *спектрограмма сигнала*
  - ⇒ *определение**:
    [функция, фиксирующая зависимость спектральной плотности мощности аудиосигнала от времени]
- *частота дискретизации сигнала*
  - ⇒ *определение**:
    [значение частоты, с которой производилась дискретизация сигнала по времени в процессе аналогово-цифрового преобразования]
  - ⇐ *типовые значения**:
    {
    - *8000 Гц*
    - *16000 Гц*
    - *22050 Гц*
    - *44100 Гц*
    - *48000 Гц*

    }
- *уровень квантования сигнала*
  - ⇒ *определение**:
    [допустимое количество дискретных уровней сигнала выраженное как степень двойки и применяемое в процессе квантования сигнала по уровню в процессе аналогово-цифрового преобразования]
  - ⇐ *типовые значения**:
    {
    - *8 бит*
    - *10 бит*
    - *12 бит*
    - *16 бит*
    - *24 бита*

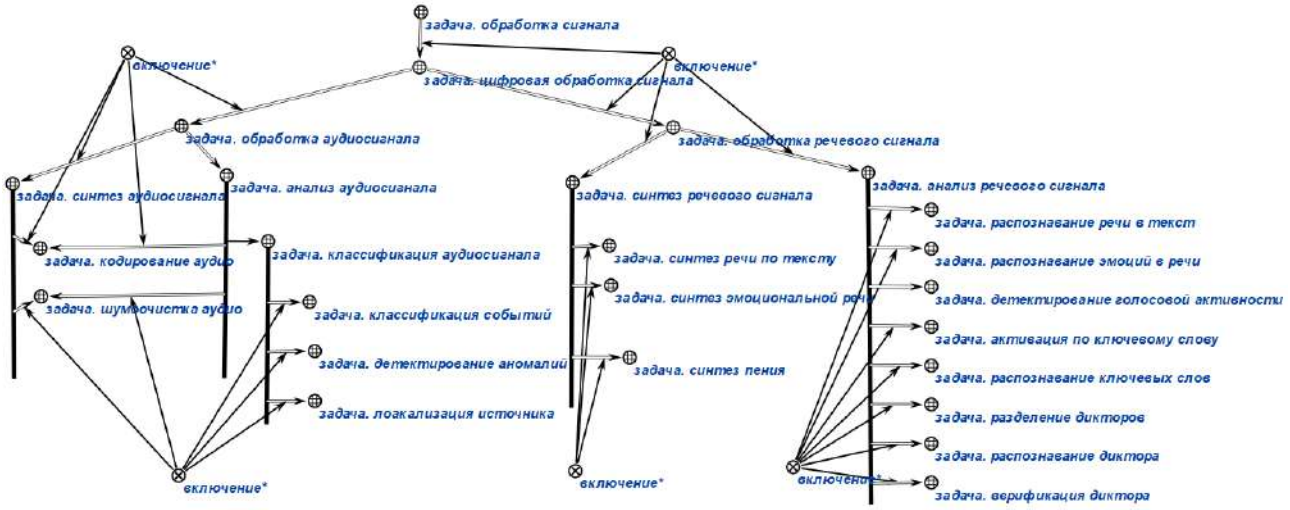
    }

}

Важным аспектом в процессе проектирования является формализация *классов задач* аудиоинтерфейса, поскольку благодаря этим знаниям в интеллектуальной системе могут (и должны) применяться соответствующие методы обработки, в зависимости от типа решаемой задачи. Фрагмент онтологии верхнего уровня типовых *задач* решаемых *интеллектуальной компьютерной системой* в области обработки аудио- и речевых сигналов представлено на *SCg-текст. Фрагмент онтологии верхнего уровня задач аудио- и речевых интерфейсов для интеллектуальных компьютерных систем нового поколения*

SCg-текст. Фрагмент онтологии верхнего уровня задач аудио- и речевых интерфейсов для интеллектуальных компьютерных систем нового поколения

=



Описание предметной области моделей сигнала будут более подробно рассмотрены в следующем пункте работы, поэтому не считаем необходимым приводить ее здесь. Онтологию основных характеристик речевого сигнала формализуем в следующем виде:

характеристика речевого сигнала

⇐ *объединение**:

{• *коммуникативная характеристика*

⇒ *примечание**:

[кодирует смысл передаваемого сообщения, зависит от намерений отправителя]

• *информативная характеристика*

⇒ *примечание**:

[кодирует дополнительную информацию передаваемого сообщения, не зависит от намерений отправителя]

• *информативная характеристика*

⇒ *примечание**:

[кодирует дополнительную информацию передаваемого сообщения, не зависит от намерений отправителя]

• *сегментная характеристика речевого сигнала (Segmantal Features)*

⇒ *примечание**:

[несет информацию о текущем состоянии источника на протяжении длительности одной или нескольких фонетичесих единиц]

• *надсегментная характеристика речевого сигнала (Suprasegmental Features)*

⇒ *примечание**:

[несет информацию о состоянии источника и переходах между ними на протяжении времени всего высказывания]

}

⊃ *лингвистическая характеристика сигнала*

∋ *вербальные средства коммуникации*

∋ *коммуникативная характеристика*

⇒ *определение**:

[характеристика несущая информацию по средствам использованием системы кодирования человеческого языка]

⇒ *примечание**:

[лингвистическая характеристика включают как фонологический код (сегментарный и надсегментный), так и грамматический код (морфологию и синтаксис). Лингвистическая коммуникация информирует получателя о намерениях отправителя с помощью явных словесных форм]

⊃ *паралингвистическая характеристика сигнала*

∋ *невербальные средства коммуникации*

∋ *коммуникативная характеристика*

⇒ *определение**:
[характеристика несущая информацию посредством дополнительных средств коммуникации не связанных непосредственно с языком]
⇒ *примечание**:
[передает информацию об отношении к предмету разговора, чувствах или эмоциональном состоянии говорящего]
⇐ *объединение**:
{• *интонация речи*
⇐ *объединение**:
{• *частота основного тона* F_0
• *изменение частоты основного тона* ΔF_0
}
• *громкость речи*
⇐ *объединение**:
{• *амплитуда сигнала*
• *интенсивность сигнала*
}
• *темп речи*
• *длительность пауз*
}
⊃ *экстралингвистическая характеристика сигнала*
∋ *информативная характеристика*
⇒ *определение**:
[характеристика, которые не кодируют непосредственно смысл сообщения, но содержит дополнительную информацию об отправителе и условиях коммуникации]
⇒ *примечание**:
[передает информацию об отношении к предмету разговора, чувствах или эмоциональном состоянии говорящего]
⇐ *объединение**:
{• *характеристика голоса диктора*
⇐ *объединение**:
{• *высота*
• *тембр*
• *громокость*
}
• *акустическое окружение*
}

Таким образом, представлен результат формализации средствами *SCg-кода* предметной области и онтологии типовых задач аудио и речевых интерфейсов для *интеллектуальных компьютерных систем нового поколения*.

Необходимо отметить, что выше были представлены варианты формализации ключевых понятий области, на основе имеющихся источников информации. Более полные результаты работы по формализации предметной области *аудиоинтерфейса* требуют доступа к закрытым стандартам *AES*, *ISO* и *IEEE*, а также привлечения более широкого круга экспертов, результаты работы с которыми будут фиксироваться в следующих вариантах стандарта и монографии.

§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала

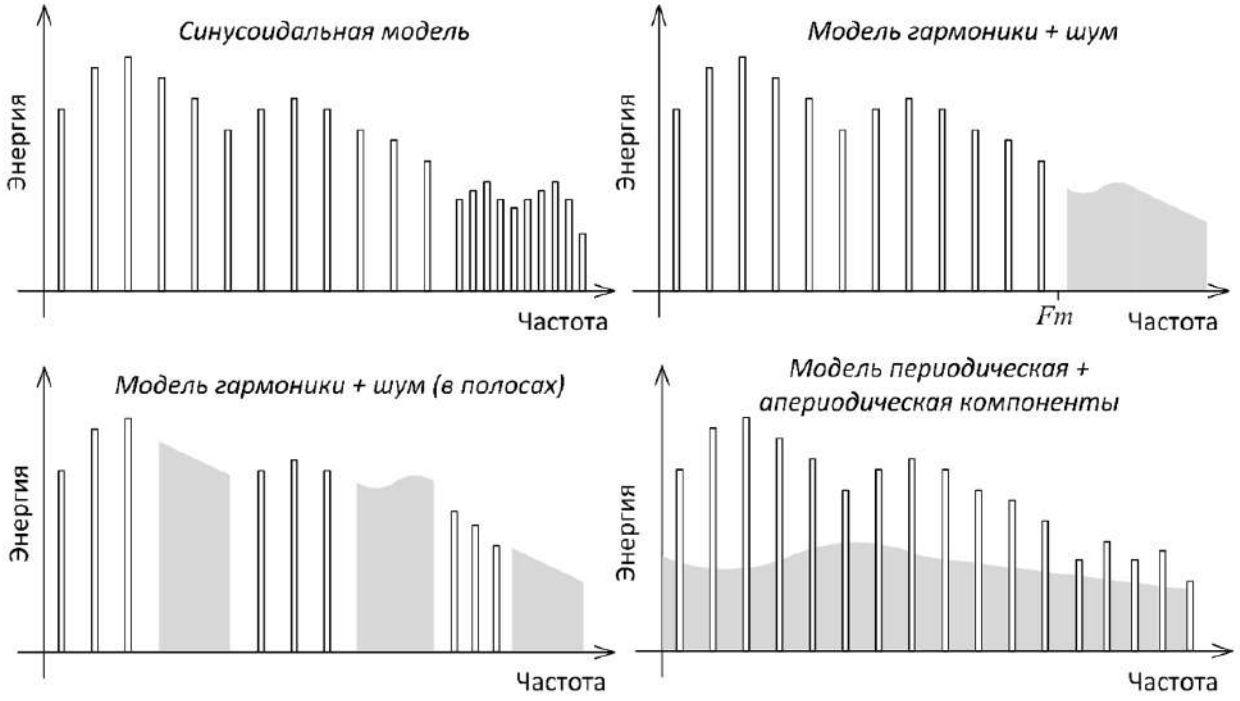
Все вышеперечисленные задачи взаимосвязаны, поскольку относятся к одному и тому же объекту исследования — *речевому сигналу*. Решение каждой из них непосредственно либо косвенно зависит от эффективности моделирования речи как сложного феномена в различных аспектах: параметрическое представление речевого сигнала и выделение его свойств, моделирование процесса фонации, восприятия и интерпретации содержания речевого сообщения (в том числе фонетического, смыслового, эмоционального). Это делает создание универсальных способов обработки речевых сигналов перспективным научным направлением. В контексте перечисленных задач моделирование речи можно условно разделить на три уровня:

- моделирование сигнала в общем виде, используя отсчеты во временной или частотной области;

- моделирование характеристик сигнала, являющихся специфическими для речи и связанных с процессом фонации (таких как частота основного тона, последовательность возбуждения и огибающая амплитудного спектра);
- моделирование высокоуровневых речевых характеристик (голос, акцент, экспрессия, фонетическое и семантическое содержание речевого сообщения). Каждый следующий уровень основывается на предыдущем и подразумевает использование специальных методов параметрического описания.

К первым двум уровням относятся широко известные в цифровой обработке речевых сигналов модели на основе линейного предсказания, кепстральных коэффициентов и синусоидальных параметров. Отличительные особенности основных типов моделей представлены на рисунке *Рисунок. Отличительные особенности основных типов моделей параметрического представления сигнала*

Рисунок. Отличительные особенности основных типов моделей параметрического представления сигнала
=



Среди подходов, использующих синусоидальное описание сигнала, в настоящее время наиболее перспективными являются смешанные (гибридные) модели, учитывающие возможность разных режимов фонации с участием голосовых связок (вокализованная речь) и без участия голосовых связок (невокализованная речь), причем каждый из этих двух режимов описывается соответствующей моделью.

Вокализованная речь рассматривается как квазипериодический (детерминистский) сигнал, в то время как невокализованная — как непериодический (стохастический) сигнал. Наиболее известной среди существующих моделей является модель гармоники+шум, которая используется для решения таких сложных задач, как создание речевых интерфейсов, распознавание речи, синтез речи по тексту, конверсия голоса, шумоподавление, повышение разборчивости и субъективного качества речевых сигналов, коррекция акцента и так далее. Ее преимуществом является теоретическая возможность моделирования вокализованных звуков в виде непрерывных функций с изменяющимися параметрами, что позволяет получить эффективное описание процесса фонации и избежать наложения смежных фрагментов, разрыва фаз при синтезе речи. Недостатком модели является высокая сложность алгоритмов анализа и синтеза, обусловленная нестационарностью речевого сигнала (см. *Serra X..aSystefSATS-1990art*, *Griffin D.W..MultiEV-1988art*, *Petrovsky A..HybriSDBoIH-2011art*, *Azarov E..InstaHRoSUM..-2013art*).

Поскольку вокализованная речь состоит из квазипериодических компонент с изменяющимися параметрами, для анализа необходимо использовать цифровые фильтры с изменяющимися характеристиками: их полоса пропускания должна меняться в соответствии с контуром частоты основного тона. Это требует использования специальных частотно-временных преобразований, позволяющих производить оценку периодических составляющих с сильной частотной модуляцией, таких как Фан-Чирп и гармоническое преобразования. Точность оценки параметров

непосредственно связана с точностью оценки контура основного тона, поэтому использование надежного и точного способа оценки является необходимым условием для успешного использования данной модели (см. *Laroche J..HNSSMBoaHNM-1993art*, *McAulay R..SpeecASBoaSR-1986art*, *Degottex G..MixedSMaiAVT-2013art*).

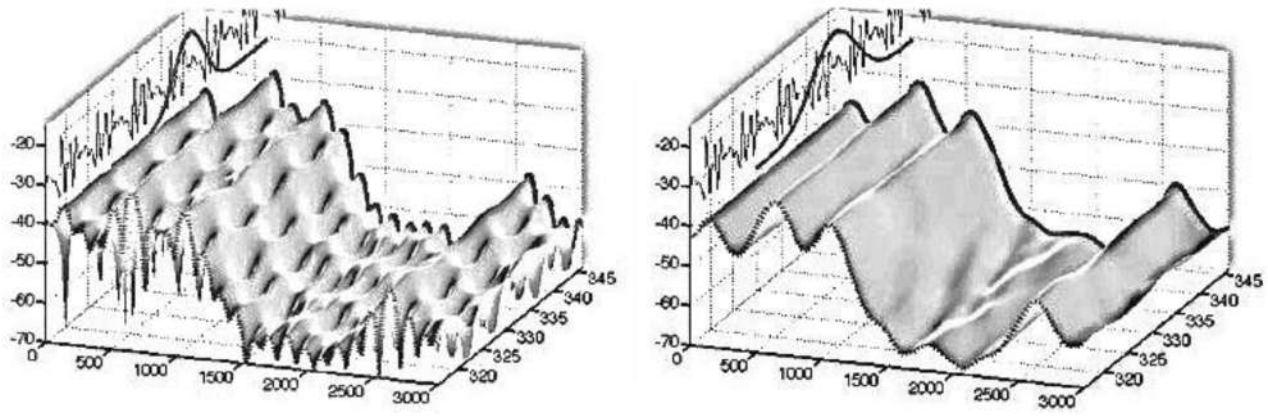
Также сложной задачей является автоматическое разделение сигнала на детерминистскую и стохастическую составляющие, для чего используются специальные детекторы периодичности.

Моделирование речевого сигнала на основе линейного предсказания является классическим подходом, который применяется в цифровой обработке речи достаточно продолжительное время. Основным преимуществом модели является раздельное описание сигнала в виде огибающей спектра и сигнала возбуждения. Огибающая спектра определяет фонетику произносимого звука и характеризует состояние речевого тракта, в то время как сигнал возбуждения характеризует состояние голосовых связок и высоту (интонацию) вокализованных звуков. Преимуществом линейного предсказания является также низкая вычислительная сложность.

Однако, несмотря на это, в последнее время предпочтение отдается моделям, использующим синусоидальное представление сигнала и в первую очередь это касается приложений, подразумевающих синтез речевого сигнала с измененными параметрами, таких как изменение интонации, конверсия голоса, синтез речи по тексту и других. Данный факт можно объяснить тем, что линейное предсказание не обеспечивает эффективных способов для параметрической обработки сигнала возбуждения и непрерывного синтеза выходного сигнала. Каждый речевой фрагмент (кадр) сигнала представляет собой отдельную независимую единицу и при синтезе возникает проблема согласования соседних кадров. Несогласованное изменение огибающей амплитудного и фазового спектра при переходе от кадра к кадру вызывает появление слышимых артефактов. Кроме того, оценка огибающей спектра при помощи классических методов линейного предсказания представляет собой усреднение по всему кадру, вследствие чего ее точность ограничена. Порядок предсказателя определяет сложность модели: для предсказателей низких порядков оценка огибающей спектра получается чрезмерно сглаженной, а для предсказателей высоких порядков точность становится избирательной. Для точек спектра, соответствующих гармоникам основного тона, точность повышается, а для всех остальных точек — снижается. Оптимальный порядок предсказателя зависит от высоты голоса, но даже в наиболее благоприятном случае точность оценки огибающей спектра имеет погрешность, приводящую к возникновению слышимых искажений.

Использование кепстральных коэффициентов для моделирования речевых сигналов также является классическим подходом. Наиболее хорошо разработанной системой моделирования речевых сигналов, использующей кепстральные коэффициенты, является *TANDEM–STRAIGHT* (см. *Kawahara H.Explo otOAoVR-2010art*, *Kawahara H..Devel oERTB-2009art*). Так же как и для классических способов анализа на основе линейного предсказания, при оценке кепстральных коэффициентов предполагается стационарность сигнала на протяжении интервала наблюдения. Оценка огибающей амплитудного спектра требует сглаживания и также недостаточно точна по сравнению с моделями на основе синусоидальных *Рисунок. Спектрограмма сигнала на основе ДПФ (слева) и спектрограмма TANDEM-STRAIGHT (справа).*

Рисунок. Спектрограмма сигнала на основе ДПФ (слева) и спектрограмма TANDEM-STRAIGHT (справа)
=



Благодаря своим широким возможностям гибридная модель на основе синусоидальных параметров является наиболее предпочтительной для использования в большинстве практических случаев. Тем не менее для преодоления существующих ее ограничений, связанных со сложностью оценки параметров, их интерпретации в виде специфиче-

ских речевых характеристик (параметры речевого тракта, последовательность возбуждения) требуется разработка специальных методов моделирования.

В зависимости от приложения процесс обработки речевого сигнала с использованием той или иной модели обычно включает анализ (определение параметров модели), модификацию (изменение параметров модели в зависимости от цели приложения) и синтез (формирование нового сигнала из измененных параметров модели). Таким образом, для обеспечения наиболее высокой практической значимости разрабатываемые методы моделирования должны включать средства анализа, обработки параметров и синтеза.

Для решения многих современных прикладных задач требуется не только наличие возможности описания речевого сигнала или процесса фонации, но и использование высокоуровневых речевых характеристик, определяющих персональный голос диктора, экспрессию, фонетику и так далее. К таким задачам относятся конверсия голоса, синтез речи по тексту, верификация диктора и многие другие. Высокоуровневое моделирование речи является очень сложной предметной областью, поскольку требует использования интеллектуальных моделей и методов машинного обучения. На данный момент не существует единого универсального способа, применяемого для разных приложений.

Чтобы подвести итог изложенным в разделе идеям приведем формальное представление некоторых из обозначенных выше концепций:

параметрическая модель сигнала
⇒ *определение**:
[математическое выражение, используемое для представления отсчетов сигнала во временной или частотной области]
⊃ *параметрическая модель речевого сигнала*
⇒ *определение**:
[математическое описание характеристик сигнала, являющихся специфическими для речи и связанных с процессом фонации (таких как частота основного тона, последовательность возбуждения и огибающая амплитудного спектра)]
⇒ *примечание**:
[к основным моделям речевого сигнала относят: модели на основе линейного предсказания; на основе кепстрального представления; синусоидальные и гибридные модели. Среди гибридных моделей наиболее известна модель гармоники+шум]

Подавляющее большинство высокоуровневых речевых моделей, используемых на практике, являются проблемно-ориентированными и могут применяться только для решения одной, узкоспециализированной задачи. Поэтому критически важным является факт того, чтобы в *базе знаний интеллектуальной компьютерной системы* содержалось достаточно информации для определения подходящего типа модели представления сигнала и активации соответствующего *агента* в рамках *многоагентной системы* в зависимости от класса м модели решаемой задачи. Более подробно соответствующие понятия рассмотрены в рамках *Главы 3.1.*

Заключение к Главе 4.3.

В главе изложены идеи, лежащие в основе оригинального подхода к проектированию аудиоинтерфейсов *интеллектуальных компьютерных систем* на основе онтологического проектирования и формализации системы понятий из соответствующей предметной области, с использованием Технологии OSTIS. Изложены основные принципы лежащие в основе данного подхода, а также их отличительные особенности от общепринятых.

К ограничениям предлагаемого подхода можно отнести следующие основные факторы. Очевидно, что для достижения поставленной цели и реализации задач по формализации любой предметной области, в том числе и аудиоинтерфейсов, требуется в первую очередь большое количество источников знаний для их пополнения. Для преодоления данной проблемы требуется привлечение большого количества экспертов, обладающих соответствующими компетенциями и знаниями в предметной области, либо же разработка механизмов надежного автоматического извлечения этих знаний из имеющихся источников.

Прямой доступ к знаниям экспертов весьма ограничен, поскольку это требует значимых усилий по подбору репрезентативной выборки таких экспертов, выстраиванию эффективных и интероперабельных взаимоотношений между сторонами процесса, что зачастую зависит от большого количества субъективных факторов, — соответственно, требует большого количества временных и материальных ресурсов.

Известно, что существенное количество информации, накопленной человечеством, хранится в виде текстов на естественных языках. Процесс извлечения данной информации и представления ее в формализованном виде — в виде знаний, также выглядит нетривиальным.

Исходя из природы данных проблем, по мнению авторов, видятся следующие основные направления их преодоления и, как следствие, две основные стратегии развития предложенного подхода.

1. Создание для экспертов, работающих в домене аудио и речевых интерфейсов, специализированных инструментальных средств по формализации и представлению знаний из данной предметной области, фиксации их в виде стандартов единой формы. Подобные инструменты должны обладать качественно новыми функциональными возможностями, обеспечивающими высокий уровень совместимости и интероперабельности в процессе накопления и стандартизации знаний, чтобы сами эксперты были заинтересованы в применении и широком распространении данной технологии для представления знаний. Данный пункт является одной из ключевых задач *Технологии OSTIS* и *Стандарта OSTIS*.
2. Создание автоматизированных и автоматических средств извлечения знаний из существующих источников информации, в первую очередь текстов на естественном языке. К видам документов, в которых содержится уже структурированная и отчасти формализованная информация, относятся в первую очередь: стандарты, протоколы, рекомендации (RFC), инструкции и так далее. Следовательно процесс автоматизации извлечения знаний должен быть направлен в первую очередь на формализацию уже существующих отраслевых стандартов разработки аудиоинтерфейсов, систем обработки и кодирования аудиоинформации, систем обработки речевых сигналов, таких как стандарты серии *ISO*, *IEEE* и *AES* (Audio Engineering Society): *ISOIECITCoAVO-2005art*, *ISOIECITMAT-2020art*, *IEEESfSGAC-2020el*, *IECAVaRE-2015el*, *AEST3250-2004art*.

Реализация подхода, предложенного в данной главе, позволит обеспечить свойства унификации, семантической совместимости и интероперабельности, при разработке аудио и речевых интерфейсов (своеобразный аналог *Модели OSI/ISO* в области проектирования *интерфейсов интеллектуальных компьютерных систем*), что в итоге позволит существенным образом сократить издержки при создании интеллектуальных компьютерных систем нового поколения для решении сложных комплексных задач.

Глава 4.4.
3D-модель внешней среды в ostis-системах

⇒ *авторы**:
- *Головатая Е.А.*
- *Головатый А.И.*

⇒ *аннотация**:
[Данная глава посвящена рассмотрению вопросов построения и использования трехмерного представления в различных задачах прикладных *интеллектуальных компьютерных систем*, а также соответствующих систем позиционирования и ориентации в пространстве. Описание самого представления, а также принципов его построения осуществляется на основе *базы знаний ostis-системы*, что позволяет проводить глубокую интеграцию различных задач и методов, а также впоследствии приводит к повышению степени конвергенции различных направлений.]

⇒ *подраздел**:
- *§ 4.4.1. Прикладные задачи трехмерной реконструкции*
- *§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции*
- *§ 4.4.3. Предлагаемый подход к трехмерной реконструкции с использованием базы знаний ostis-системы*
- *§ 4.4.4. Семантическое представление объектов и сцены*
- *§ 4.4.5. Трехмерное представление объектов в сцене*
- *§ 4.4.6. Трехмерная реконструкция объектов окружающего мира*
- *§ 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции*
- *§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции*
- *§ 4.4.9. Онтология действий, выполняемых в предметной области трехмерной реконструкции*

⇒ *ключевое понятие**:
- *сцена в трехмерном представлении*
- *трехмерное представление объекта*
- *трехмерная модель объекта*
- *трехмерная реконструкция*
- *система локального позиционирования*
- *компьютерное зрение*
- *оптическая система компьютерного зрения*
- *локальный признак изображения*

⇒ *библиографическая ссылка**:
- *Luhmann T..CloseRPa3DI-2018bk*
- *Halavataya K..3DRep oOiNGICS-2022art*
- *Kontakis K..DECOaOFaI3DI-2014art*
- *Flotyski J..OntolBRaMoS-2017art*
- *Huang R..JointRLfTa3DPC-2023art*
- *Берестнева В.Г..СетевКМПКМ-2022cm*
- *Hervas R..aConteMBoOLaPfIV-2010art*
- *Davies E.R..AdvanMaDLiCV-2021bk*
- *Головатая Е.А.МоделФИдТРСпДВИ-2019cm*
- *Krig S.CompuVM-2016bk*
- *Rosten E..MachiLfHS-2006art*
- *Harris C..aCombiCaED-1988art*
- *Smith S.M..SUSAN aNAtLL-1997art*
- *Lowe D.G.DistiIFfSIK-2004art*
- *Mair E..Adapt aGCDBotAST-2010art*
- *Rosten E..Faste aBaMLA-2010art*
- *Verdie Y..TILDE aTILD-2015art*
- *Bay H..SpeedURF-2008art*
- *Calonder M..BRIEF-2010art*
- *Rublee E..ORB:aEAtS-2011art*

- *Leutenegger S..BRISKBRISK-2011art*
- *Lucas B..aIteraIRTwaAtSV-1981art*
- *Horn B.DeterOF-1981art*
- *Halavataya K.LocalFDIfIMaODiRTA-2020art*
- *Halavataya K..AdjusV3DRRUTD-2022art*

§ 4.4.1. Прикладные задачи трехмерной реконструкции

Неотъемлемой частью проектирования *интеллектуальных кибернетических систем* является организация взаимодействия отдельных компонентов и составляющих. Организация данного взаимодействия относится к области проектирования интерфейсов, которые могут быть диверсифицированы относительно объектов и принципов взаимодействия (подробнее в *Главе 4.1. Общие принципы организации интерфейсов ostis-систем*)

Многие компьютерные системы основаны на визуальном взаимодействии с внешней средой. Визуальное взаимодействие с внешней средой осуществляется посредством *системы технического зрения*. Под внешней средой в этом случае понимается совокупность объектов, информация о которых доступна *системе технического зрения*. С точки зрения обработки информации хорошо изученными являются двумерные представления окружающей среды, но более естественными для человека и более удобными (а часто и необходимыми) для использования во многих областях являются трехмерные представления, обладающие информацией о физическом положении, форме и размерах объектов в трехмерном пространстве. Работа с трехмерными представлениями вызывает необходимость поддержки внутреннего аналогичного представления в рамках системы, а также разработки средств его получения, то есть проектирование 3D-интерфейса *интеллектуальной кибернетической системы*.

Таким образом, для взаимодействия *интеллектуальной кибернетической системы* с объектами внешней среды в прикладных задачах необходимо создание внутреннего представления этих объектов и самой среды. Одним из видов такого внутреннего представления может являться описание объектов в виде *трехмерной модели*. При этом формирование такого внутреннего представления относится к классу задач анализа сенсорной информации и требует определения точного расположения объекта, на основании которого прикладные системы могут реализовывать различные сценарии взаимодействия. В соответствии с этим особую важность приобретают системы ориентации в пространстве и *трехмерной реконструкции*, способные формировать и обрабатывать подобные представления.

К прикладным областям, в которых требуется решение подобных задач, можно отнести (см. *Luhmann T..CloseRPa3DI-2018bk*):
- Интеллектуальные робототехнические системы. Примеры задач: анализ окружения, построение траекторий движения, восстановление трехмерных координат объектов.
- Интеллектуальные системы управления производством. Примеры задач: анализ деформаций объектов и структурных изменений, бесконтактное измерение размеров объектов произвольного масштаба и конфигурации, контроль производственного процесса.
- Интеллектуальные системы комплексного медицинского мониторинга и обслуживания. Примеры задач: анализ результатов обследований, отслеживание динамики развития заболеваний, планирование лечения.
- Научно-исследовательская деятельность.
- Другие прикладные области (архитектура, картография и так далее).

Предметная область *трехмерного представления объектов* затрагивает, как само описание объекта, так и методов его получения. На основании данных представлений могут быть решены следующие классы задач:
- построение *трехмерного представления объекта*, группы объектов или окружения;
- определение размеров объекта, включающее и определение отклонений от заданного шаблона или параметров, например, в системах медицинской диагностики;
- проведение дополнительных построений, доопределяющих созданное сформированное *трехмерное представление объекта*;
- построение траектории движения;
- и так далее.

Несмотря на то, что часть указанных выше задач требует дополнительных действий, все они основываются на получении *трехмерного представления объекта*.

Таким образом, декларативной формулировкой задачи *трехмерной реконструкции* является получение внутреннего представления объекта, относящегося к классу *трехмерных представлений объектов*.

§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции

На данный момент существует большое количество методов реконструкции, оперирующих разными представлениями входных данных и информации: фотограмметрическое восстановление по фото/видеосъемке, радиочастотные методы в разных диапазонах, магнитные и инерциальные методы, нейросетевой анализ и так далее. Например, искусственные нейронные сети, решающие задачу *трехмерной реконструкции*, могут принимать в качестве входного значения отдельные изображения, наборы изображений, панорамные и стереоскопические изображения, комбинацию изображений и данных с различных видов датчиков, наборы ключевых точек, воксельное облако. Для каждого метода может быть определен набор характеристик внешней среды (помещение, освещение, наличие движения) и входного представления (размер, тип поверхности), в пределах которых данный метод является корректным и демонстрирует наилучшие результаты работы по одному из целевых критериев. Полученное внутреннее представление также может отличаться: часть методов позволяет восстановить внутреннюю структуру, другие — только поверхность (внешнюю форму) объекта.

Кроме этого в задачах анализа сенсорной информации, как правило, есть возможность установить несколько разных типов датчиков, но они должны быть, во-первых, пригодны для исследования данного типа объекта, а, во-вторых, полученная информация должна дополнять друг друга (увеличивать детализацию модели, разрешающую способность, точность и так далее), то есть система должна иметь возможность адаптироваться под конкретную задачу и внешние условия.

Все эти факторы накладывают серьезные ограничения на возможность применения различных методов *трехмерной реконструкции* при решении конкретной прикладной задачи, которые должны быть учтены при проектировании системы (см. *Halavataya K..3DRep oOiNGICS-2022art*).

К проблемам существующих решений можно отнести:

- Отсутствие согласованности систем понятий и описаний методов в различных источниках. Встречаются разные описания одного и того же метода, существует много модификаций, постоянно возникают сложности и, как минимум, недопонимания в связи с этим.
- Отсутствие привязки и недостаточное внимание вопросам *конвергенции Предметной области трехмерной реконструкции* с *Предметной областью формирования трехмерных сцен и окружений интерактивного пользовательского взаимодействия с трехмерной средой*, например, в системах трехмерного моделирования, виртуальной и дополненной реальности.
- Высокая сложность разработки прикладных систем, использующих методы *трехмерной реконструкции*, и необходимость привлечения опытных высококвалифицированных разработчиков к решению соответствующих задач.
- Отсутствие комплексной технологии проектирования. Несмотря на обилие алгоритмов и методов, анализ их применимости к различным видам прикладных задач крайне поверхностный. Вследствие этого в большинстве случаев лучший метод выбирается перебором или эмпирически.
- Отсутствие средств интеграции отдельных компонентов, этапов различных методов, различного типа данных при описании объекта и полученных внутренних представлений. Кроме вариативности отдельных действий, в результате каждый метод работает со своим классом внутреннего представления (поверхность, заданная полигонально, воксельное облако с регулярной сеткой, набор отдельных координат в трехмерном пространстве и так далее).

Таким образом, систематизация знаний данной предметной области, а также создание технологии разработки и автоматизации процесса проектирования *интеллектуальных кибернетических систем* в данной области являются актуальными и нерешенными на данный момент задачами.

Кроме этого, данные подходы изначально не учитывают предметные области объектов исследования, что в свою очередь может значительно влиять на качество полученных результатов. Существует ряд работ, подтверждающих целесообразность использования онтологического *трехмерного представления объектов*, расширенного знаниями из предметных областей. Например, в работе *Kontakis K..DECOaOFaI3DI-2014art* авторы представили интегрированную структуру для оформления интерьера, объединяющую технологии *Web3D* с *SVG графикой* и онтологиями *Semantic Web*. Система основана на онтологической структуре *OWL*, описывающей весь спектр концепций дизайна интерьера, а также использует рекомендательную систему SWRL, способную автоматически предлагать клиентам решения визуального дизайна для их нужд с использованием *интеллектуальной системы*, основанной на правилах. В работе *Flotyski J..OntolBRaMoS-2017art* представлен более полный обзор современного состояния представления и моделирования 3D-контента на основе онтологий семантической сети, а также классификация, характеристика и обсуждение конкретных подходов. В работе отмечается также важность использования гибридных представлений, позволяющих сопоставлять и объединять понятия, относящиеся к визуальному и смысловому уровням абстракции. Описанные подходы и инструменты предоставляют широкие возможности при работе с *трехмерными представлениями объектов*, однако в ходе анализа отмечается все еще общая недостаточность использования возможностей вывода новых знаний из представлений контента, при этом многие системы используют онтологии только в качестве описаний метаданных. Кроме этого, как правило, не описывается возможность реализации систем, задачей

которых является не явное трехмерное моделирование, а использование *трехмерных представлений объектов* в процессе реализации взаимодействия и интерфейсов *интеллектуальных кибернетических систем*.

§ 4.4.3. Предлагаемый подход к трехмерной реконструкции с использованием *базы знаний ostis-системы*

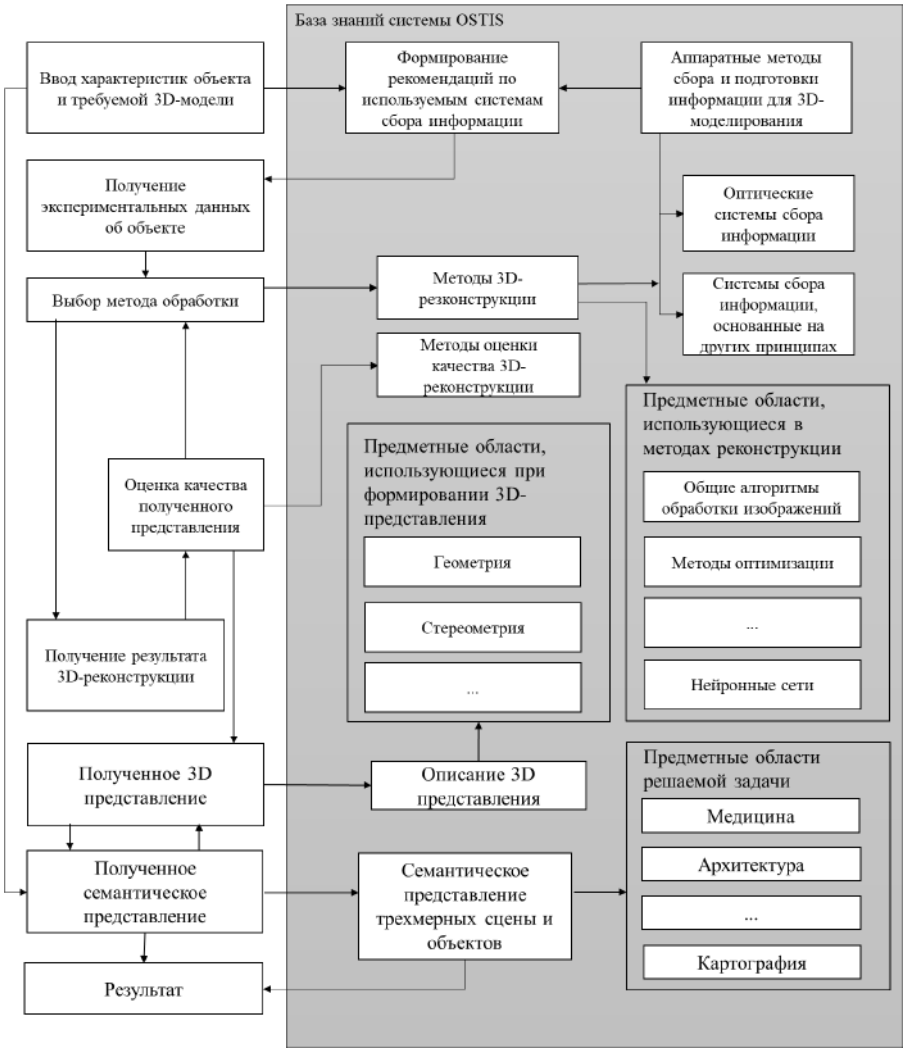
Описание самого представления, а также принципов его построения осуществляется в виде *базы знаний ostis-системы*. В рамках формирования *базы знаний* и платформы для разработки *интеллектуальных кибернетических систем* в данной *предметной области* должны быть решены следующие задачи:

- выделение семантического представления трехмерных сцен;
- систематизация предметной области, существующих подходов и установление связей со смежными областями;
- разработка набора агентов, определяющих подходящие методы и средства для конкретных прикладных задач;
- разработка набора агентов, проводящих агрегацию разных методов, с целью уточнения или проверки параметров трехмерного представления (положения) объекта.

Последовательность основных этапов процесса *трехмерной реконструкции* и их связь с *базой знаний ostis-системы* представлена на рисунке *Рисунок. Описание основных этапов процесса трехмерной реконструкции и их связи с базой знаний ostis-системы трехмерного представления объектов*. В рамках *базы знаний ostis-системы трехмерного представления объектов* выделены следующие блоки и взаимосвязи:

- описание характеристик наблюдаемых объектов;
- описание типов и принципов задания *трехмерных представлений объектов*;
- описание физических принципов работы и спецификаций оборудования, с помощью которого может быть собрана информация об исследуемом объекте;
- набор различных методов реконструкции и локализации с ограничениями и решателями конкретных задач;
- методы оценки результатов полученных *трехмерных представлений объектов*;
- описание семантического представления трехмерных сцен и объектов.

Рисунок. Описание основных этапов процесса трехмерной реконструкции и их связи с базой знаний ostis-системы трехмерного представления объектов
=



Далее более подробно рассмотрим основные указанные *предметные области* и *онтологии*.

§ 4.4.4. Семантическое представление объектов и сцены

Важной составляющей *интеллектуальных кибернетических систем*, использующих внутреннее *трехмерное представление объектов*, является описание семантики этого представления. Информация об индивидуальных точках, поверхностях, полигонах или других примитивах не предоставляет информацию о смысловом наполнении этого представления — точно так же, как отдельные буквы не позволяют оценить семантическую составляющую текстовых сообщений. *ostis-система* предоставляет возможность формирования связанного семантического представления объекта, использующего дополнительную информацию из *предметной области* рассматриваемого объекта.

В работе *Huang R..JointRLfTa3DPC-2023art* продемонстрирована возможность использования такого комбинированного представления в задачах машинного обучения и распознавания образов. Данное представление также является основой для задач создания контента, в том числе основанных на генеративных состязательных нейросетевых моделях. Кроме этого, сопоставление пространственных и семантических свойств объекта способствует расширению функциональности и точности человеко-машинного взаимодействия в системах дополненной и виртуальной реальности, например, при проектировании обучающих или вспомогательных приложений (см. *Берестнева В.Г..СетевКМПКМ-2022cm*, *Hervas R..aConteMBoOLaPfIV-2010art*).

Семантическое описание объекта подразумевает ассоциацию множества точек, соответствующего некоторому объекту в трехмерном пространстве, с некоторым объектом имеющейся *базы знаний*. Отношения в таких ассоциациях

могут быть представлены в виде “объект A представляет собой трехмерный образ некоторой сущности B”, где объект A задается внутренним *трехмерным представлением объекта*, а сущность B — некоторым описанием в *базе знаний*.

Некоторые свойства сущности B, ассоциируемой с объектом A, могут быть естественным образом выведены из *трехмерного представления этого объекта* — в частности, информация о форме, свойствах поверхности, окраске и текстурировании может уже присутствовать в трехмерном описании. Таким образом, с помощью дополнительной обработки *трехмерное представление объекта* A может являться источником фактологической информации о связанной сущности B. В то же время знание свойств сущности B может помочь при формировании более точного трехмерного образа объекта A.

Аналогично семантическому описанию отдельного объекта может быть введено семантическое описание составных объектов. Следует иметь в виду, что семантическое наполнение индивидуальных составляющих не позволяет в полной мере определить семантику всего составного объекта целиком, поэтому аналогичную ассоциацию также необходимо задать и для всей совокупности. Например, объект “велосипед” может быть декомпозирован на составные объекты “цепь”, “рама”, “колесо” и так далее, однако набор семантических описаний *трехмерных представлений объектов* составляющих по отдельности не позволяет формировать или учитывать семантику всего составного объекта целиком.

Семантическое описание части объекта должно содержать как информацию о базовом объекте, так и о дополнительном контексте относительно смыслового наполнения рассматриваемой части. Например, при видеоэндоскопическом анализе участка пищевода важной становится также информация о том, к какой именно части пищевода в организме относится этот участок.

Сцена представляет собой сложную композицию нескольких простых или составных объектов в некотором общем пространстве, дополненную данными об их взаимном расположении. Как и в случае с составными объектами, семантическое описание сцен должно включать не только описания индивидуальных составляющих, но и также смысловое наполнение эмержентных свойств всех этих составляющих в совокупности.

Таким образом, выделены следующие виды основных сущностей в рамках семантического представления сцен и объектов в *трехмерном представлении*:

объект в трехмерном представлении
:= [множество точек в пространстве, соединенных между собой и имеющих одно смысловое представление]

составной объект в трехмерном представлении
:= [*объект в трехмерном представлении*, подразумевающий декомпозицию на отдельные индивидуальные объекты]

часть объекта в трехмерном представлении
:= [множество точек в пространстве, принадлежащих некоторому *объекту в трехмерном представлении*, которое может быть выделено по его геометрическому или смысловому представлению]

сцена в трехмерном представлении
:= [совокупность нескольких *объектов в трехмерном представлении* и данных об их взаимном расположении в пространстве и свойствах, включающих абсолютные и относительные референтные ориентированные и неориентированные отношения]

Для *сцены в трехмерном представлении* важными являются семантические характеристики визуального восприятия сцены с позиции некоторого помещенного в нее наблюдателя или *системы компьютерного зрения*. В этом контексте сцена может быть представлена в виде отдельного наблюдения, например, двумерной проекции (или пары двумерных проекций в случае стереоскопического зрения). При этом семантика описаний исходных *объектов в трехмерном представлении* и соответствующих им участков полученных проекций также может отличаться — например, некоторые объекты могут находиться вне поля видимости, или восприниматься наблюдателем по-другому из-за наличия внешних факторов. К внешним факторам можно отнести наличие оптических, перспективных или психофизиологических эффектов (например, разница в освещении, оптическая дисторсия, различные иллюзии восприятия цвета или глубины, например, комната Эймса, и другие).

Таким образом, важным является тот факт, что семантическое описание сцены должно включать как определение принадлежности индивидуальных объектов сцены некоторым сущностям имеющейся *базы знаний*, так и описание возможных взаимосвязей между этими объектами, возникающих в силу их присутствия в сцене, или в силу их попарного взаимного расположения с позиции некоторого наблюдателя. Эта информация может естественным образом использоваться в качестве референтных свойств объектов сцены. Например, при наличии в сцене двух одинаковых объектов типа A и одного объекта типа B, некоторое логическое высказывание или запрос на естествен-

ном языке может использовать факт их взаимного расположения, например, для идентификации — “тот объект типа А, который находится слева от объекта типа Б”.

внешний фактор наблюдения сцены в трехмерном представлении
:= [параметр наблюдения *сцены в трехмерном представлении*, определяющий пространственные, психофизические, визуальные и другие свойства восприятия системы наблюдения]

наблюдение сцены в трехмерном представлении
:= [описание *сцены в трехмерном представлении* при фиксированном состоянии одного или *нескольких внешних факторов наблюдения*. Может заключаться в понижении размерности представления информации, например, при приведении к двумерной проекции]

референтное отношение описания сцены в трехмерном представлении
:= [абсолютные и относительные ориентированные и неориентированные отношения отдельного *наблюдения сцены в трехмерном представлении*]

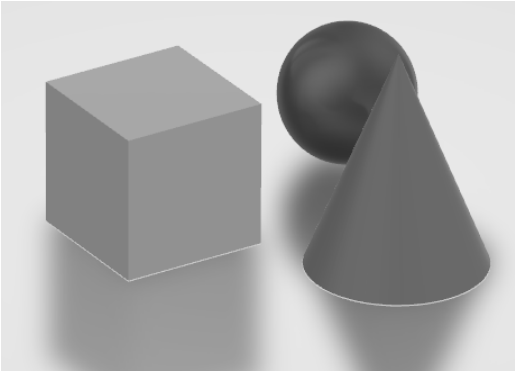
Ниже приведены некоторые возможные *референтные отношения описания сцены в трехмерном представлении* относительно различных *внешних факторов наблюдения*. Следует отметить также, что в общем случае *внешний фактор наблюдения* может быть задан произвольно соответствующей оценочной функцией, например, определения "схожести" объектов.

- информация о присутствии на сцене или проекции сцены:
  - объект отсутствует на сцене;
  - объект присутствует, но не виден на проекции;
  - объект присутствует и на проекции виден частично;
  - объект присутствует и на проекции виден полностью;
- информация о взаимном расположении на проекции сцены:
  - по глубине:
    - за другим объектом;
    - перед другим объектом;
  - по высоте:
    - выше другого объекта;
    - на одном уровне с другим объектом;
    - ниже другого объекта;
  - по расположению вдоль линии горизонта:
    - слева от другого объекта;
    - справа от другого объекта;
- информация об относительном размере объектов на проекции:
  - больше другого объекта;
  - одного размера с другим объектом;
  - меньше другого объекта;
- информация о визуальном сходстве объектов:
  - похож или не похож на другой объект по форме;
  - похож или не похож на другой объект по цвету;
  - похож или не похож на другой объект по размеру.

Пример описания *сцены в трехмерном представлении*, включающий *наблюдение сцены в трехмерном представлении* с точки зрения взаимного расположения объектов, представлен на рисунках *Рисунок. Пример трехмерной сцены, визуализированной в виде двумерной проекции с определенного положения камеры* и *SCg-текст. Семантическое описание некоторых связей взаимного расположения объектов в сцене с положения наблюдателя*.

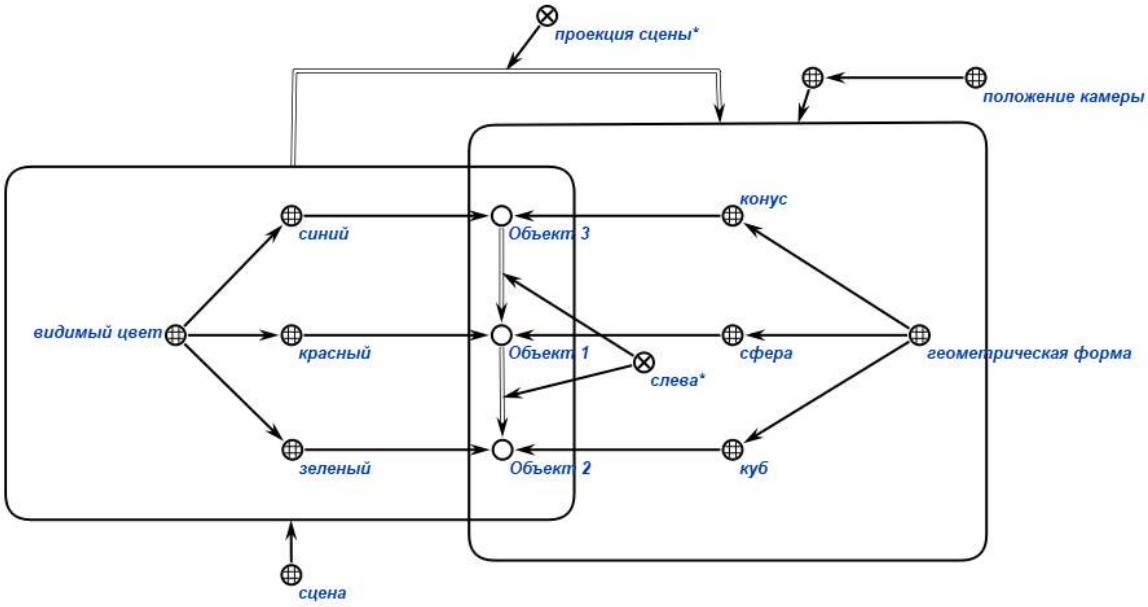
Рисунок. Пример трехмерной сцены, визуализированной в виде двумерной проекции с определенного положения камеры

=



SCg-текст. Семантическое описание некоторых связей взаимного расположения объектов в сцене с положения наблюдателя

=



В данном случае семантическое описание *сцены в трехмерном представлении* является совокупностью абсолютных и относительных характеристик, что позволяет задавать семантическое наполнение различным категориям высказываний. Допустим, объекты на рисунке имеют дополнительную визуальную характеристику цвета: зеленый куб, красная сфера, синий конус. Тогда для идентификации куба может быть приведено абсолютное описание свойства Объекта 2 “видимый цвет” — “зеленый”, но значение данного свойства может быть изменено при введении референтных отношений при таких внешних факторах как “спектральный состав освещения” или “наличие у наблюдателя заболевания, связанного с цветовосприятием”.

На рисунке приведено также референтное описание относительно внешнего фактора “положение камеры” — “куб слева от конуса”, выделяющее отдельное наблюдение сцены, представляющее собой в данном случае двумерную проекцию сцены. Структура *сцены в трехмерном представлении*, основанная на описании присутствующих на этой сцене объектов, а также их свойств и *референтных отношений* относительно *внешнего фактора наблюдения* ложится в основу семантического описания сцены в большинстве задач *компьютерного зрения*. Гранулярность и полнота семантического описания *сцены в трехмерном представлении* определяется конкретной решаемой задачей. Использование такого описания открывает широкие возможности по организации *пользовательского интерфейса*, поскольку предоставляет возможность выведения необходимого контекста для определения смыслового

наполнения высказываний на *естественном языке*. Кроме этого, такое представление может использоваться для генерации *сцен в трехмерном представлении* и формирования интерактивных *интерфейсов* взаимодействия с трехмерной средой.

§ 4.4.5. Трехмерное представление объектов в сцене

Как показано в предыдущем разделе системы позиционирования, распознавания и отображения объектов реального мира опираются не только на качественную составляющую описания, но и взаимное расположение объектов в пространстве или их отдельных частей. В соответствии с восприятием окружающего мира человеком ***трехмерное представление объекта*** является наиболее информативным, хотя и необязательным. Полученные двумерные изображения, используемые во многих задачах, являются проекциями трехмерного пространства. Поэтому для рассматриваемых в данной главе задач максимальным классом объектов исследования выбрано *трехмерное представление объектов*, включающее класс описаний, содержащих информацию о взаимном расположении объектов или их частей, заданных в трех координатах.

трехмерное представление объекта
:= [3D-представление объекта]
:= [способ записи информации об объекте с сохранением его пространственных свойств в трех координатах]
⇒ *разбиение**:
  {• *дискретное трехмерное представление*
  := [трехмерное представление информации об объекте в виде дискретной совокупности трехмерных геометрических примитивов]
  ⇒ *включает**:
    {• *воксельное представление*
    := [представление в виде совокупности элементов, расположенных в регулярной сетке в трехмерном пространстве]
    • *облако точек*
    := [трехмерное представление в виде неструктурированной совокупности точек]
    • *карта глубины*
    := [двумерное изображение, на котором яркость пикселя определяет расстояние от плоскости изображения до соответствующей точки в пространстве]
    }
  • *поверхностное представление*
  := [представление информации об объекте в виде совокупности поверхностей]
  ⇒ *включает**:
    {• *аналитически задаваемая поверхность*
    := [поверхность, заданная в трехмерном пространстве аналитически, например, уравнением]
    • *полигональная сетка*
    := [совокупность вершин, ребер и граней, задающая некоторый многогранник в трехмерном пространстве]
    • *NURBS-поверхность*
    := [поверхность, задаваемая с помощью совокупности неоднородных рациональных B-сплайнов через последовательность контрольных точек]
    • *поверхность разделения*
    := [поверхность, образованная из полигональной сетки с помощью рекурсивного разделения граней этой сетки]
    • *поверхность на основе T-сплайнов*
    := [модификация NURBS-поверхности, для которой последовательность контрольных точек может терминироваться без полного обхода всей поверхности]
    }
  • *специальные представления*
  ⇒ *включает**:
    {• *видео 360*
    := [видеопоследовательность, полученная с помощью захвата окружающего пространства во всех направлениях]
    }
  }

трехмерная модель объекта
:= [3D-модель объекта]
:= [представление информации о конкретном объекте, записанное на основе одного из возможных трехмерных представлений]

Ниже даны определения некоторых терминов, относящихся к предметной области геометрии, стереометрии, функционального анализа и топологии, но необходимых для дополнения описания рассматриваемых видов *трехмерного представления объектов*.

точка
:= [местоположение в пространстве, однозначно определяемое в системе координат]

геометрический примитив
:= [объект трехмерного пространства, который является атомарным (то есть непредставляемый в виде совокупности других объектов)]

вершина
:= [точка в трехмерном пространстве, задаваемая тремя координатами]

ребро
:= [отрезок между двумя вершинами]

грань
:= [замкнутая совокупность копланарных ребер]

полигон
:= [совокупность копланарных граней]

поверхность
:= [двумерное топологическое многообразие]

B-сплайн
:= [базисный сплайн]
:= [сплайн-функция с наименьшим носителем для заданной степени порядка гладкости и разбиения области определения]

§ 4.4.6. Трехмерная реконструкция объектов окружающего мира

Получить *трехмерное представление объекта* на основе экспериментальных данных позволяет ***трехмерная реконструкция*** объекта. *Предметная область трехмерного представления объектов* затрагивает, как само описание объекта, так и методы его получения.

трехмерная реконструкция
:= [задача определения истинной формы объектов в *трехмерном пространстве* на основании информации об этих объектах, получаемой в результате измерений, наблюдений или опытов]

Несмотря на то, что часть указанных выше задач требует дополнительных действий, все они основываются на получении *трехмерного представления объекта*. В соответствии с этим каждая из задач также дополнительно может быть определена как внутренним состоянием, так и описанием условий, в рамках которых она осуществляется:
- описание цели — тип *трехмерного представления объекта*, разрешающая способность и точность;
- условия по возможности осуществления действий — например, имеющееся оборудование или допустимое время выполнения;
- вид входных данных — описание типа входного объекта.

трехмерное представление объекта или окружения, независимо от конкретного подкласса, может быть получено разными методами, в то же время конкретный метод может определять только одно представление.

методы трехмерной реконструкции

⇒ *включает**:

{• *методы фотограмметрической реконструкции*

:= [методы интерпретации изображений для определения *трехмерного представления объекта* и его положения в пространстве по одной или нескольким фотографиям этого объекта]

⇒ *подразделяется по взаимному положению объекта и камеры на**:

{• *мобильные системы*

• *макрофотограмметрия*

• *спутниковая фотограмметрия*

• *аэрофотограмметрия*

• *наземная фотограмметрия*

• *ближняя фотограмметрия*

}

⇒ *подразделяется по виду входной информации на**:

{• *одно изображение*

• *стерео изображения*

• *многокадровые*

}

• *методы томографической реконструкции*

⇒ *включает**:

{• *реконструкция на основе Фурье-проекций*

• *алгоритм обратной проекции*

• *алгоритм итеративной реконструкции*

• *реконструкция по коническому лучу*

• *реконструкция на основе методов глубокого обучения*

}

• *методы структурированного подсвета*

⇒ *включает**:

{• *методы на основе световых сечений*

• *методы на основе проекций полос*

• *методы на основе фазового сдвига*

}

• *методы по оценке отраженного сигнала*

⇒ *включает**:

{• *измерение расстояния методами оптической модуляции*

• *импульсная модуляция*

}

• *методы оценки по фокусировке*

⇒ *включает**:

{• *методы оценки из фокусировки*

• *методы оценки из расфокусировки*

}

• *методы на основе теоремы о Фурье-проекциях*

• *нейросетевые модели*

}

Выше представлен один из вариантов онтологического описания *методов* создания *трехмерных представлений объектов*. Каждый из *методов трехмерной реконструкции* объектов в рамках этого описания может быть определен также кроме физического принципа его разрешающей способностью, типом входных данных, размером реконструируемых объектов и так далее. Полное описание представляет собой неиерархическую онтологическую структуру, строящуюся на основе *базы знаний ostis-системы*. Для дальнейшего взаимодействия агентов с данной структурой все эти описания ставятся в соответствие некоторым характеристикам. Характеристики могут относится как к отдельному методу, так и к группе методов, например, разрешающая способность может быть общей для всего подкласса электромагнитных волновых методов. Данные характеристики должны получаться агентами из самого представления знаний динамически, что позволяет дополнять и модифицировать общую структуру. Для удобства представления данные характеристики можно выделить в спецификацию метода, на основании которой описывается область и возможность его применения. Она дает возможность использования методов для решения конкретных прикладных задач. В рамках спецификации (и, соответственно, структуры представления *методов* в *базе знаний*) задаются:

- тип возможных входных параметров;
- тип выходного представления — в данном случае соответствующее трехмерное представление;
- время работы;

- разрешающая способность метода;
- расстояние от объекта до камеры;
- тип реконструируемого объекта;
- композиция сцены (отдельный объект, группа объектов, окружающее пространство);
- тип поверхности (глянцевость, прозрачность, цветность);
- наличие внутренней структуры;
- размер объекта.

Описанная спецификация может быть интерпретируема промежуточным отношением для каждого метода. При наличии такого описания для промежуточных этапов данный подход позволяет проводить также комбинацию методов. Например, методы радиочастотного диапазона не дают возможность построить карту глубины, но позволяют получить положение камеры в глобальной системе координат в конкретный момент времени.

§ 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции

В основе многих систем *трехмерной реконструкции* лежит решение задачи локального позиционирования. Следует отметить, что в целом задача локального позиционирования является более общей, однако обычно рассматривается относительно наблюдателя, а не объекта.

система локального позиционирования
:= [автоматизированная система, обеспечивающая идентификацию, определение координат, отображение на плане местонахождения отдельных объектов или их частей в пределах территории, охваченной необходимой инфраструктурой]

Характеристики *систем локального позиционирования*:
- точность позиционирования;
- достоверность позиционирования;
- периодичность опроса;
- надежность;
- габаритность.

методы локального позиционирования
⇒ *включает**:
{• *триангуляционные методы*
:= [методы, основанные на определении направления на источник сигнала]
- *угол прихода сигнала*
:= [angle of arrival]
:= [AoA]
:= [направление, из которого принимается сигнал (например, радио, оптический или акустический)]
- *угол вылета*
:= [angle of departure]
:= [AoD]
:= [направление, в котором отправляется сигнал (например, радио, оптический или акустический)]

• *трилатерационные методы*
:= [методы, основанные на трилатерации, то есть на основе построения на местности смежных треугольников]
- *время прибытия*
:= [time of arrival]
:= [ToA]
:= [абсолютный момент времени, когда радиосигнал, исходящий от передатчика, достигает удаленного приемника]
- *разница во времени прибытия*
:= [time difference of arrival]
:= [TDoA]
:= [разница между временем прибытия сигнала до двух базовых станций]
- *время полета*
:= [time of flight]
:= [ToF]

:= [время, затрачиваемое объектом, частицей или волной (акустической, электромагнитной и так далее) на преодоление расстояния через среду]

- *двустороннее определение дальности*
  := [two-way ranging]
  := [TWR]
  := [метод определения дальности, который использует две задержки, которые обычно возникают при передаче сигнала, для определения дальности между двумя станциями: задержка распространения сигнала между двумя беспроводными устройствам, задержка обработки подтверждений в беспроводном устройстве]
- *симметричное двустороннее определение дальности*
  := [symmetrical double-sided two-way ranging]
  := [SDS TWR]
  := [метод определения дальности, который использует двустороннее определение дальности дважды: относительно базовой станции и относительно мобильного устройства]

- *методы на основе измерения силы сигнала*
  := [received signal strength indicator]
  := [RSSI]
  := [индикатор уровня мощности принимаемого сигнала. Данный метод позволяет определить местоположение устройства, основываясь на уровне силы сигнала, полученного базовой станцией или наоборот]
- *одометрия*
  := [использование данных о движении приводов для оценки перемещения]
  - *визуальная одометрия*

}

физические принципы позиционирования

⇒ *включает**:

{• *акустические*
  := [основаны на использовании ультразвуковых (высокочастотных) звуковых волн]
- *радиочастотные*
- *магнитные*
  := [магнитный трекинг основан на измерении интенсивности магнитного поля в различных направлениях. Как правило, в таких системах есть базовая станция, которая генерирует переменное или постоянное магнитное поле]
- *оптические*
  := [совокупность методов позиционирования на основе данных с камер видимого или инфракрасного диапазона]
  - *с внешней камерой*
  - *с внутренней камерой*
  - *визуальная одометрия*
    := [метод оценки положения и ориентации робота или иного устройства с помощью анализа последовательности изображений, снятых установленной на нем камерой (или камерами)]
  - *лазерное позиционирование*
    := [группа методов, основанных на оценке времени прохождения лазерных импульсов определенной периодичности]
- *инерциальные*
  := [инерциальное позиционирование основано на свойствах инерции тел. Основная особенность этих методов состоит в том, что они не требуют внешних ориентиров или поступающих извне сигналов]
  - *гироскоп*
  - *акселерометр*
  - *магнитометр*
  - *барометр*
  - *гибридные*

}

Описание и дальнейшее проектирование различных *систем локального позиционирования* вызывает необходимость использования и других разделов *базы знаний ostis-системы*.

§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции

⇒ *подраздел**:
- *Пункт 4.4.8.1. Оптические системы компьютерного зрения*
- *Пункт 4.4.8.2. Локальные признаки изображений*

компьютерное (техническое) зрение как отдельная область исследований начала формироваться еще в 1960-х годах как попытка имитировать зрительную систему человека в робототехнике. Данное направление изначально основывалось на достижениях в области цифровой обработки изображений, однако в дальнейшем расширило перечень рассматриваемых задач, в том числе засчет попытки извлечения трехмерной структуры изображений с целью более полного понимания сцены. Понимание в этом контексте означает преобразование визуальных образов в описания мира, которые могут взаимодействовать с другими процессами и вызывать соответствующие действия. Считается, что в область *компьютерного зрения* входят задачи распознавания образов, наблюдения, роботизированного управления, сегментации и интерпретации медицинских изображений, автоматического осмотра и сборки в заводских условиях, распознавания отпечатков пальцев и лиц, интерпретации жестов и многие другие (см. *Davies E.R..AdvanMaDLiCV-2021bk*).

компьютерное зрение
:= [техническое зрение]
:= [междисциплинарная предметная область, описывающая совокупность технологий, методов и алгоритмов регистрации, обработки, анализа и получения символьного описания потоков данных с помощью различных источников визуальной информации, включая камеры видимого и инфракрасного спектра, трехмерные датчики и другие устройства визуализации, такие как компьютерные и магнитно-резонансные томографы.]

Пункт 4.4.8.1. Оптические системы компьютерного зрения

В области *трехмерной реконструкции* используются системы, регистрирующие излучение в различных диапазонах длин волн — от рентгеновского (рентгеновский сканер) до радиоволнового (электромагнитная катушка). Наиболее распространенным подходом для анализа является использование излучения в видимом диапазоне (380 нм — 720 нм), регистрируемого *оптической системой компьютерного зрения*, например, камерой (фотоаппаратом).

оптическая система компьютерного зрения
:= [совокупность оптических элементов, обеспечивающая регистрацию электромагнитного поля видимого, ультразвукового и инфракрасного диапазона с целью дальнейшей обработки и получения количественного и символьного описания информации.]

Одним из наиболее распространенных видов *оптической системы компьютерного зрения* для регистрации информации об объекте является цифровая камера. При использовании цифровой камеры формой представления полученных данных является снимок, который может быть представлен в виде растрового изображения. При этом в контексте конкретной рассматриваемой задачи важными могут являться не только параметры полученного изображения, но и фиксация условий проведения съемки и параметров *оптической системы компьютерного зрения*.

Изображение является двумерной проекцией исходного трехмерного пространства. *трехмерную модель* рассматриваемого объекта по нескольким снимкам или видеоряду можно получить на основе восстановления параметров проекций, связанных с каждым из снимков, и выстраиванием этих проекций в одном трехмерном пространстве. При наличии пересекающихся лучей, направленных в одну и ту же точку реального объекта, по проекциям можно восстановить параметры объекта и сцены в связанной с этим пространством системе координат. По схожему принципу устроена зрительная система человека, основанная на анализе разницы между изображениями, получаемыми левым и правым глазом, а также последовательно полученными изображениями на сетчатке одного глаза в небольшом временном диапазоне.

проекция
:= [отображение точек, фигур, векторов пространства любой размерности на его подпространство любой размерности]

бинокулярное зрение
:= [способность одновременно четко видеть изображение предмета обоими глазами; в этом случае человек видит одно изображение предмета, на который смотрит]

стереоскопическое зрение

:= [вид зрения, при котором возможно восприятие формы, размеров и расстояния до предмета, благодаря восприятию двух изображений, полученных в пределах бесконечно малого промежутка времени при разных параметрах внешнего ориентирования оптической системы в пространстве]

Для описания преобразования между объектами в исходном трехмерном пространстве и их проекцией используется модель проекции. Модель проекции при использовании в системах *трехмерной реконструкции* описывает ход световых лучей в *оптической системе компьютерного зрения* и относится к области изучения эпиполярной геометрии. Наиболее известной, использующейся в геометрии является модель ортогональной проекции. Но для работы с *оптическими системами компьютерного зрения* больше подходит модель центральной проекции, которая описывает ход лучей в камере-обскуре. *оптическую систему компьютерного зрения* можно считать аналогичной камере-обскуре, если она содержит симметричную систему линз и не вносит криволинейных искажений, то есть все прямые исходной трехмерной сцены остаются прямыми и не искривляются после проекции. В общем случае модель проекции может учитывать различные особенности *оптической системы компьютерного зрения* или среды получения изображения. Например, в работе *Головатая Е.А.МоделФИдТРСпДВИ-2019ст* предлагается модель широкоугольной сферической проекции, учитывающая дисторсионные искажения, вызываемые широкоугольной камерой. Использование корректной модели проекции напрямую влияет на качество реконструируемых трехмерных сцен и объектов.

эпиполярная геометрия

:= [один из подразделов оптики, который занимается геометрической интерпретацией формирования двумерных изображений по трехмерной сцене. Чаще всего в этом контексте рассматривается задача геометрических преобразований лучей света, которые отражаются от наблюдаемой трехмерной сцены, улавливаются некоторой *оптической системой компьютерного зрения* и проецируются на фоточувствительную двумерную поверхность (например, пленку или цифровую матрицу) для формирования изображения]

модель проекции

:= [математическая модель, в соответствии с которой осуществляется построение проекции]

⇒ *включает**:

{• *параллельная проекция*

:= [вид проекции, при котором проецирующиеся лучи строятся параллельными друг другу]

• *ортогональная проекция*

:= [вид параллельной проекции, в которой проецирующие лучи строятся перпендикулярно некоторой заданной плоскости, например, плоскости изображения]

• *центральная проекция*

:= [вид проекции, при котором все прямые, построенные на проецирующихся лучах, пересекаются в одной точке, называемой центром проекции. Параллельная проекция может считаться предельным случаем центральной проекции с центром проекции, удаленным на бесконечное расстояние]

}

Для полного задания модели проекции вводятся параметры, называемые параметрами ориентирования, однозначно характеризующие положение снимка относительно некоторой глобальной системы координат. Выделяют параметры внутреннего и внешнего ориентирования. Параметры внутреннего ориентирования задают относительное положение точки фотографирования и самого снимка. В модели центральной проекции к параметрам внутреннего ориентирования относятся 3 величины: двумерные координаты главной точки снимка (x_0, y_0) и фокусное расстояние f. К параметрам внешнего ориентирования относят 6 величин, задающих связь проецирующих лучей в момент съемки с глобальной системой координат: координаты точки положения камеры (x_C, y_C, z_C), два угла, определяющие положение главного луча камеры (ω, φ) и угол поворота снимка β. Для перехода в новое пространство необходимо осуществить пространственное преобразование между системами координат. Для этого используется преобразование Гельмерта, зависящее от 7 параметров: координаты начала системы координат снимка xyz в глобальной координатной системе OXYZ (X_0, Y_0, Z_0), три угла поворота (ω, φ, β) относительно осей X,Y,Z и коэффициент масштаба m. На основании данных углов можно построить матрицу поворота, задающую преобразование координат из одной системы в другую при последовательном повороте вокруг каждой из осей.

параметры ориентирования модели центральной проекции

:= [параметры, задающие связь координат объекта на изображении и в трехмерном пространстве при условии, что ход лучей и, соответственно, формирование изображения в *оптической системе компьютерного зрения* описывается центральной проекцией]

⇒ *включает**:

{• *параметры внутреннего ориентирования*

⇒ *включает**:

{• *фокусное расстояние*
• *координаты главной точки снимка (x_0, y_0)*
}
• *параметры внешнего ориентирования*
⇒ *включает**:
{• *координаты точки положения камеры (x_C, y_C, z_C)*
• *углы, определяющие положение главного луча камеры* (ω, φ)
• *угол поворота снимка* β
}
}

калибровка камеры
:= [задача получения внутренних и внешних параметров камеры по имеющимся фотографиям или видео, отснятыми ею]

Калибровка камеры часто используется на начальном этапе решения многих задач *компьютерного зрения* и, в особенности, задач, связанных с дополненной реальностью. Кроме того, калибровка камеры может использоваться для исправления искажений, вызванных *оптической системой компьютерного зрения*, например, дисторсии.

оптическая аберрация
:= [отклонение от идеальной модели формирования изображения в оптической системе]

дисторсия
:= [аберрация оптических систем, при которой коэффициент линейного увеличения изменяется по мере удаления отображаемых предметов от оптической оси]
⇒ *включает**:
{• *радиальная дисторсия*
:= [дисторсия, вызванная сферической поверхностью линз оптической системы]
• *тангенциальная дисторсия*
:= [дисторсия, вызванная неперпендикулярностью главной оптической оси и плоскости изображения и прохождением главной оптической оси не через центр снимка]
}

Пункт 4.4.8.2. Локальные признаки изображений

Одним из первых этапов *трехмерной реконструкции* по изображениям, полученным с различных ракурсов, является нахождение всевозможных комбинаций соответствующих точек на изображениях, которые являются одной и той же точкой исходной сцены. Традиционно для решения этой задачи используются алгоритмы для работы с *локальными признаками изображения*.

локальный признак изображения
:= [некоторое подмножество точек изображения, пространственные окрестности которых определенным образом характеризуют изображение целиком или присутствующие на нем объекты]
⇒ *включает**:
{• *ключевая точка*
}

Основной сложностью при реализации и работе с алгоритмами нахождения *локальных признаков изображения* является тот факт, что универсального или однозначно точного определения того, что именно является локальным признаком, не существует. Конкретное определение *локального признака* в произвольной точке произвольного изображения зависит от вида используемого алгоритма и решаемой задачи. К основным концепциям, на основании которых принимается решение о наличии в точке *локального признака изображения*, в различных исследованиях относят модуль и направление градиента, определяемые по разностной схеме значения лапласиана или гессиана в окрестности точки, собственные значения и векторы и тому подобные. В работе *Krig S.CompuVM-2016bk* приводится таксономия атрибутов описательных характеристик *локальных признаков изображения* для их более высокоуровневого анализа. Алгоритмы детектирования *локальных признаков изображения*, как правило, являются входной точкой для последующей обработки другими методами. Методы глубокого обучения, такие как сверточные

нейронные сети, также основаны на выделении низкоуровневневых *локальных признаков изображения* с помощью операции свертки.

В задачах *трехмерной реконструкции* основной интерес представляет не само нахождения *локальных признаков изображения*, а их сопоставление между несколькими кадрами для нахождения смещения объектов. Зная параметры *оптической системы компьютерного зрения* в момент получения каждого снимка и сопоставив смещение объектов на отдельных изображениях, можно получить расстояние между объектами в глобальной системе координат. Поэтому базовой задачей является построение карты диспарантности.

диспарантность
:= [различие взаимного положения двух точек, отображаемых на двух изображениях и соответствующих одной точке пространства]

карта диспарантности
:= [набор значений, соответстветствующих пикселям исходного изображения и отображающих смещение каждого исходного пикселя относительно его положения на другом изображении. Чаще всего отображается также в виде изображения, при этом значение яркости пикселей отражает величину смещения]

Карты диспарантности могут быть получены различными методами, основанными как на статистических характеристиках отдельных областей изображений, так и на нахождении ключевых точек и сопоставлении их на основе некоторого описания окрестности (то есть значения, полученного с помощью алгоритма дескриптора). Во втором случае карта диспарантности может быть построена не для всего изображения, а только для подмножества пикселей, соответствующих ключевым точкам. Данный подход является одним из основных этапов методов *фотограмметрической реконструкции*, включающим нахождение всевозможных комбинаций соответствующих точек на изображениях проекций, которые относятся к одной и той же точке исходной сцены.

методы построения карты диспарантности
⇒ *включает**:
- *вычисление суммы абсолютных разностей для небольших окрестностей изображения*
- *вычисление суммы квадратов разностей для небольших окрестностей изображения*
- *вычисление нормализованной кросскорреляции для небольших окрестностей изображений*
- *нейросетевые алгоритмы для вычисления карты диспарантности*
- *детекторы и дескрипторы ключевых точек*
- *детекторы*
  := [алгоритмы, которые по входному изображению генерируют набор ключевых точек]
  - *эмпирические детекторы*
    := [алгоритмы поиска ключевых точек, которые основаны на некоторых интуитивных представлениях, практическом опыте и результатах тестирования]
    - *FAST (см. Rosten E..MachiLfHS-2006art)*
    - *детектор Харриса (см. Harris C..aCombiCaED-1988art)*
    - *SUSAN (см. Smith S.M..SUSAN aNAtLL-1997art)*
  - *статистические детекторы*
    := [алгоритмы поиска ключевых точек, которые основаны на сравнении статистических характеристик точки или ее окрестности с некоторыми заранее известными]
    - *SIFT (см. Lowe D.G.DistiIFfSIK-2004art)*
    - *AGAST (см. Mair E..Adapt aGCDBotAST-2010art)*
  - *детекторы на основе машинного обучения*
    := [алгоритмы поиска ключевых точек, которые основаны, как правило, на существующих эмпирических или статистических детекторах, но использующие оптимизацию некоторых варьируемых параметров детектирования методами машинного обучения]
    - *FAST-ER (см. Rosten E..Faste aBaMLA-2010art)*
    - *TILDE (см. Verdie Y..TILDE aTILD-2015art)*
- *дескрипторы*
  := [алгоритмы, которые по входному изображению и набору координат точек на них, генерируют вектор признаков, описывающий эти точки в некотором метрическом пространстве]
  - *гистограммные дескрипторы*
    := [алгоритмы, строящие вектор представления для описания ключевой точки на основании гистограммы некоторой статистики, которую можно вычислить по ее окрестности. Вектора описаний гистограммных дескрипторов обычно имеют размерность не менее 128, значения отдельных элементов являются вещественными числами, так как гистограмма, как правило, нормализуется]
    - *SIFT (см. Lowe D.G.DistiIFfSIK-2004art)*

- *SURF (см. Bay H..SpeedURF-2008art)*
- *бинарные дескрипторы*
  - := [алгоритмы, разработанные с целью ускорения сопоставления векторов описаний точек и строящие вектор представления для описания ключевой точки, состоящий из элементов булева множества]
  - *BRIEF (см. Calonder M..BRIEF-2010art)*
  - *ORB (см. Rublee E..ORB:aEAtS-2011art)*
  - *BRISK (см. Leutenegger S..BRISKBRISK-2011art)*
- *методы оптического потока*
  - := [методы отображения видимого движения объектов, поверхностей или краев сцены, получаемых в результате относительного перемещения наблюдателя и объектов сцены]
  - *фазовая корреляция*
    - := [методы оптического потока, основанные на инверсии нормализованного перекрестного спектра]
  - *блочные методы*
    - := [методы оптического потока, основанные на минимизации суммы квадратов или суммы модулей разностей]
  - *дифференциальные методы оценки оптического потока, основанные на частных производных сигнала*
  - *алгоритм Лукаса-Канаде (см. Lucas B..aIteraIRTwaAtSV-1981art)*
  - *Horn-Schunck (см. Horn B.DeterOF-1981art)*
    - := [алгоритмы, основанные на минимизации функционала, описывающего отклонение от предположения о постоянстве яркости и гладкости получаемого векторного поля]
  - *общие вариационные методы*
    - := [методы, основанные на модификации метода Horn-Schunck, но использующие другие ограничения на данные и другие ограничения на гладкость]
  - *дискретные методы оптимизации*
    - := [методы, основанные на квантовании поискового пространства и присвоении каждому пикселю изображения метки таким образом, чтобы расстояние между последовательными кадрами было минимальным]

}

Количество и распределение выделенных *локальных признаков изображения* влияет на возможность не только проведения *трехмерной реконструкции*, но и на качество результатов других задач *компьютерного зрения*. При этом как не существует точного определения, что должен представлять *локальный признак изображения*, так и не существует универсального алгоритма, позволяющего работать с любым видом изображений. В работе *Halavataya K.LocalFDIfIMaODiRTA-2020art* приведены примеры параметров изображения, на основании которых могут подбираться соответствующие параметры алгоритмов детекторов и дескрипторов ключевых точек. Данные параметры также могут быть описаны в рамках *базы знаний ostis-системы*, что позволит построить адаптивную систему подбора алгоритма выделения *локальных признаков изображения* в связанных задачах *компьютерного зрения*.

§ 4.4.9. Онтология действий, выполняемых в предметной области трехмерной реконструкции

⇒ *подраздел**:
- *Пункт 4.4.9.1. Действия по формированию промежуточного трехмерного представления*
- *Пункт 4.4.9.2. Действия по формированию итогового трехмерного представления*
- *Пункт 4.4.9.3. Действия по подбору и генерации алгоритма*

На верхнем уровне любой метод *трехмерной реконструкции* может быть представлен в виде процедурного знания, например, в виде последовательности этапов, в соответствии с которыми входные данные для *трехмерной реконструкции* преобразовываются в итоговое *трехмерное представление объекта*.

Общая черта многих методов *трехмерной реконструкции* — использование промежуточного (или конечного) представления об объектах окружающего мира в трех координатах. Другими словами, метод *трехмерной реконструкции* можно представить в виде последовательности действий по формированию набора элементов, характеризующихся координатами в общем трехмерном пространстве, которые в дальнейшем, при необходимости, могут быть достроены до поверхностей, скомбинированы с двумерными представлениями и так далее для формирования нужного выходного представления:

$$r : Im(R^3) \to O \tag{4.4..1}$$

где I — входные данные, R^3 — общее трехмерное декартово пространство, $m(R^3)$ — описание элемента в системе координат общего трехмерного пространства, O — выходное *трехмерное представление объекта*. Чаще всего в качестве элементов промежуточного представления выступают отдельные точки трехмерного пространства — в этом случае такое представление называют облаком точек.

В качестве элементов могут также выступать отрезки кривых, аналитически задаваемые поверхности, полигоны и другие виды объектов.

В качестве источников данных о трехмерных координатах для промежуточного представления могут выступать:
- Непосредственно абсолютные значения трехмерных координат точек, то есть в этом случае входные данные I представляют собой набор точек R^3. Это характерно для всех методов, которые позволяют строить карту глубины сцены, так как представление в виде карты глубины с известными координатами положения наблюдателя и фокусного расстояния карты позволяет определить трехмерные координаты любой точки на ней.
- Данные, по которым с помощью некоторой дополнительной обработки могут быть получены значения трехмерных координат отдельных точек. Такие представления, как правило, намного более распространены и просты в получении.

Следует отметить, что различные источники данных при формировании промежуточного представления могут использоваться совместно, при наличии у каждого из источников некоторой привязки к общей системе координат.

Как для получения промежуточного *трехмерного представления объекта*, так и для формирования на его основе итогового представления могут использоваться различные методы и алгоритмы, структура и взаимосвязь которых определяют итоговую последовательность действий по выполнению *трехмерной реконструкции* объекта или сцены.

Пункт 4.4.9.1. Действия по формированию промежуточного трехмерного представления

Многие методы дистанционного зондирования полагаются на наличие априорной информации о трехмерных координатах исследуемого объекта, по которой можно построить облако точек для промежуточного *трехмерного представления объекта*. К таким относятся методы, которые позволяют оценивать расстояние от измерительного прибора до объекта. Тем не менее, использование этих методов требует более сложного оборудования, применение которого может быть затруднено в некоторых сценариях.

В этой связи, в качестве отдельной категории методов исследования можно выделить группу методов формирования промежуточного представления в виде облака точек по более традиционным видам информации. К таким относятся методы генерации по одному изображению, стереопаре, набору изображений или видеоряду, в которых также может использоваться информация об *оптической системе компьютерного зрения*, которая была использована для получения изображения.

Таким образом можно выделить следующие виды входных данных:
- статическое изображение или набор статических изображений;
- видеоряд (набор статических изображений с временной меткой);
- стереопара (2 статических изображения с параметрами *оптической системы компьютерного зрения*) или набор стереопар;
- стерео-видеоряд (набор стереопар с временной меткой);
- информация об *оптической системе компьютерного зрения*.

В качестве выходного представления в данном случае выступает разреженное облако точек *трехмерного пространства*, с привязкой каждой из точек этого пространства к одной или нескольким точкам исходных входных изображений.

Формирование разреженного облака точек включает в себя следующие этапы, для каждого из которых могут быть определены указанные параметры:
- предобработка;
- нахождение *локальных признаков изображения* (обычно заключается в детектировании ключевых точек):
  - алгоритм-детектор;
- сопоставление ключевых точек:
  - алгоритм-дескриптор или алгоритм оптического потока;
  - прореживание;
- оценка положений камеры:
  - модель проекции;
  - алгоритм оценки связок;

- постобработка.

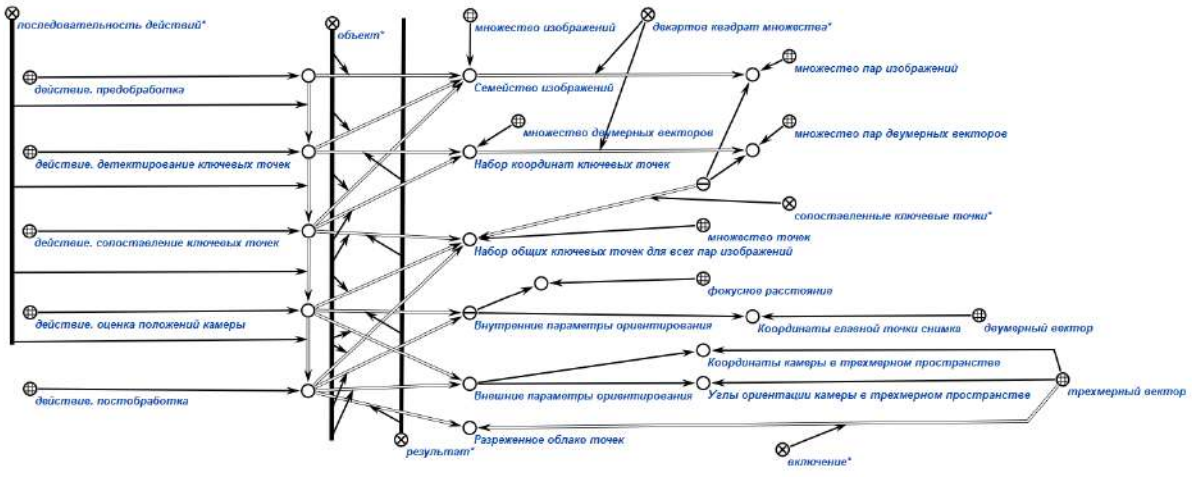
Детектирование и сопоставление ключевых точек позволяет определить точки, принадлежащие одному и тому же объекту исследуемого трехмерного пространства, при наличии достаточного количества изображений. На этапе оценки положения камеры используется математическая модель проекции, задающая взаимосвязь между двумерными координатами точки на изображении и соответствующими ей трехмерными координатами в пространстве моделирования; на этом же этапе может осуществляться эмпирическая оценка глубины, например, при помощи *нейросетевых методов*. Каждое соответствие, описанное математически с помощью модели проекции, в дальнейшем используется в алгоритме оценки связок для того, чтобы восстановить в трехмерном пространстве положения камеры для каждого из исследуемых изображений, а также определить расстояния от камер до точек, исходя из некоторого критерия минимизации ошибки обратной проекции.

Например, классический метод *трехмерной реконструкции Structure from Motion* по нескольким входным изображениям в рамках представленного конвейера можно описать следующими характеристиками:
- предобработка — преобразование к оттенкам серого;
- алгоритм-детектор — SIFT, SURF, FAST;
- алгоритм-дескриптор — SIFT, SURF, ORB;
- прореживание — RANSAC;
- модель проекции — центральная проекция;
- алгоритм оценки связок — глобальный метод связок с оптимизацией методом Левенберга-Марквардта.

В качестве еще одного примера рассмотрим метод одновременной локализации и построения карты ORB-SLAM, использующий в качестве входного представления видеоряд:
- предобработка — прореживание кадров;
- алгоритм-детектор — FAST;
- алгоритм-дескриптор — ORB + метод оптического потока Лукаса-Канаде;
- прореживание — RANSAC, прореживание по инвариантам движения;
- модель проекции — центральная проекция;
- алгоритм оценки связок — инкрементальный метод связок с фиксированным положением камеры и фиксированным положением ключевых точек между кадрами, метод построения карты по движению;
- постобработка — уточнение траектории и детектирование петель.

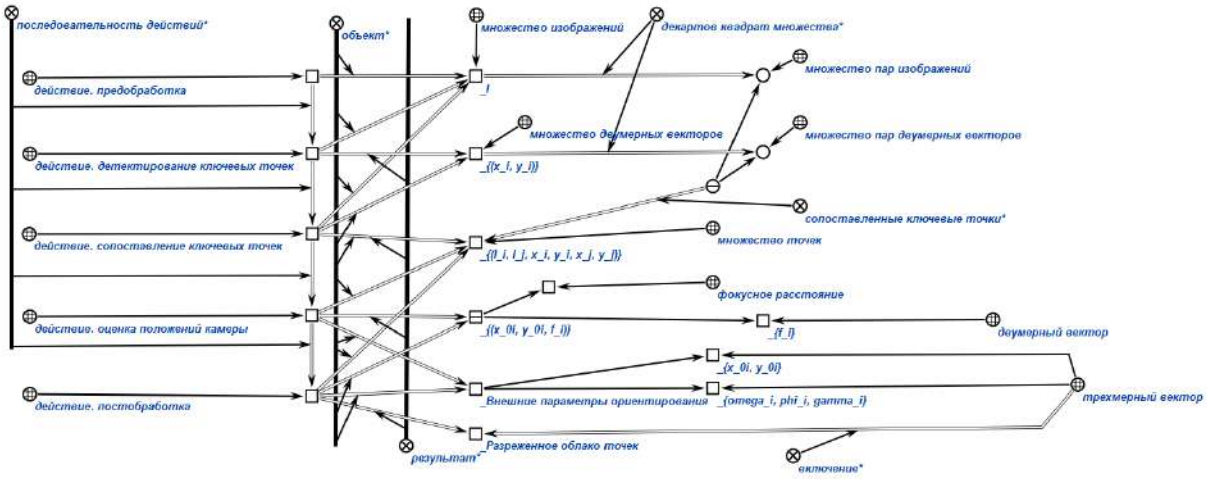
Онтологическое описание представленной последовательности действий, а также пример конкретного алгоритма, реализованного по этой последовательности действий в рамках *ostis-системы*, представлены на рисунках *SCg-текст. Онтологическое описание последовательности действий по построению промежуточного трехмерного представления в виде разреженного облака точек* и *SCg-текст. Пример алгоритма построения разреженного облака точек на основе представленного описания*.

SCg-текст. Онтологическое описание последовательности действий по построению промежуточного трехмерного представления в виде разреженного облака точек
=



SCg-текст. Пример алгоритма построения разреженного облака точек на основе представленного описания

=



Поскольку каждый из предложенных этапов описан в виде функционального отображения, предполагается добавление, удаление или модификация этапов при обработке, если соблюдается соответствие типов входных и выходных представлений в контексте решаемой задачи.

Пункт 4.4.9.2. Действия по формированию итогового трехмерного представления

В некоторых задачах разреженное облако точек в трехмерном пространстве может выступать достаточным представлением, и может считаться итогом работы алгоритма *трехмерной реконструкции*. Также достаточно популярным является представление в виде уплотненного цветного облака точек.

Тем не менее, во многих задачах такого представления недостаточно, поэтому можно выделить класс действий при формировании более сложного *трехмерного представления объекта*, в зависимости от типа требуемого выходного представления. В качестве входного представления для этого этапа выступает разреженное облако точек, а также дополнительная информация о связи конкретных точек облака с исходными представлениями.

Описание методов формирования итогового *трехмерного представления объекта* можно представить в виде совокупностей следующих этапов, с соответствующими параметрами:

- уплотнение облака точек:
  - алгоритм уплотнения;
  - связь с исходными данными;
- формирование поверхностей:
  - тип поверхности;
  - алгоритм формирования поверхностей;
- уточнение поверхностей:
  - алгоритм уточнения поверхностей;
  - связь с исходными данными;
- текстурирование поверхностей:
  - метод текстурирования;
  - связь с исходными данными;
  - разрешение конфликтов.

На этапе уплотнения облака точек информация о связи трехмерных координат точек разреженного облака с исходными данными используется для переноса дополнительных точек напрямую из исходного представления в *трехмерную модель*. Далее путем анализа полученного плотного облака точек и исходных данных формируется грубая оценка итоговой трехмерной поверхности в виде некоторого трехмерного примитива, как правило, путем формирования полигональной сетки объединением ближайших точек в треугольники. На этапе уточнения поверх-

ностей может происходить сглаживание, прореживание и объединение примитивов, полученных на предыдущем этапе, на основании некоторой информации из исходных данных. Наконец, на этапе текстурирования исходное представление переносится в построенную *трехмерную модель*, чтобы обеспечить ее реалистичность; также на этом этапе происходит разрешение конфликтов для выбора корректной стратегии текстурирования в условиях наличия нескольких равноправных источников информации о текстуре.

Например, алгоритм реконструкции поверхностей, предложенный в рамках наиболее популярной на сегодняшний день реализации метода связок можно описать в виде следующего набора параметров:
- алгоритм уплотнения — перенос окрестностей ключевых точек;
- тип поверхностей — полигональные с прямоугольными полигонами;
- алгоритм формирования поверхностей — оценка переносов камеры с помощью триангуляции Делоне, оценка планарных переносов камеры, интерполяция между положениями камеры, ручная подстройка;
- алгоритм уточнения поверхностей — отсутствует;
- метод текстурирования — прямой перенос с исходных изображений;
- метод разрешения конфликтов текстурирования — альфа-смешение пропорционально расстоянию до точки.

Пункт 4.4.9.3. Действия по подбору и генерации алгоритма

Подробное описание структуры методов не является достаточным для непосредственного выполнения *трехмерной реконструкции* по некоторому алгоритму. Поскольку для каждого из этапов итогового конвейера существует большое множество реализаций, возникает также задача оптимального выбора из набора реализаций каждого из этапов для наиболее эффективного решения поставленной задачи.

Как для выбора этапов алгоритма *трехмерной реконструкции*, так и для его непосредственной реализации может использоваться *агентно-ориентированный подход* к обработке информации. Коммуникация *агентов* осуществляется на основе обращения к общему представлению в базе знаний, по которой генерируется ряд вопросов, специфицирующих дальнейшие действия.

В качестве основных вопросов, на основании которых может осуществляться генерация алгоритма, можно выделить следующие:
- Какие входные данные будут использоваться для реконструкции?
  - Можно ли по входным данным определить расстояние до точки в трехмерном пространстве, или непосредственно ее трехмерные координаты?
  - В качестве данных будут использованы изображения?
    - Известны ли *оптические параметры системы компьютерного зрения*?
    - Известны ли положения или ориентации камеры в пространстве для каждого изображения?
    - Является ли набор изображений непрерывным видеорядом с известной частотой кадров?
  - При наличии нескольких источников входных данных, имеется ли информация о привязке описываемых ими данных к общей системе координат?
- Какой тип трехмерной модели должен быть сформирован по этим данным?
  - Требуется ли реконструкция трехмерных поверхностей?
    - Могут ли исходные данные использоваться для формирования текстур поверхностей?

На основе рассмотренного подхода может осуществляться как подбор метода, удовлетворяющего требованиям задачи, так и подбор отдельных этапов алгоритма, например, выбор наиболее оптимального алгоритма детектирования и описания ключевых точек. Кроме этого, может осуществляться комбинация *3D-представлений*, полученных разными методами (см. *Halavataya K..AdjusV3DRRUTD-2022art*).

Заключение к Главе 4.4.

В главе рассмотрено онтологическое описание *предметной области трехмерной реконструкции* и действий по их обработке в *базах знаний* на основе *Технологии OSTIS*. Описание представлено в виде доменной области самих представлений, методов их получения и соответствующих действий по их обработке.

Преимуществами использования *Технологии OSTIS* для рассмотренных задач являются:
- введение общей системы понятий и описаний методов в унифицированном и согласованном виде;
- возможность конвергенции предметной области *трехмерной реконструкции* с *предметной областью* формирования трехмерных сцен и окружений и смежными областями;
- упрощение разработки прикладных систем, использующих методы *трехмерной реконструкции*;

- возможность построения комплексной технологии проектирования с использованием интеллектуальных агентов, потребляющих предложенное описание;
- возможность создания средств интеграции отдельных компонентов, этапов различных методов и полученных внутренних представлений.

С помощью предложенного подхода становится возможным создавать *интеллектуальные кибернетические системы*, которые могут получать и оперировать *трехмерным представлением объектов* для дальнейшей обработки в прикладных задачах.

Часть 5.

Методы и средства проектирования интеллектуальных компьютерных систем нового поколения

:= [Часть 5. Компонентное проектирование интеллектуальных компьютерных систем нового поколения]
:= [Часть 5. Компонентное проектирование баз знаний, решателей задач и интерфейсов ostis-систем]
:= [Часть 5. Принципы, виды и структура библиотек многократно используемых компонентов ostis-систем]

⇒ *аннотация**:
[В данной части рассматриваются методы и средства, позволяющие повысить качество и удобство проектирования современных интеллектуальных систем. Принципы, лежащие в основе компонентного проектирования баз знаний, решателей задач и интерфейсов ostis-систем. Приведена типология многократно используемых компонентов ostis-систем, а также модель библиотек многократно используемых компонентов ostis-систем.]

⇒ *подраздел**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

Введение в Часть 5.

Основным результатом Искусственного интеллекта являются не сами интеллектуальные системы, а мощные и эффективные технологии их разработки. Анализ современных технологий проектирования интеллектуальных компьютерных систем показывает, что наряду с весьма впечатляющими достижениями имеют место следующие серьезные проблемы:
- Высокие требования к начальной квалификации пользователей и разработчиков. Технологии Искусственного интеллекта не ориентированы на широкий круг разработчиков и пользователей интеллектуальных систем и, следовательно, не получили массового распространения.
- Длительные сроки разработки интеллектуальных компьютерных систем и высокий уровень трудоемкости их сопровождения и расширения.
- Высока степень зависимости интеллектуальных компьютерных систем и их компонентов друг от друга, отсутствие их автоматической синхронизации. Отсутствие самодостаточности систем и компонентов, способности их функционировать отдельно друг от друга без утраты целесообразности их использования.
- Возрастание времени решения задачи при расширении функционала решателя задач и при расширении базы знаний системы.
- Отсутствие методик проектирования интеллектуальных компьютерных систем. Обновление компьютерных систем часто сводится к разработке различного рода "заплаток", которые устраняют не причины выявленных недостатков обновляемых компьютерных систем, а только некоторые следствия этих причин.
- Отсутствие инструментальных средств проектирования интеллектуальных компьютерных систем, в том числе интеллектуальных обучающих подсистем, подсистем компонентного проектирования компьютерных систем.

Детальнее перечисленные проблемы рассматриваются в работах *Голенков В.В..ПринцПМСТ-2011ст*, *Ивашенко В.П.СеманТКПБЗ-2011ст*, *Заливако С.С..СеманТКПИРЗ-2011ст*, *Корончик Д.Н.СеманТКПП-2011ст*, *Голенков В.В..ОткрыПНнСТ-2013ст* и *Голенков В.В..ПроекОСТКПИСЧ1-2014ст*.

Для устранения возникших недостатков предлагается использовать технологию, в основе которой лежат принципы, описанные в работах *Голенков В.В..ПринцПМСТ-2011ст*, *Заливако С.С..СеманТКПИРЗ-2011ст*, *Заливако С.С..СеманТКПИРЗ-2012ст*, *Голенков В.В..ОткрыПНнСТ-2013ст*, *Давыденко И.Т.ТехноКПБЗнО-2013ст*, *Шункевич Д.В.Модели иСКПМ-2013ст* и *Голенков В.В..ПроекОСТКПИСЧ1-2014ст*:
- Доступность. Сравнительно неподготовленный пользователь может использовать технологию Искусственного интеллекта.
- Полнота. Технология должна обеспечивать решение как можно большего набора задач.
- Модульность. Ориентация на компонентное (модульное, сборочное) проектирование интеллектуальных систем.
- Поэтапное эволюционное проектирование на основе быстрого прототипирования.

- Высокий уровень гибкости интеллектуальных систем, простота их модификации (расширяемость). Возможность свободного добавления новых компонентов самого различного вида в интеллектуальную систему. Для этого необходимо достичь максимального уровня независимости эволюции баз знаний от эволюции решателей задач и интеллектуальных интерфейсов этих систем.
- Полная совместимость инструментальных средств проектирования, а также инфраструктура проектирования с проектируемыми системами — инструментальные средства строятся как интеллектуальные системы на основе тех же принципов.
- Включение в состав интеллектуальных систем обучающей подсистемы как для разработчиков, так и для конечных пользователей.
- Включение подсистем самоанализа, самотестирования, верификации, отладки и самосовершенствования в состав разрабатываемых интеллектуальных систем.

Интеллектуальная система должна быть ориентирована не столько на разработку определенного класса систем, сколько на их постоянное совершенствование на основе указанных принципов.

В состав технологии проектирования интеллектуальных систем входят следующие составляющие (см. *Заливако С.С..СеманТКПИРЗ-2012ст*, *Давыденко И.Т.ТехноКПБЗнО-2013ст* и *Шункевич Д.В.Модели иСКПМ-2013ст*):

- Модель интеллектуальной компьютерной системы.
- Библиотека многократно используемых компонентов интеллектуальных компьютерных систем.
- Интеллектуальная интегрированная система автоматизации коллективного проектирования интеллектуальных компьютерных систем, включая подсистемы редактирования, отладки, оценки производительности и визуализации разработанных компонентов, а также подсистему имитационного моделирования.
- Методика проектирования интеллектуальных компьютерных систем.
- Интеллектуальный пользовательский интерфейс.
- Обучающие подсистемы проектирования интеллектуальных компьютерных систем, включая подсистему ведения диалога с разработчиком и пользователем.
- Подсистема тестирования и верификации интеллектуальных компьютерных систем, включая подсистему проверки совместимости разработанной системы с другими системами.

Глава 5.1.

Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

⇒ *эпиграф**:
[Все, что можно сделать одинаково, нужно делать одинаково.]

⇒ *автор**:
- *Орлов М. К.*
- *Шункевич Д. В.*
- *Ковалев М. В.*
- *Садовский М. Е.*
- *Загорский А. Г.*
- *Банцевич К. А.*

⇒ *аннотация**:
[Важнейшим этапом эволюции любой технологии является переход к компонентному проектированию на основе постоянно пополняемой библиотеки многократно используемых компонентов. Идея библиотеки компонентов не нова, но семантическая мощность ***Библиотеки Экосистемы OSTIS*** значительно выше аналогов за счет того, что подавляющее большинство компонентов библиотеки — компоненты *базы знаний*, представленные на унифицированном языке смыслового представления знаний (*SC-коде*). Таким образом, в Библиотеке Экосистемы OSTIS обеспечивается высокий уровень семантической совместимости компонентов, что приводит к высокому уровню семантической совместимости *ostis-систем*, использующих комплексную библиотеку многократно используемых семантически совместимых компонентов ostis-систем.]

⇒ *подраздел**:
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*
- *§ 5.1.4. Менеджер многократно используемых компонентов ostis-систем*

⇒ *ключевой знак**:
- *Библиотека Метасистемы OSTIS*
- *Библиотека Экосистемы OSTIS*
- *Реализация менеджера многократно используемых компонентов ostis-систем*

⇒ *ключевое понятие**:
- *библиотека многократно используемых компонентов ostis-систем*
- *многократно используемый компонент ostis-систем*
- *отношение, специфицирующее многократно используемый компонент ostis-систем*
- *менеджер многократно используемых компонентов ostis-систем*

⇒ *ключевое знание**:
- *Классификация многократно используемых компонентов ostis-систем*
- *спецификация многократно используемых компонентов ostis-систем*

⇒ *библиографическая ссылка**:
- *Житко В.А.ТехноКПСНиПвСС-2011ст*
- *Заливако С.С..СеманТКПИРЗ-2012ст*
- *Борисов А.Н.ПострИСОнЗсП-2014ст*
- *Голенков В.В..СеманТКПС-2015ст*
- *Житко В.А..СеманТКПЕЯИИВОС-2011ст*
- *Голенков В.В..ОткрыПНнСТ-2013ст*
- *Шункевич Д.В..СредстваПКПС-2015ст*
- *Заливако С.С..СеманТКПИРЗ-2011ст*
- *Голенков В.В..ПроекОСТКПИСЧ1-2014ст*

- *Iyengar A.CompoDfRD-2021art*
- *Ford B..CompoD-2019art*
- *Wilkes M.V.Prepa oPfaEDC-1951bk*
- *Уилкс М..СоставПдЭСМ-1953кн*
- *Волченскова Н.И.ТехноМРиЖБ-1984ст*
- *Blahser J..ThineAfaFTDP-2021art*
- *Hepp M.OntolSotABPaGC-2008art*
- *Memduhoglu M..PossiCoSSMaTtMRSDP-2018art*
- *Fritzson P.ModelLO-2014art*
- *Prakash S.P.Worki wMPA-2022art*
- *Москаленко Ф.М..ТехноРРЗИСдОПИнОРРОИ-2016ст*
- *Пивоварчик О.В.КомпоАИСК-2015ст*
- *Корончик Д.Н.СеманТКПП-2011ст*
- *Давыденко И.Т.ТехноКПБЗнО-2013ст*
- *Ивашенко В.П.УнифиПиИЗ-2013ст*
- *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*
- *Orlov M.K.ComprLoRSC-2022art*
- *Голенков В.В..ПринцПМСТ-2011ст*
- *Ивашенко В.П.СеманТКПБЗ-2011ст*
- *Лазуркин Д.А.ТехноКППОнОСС-2011ст*
- *Давыденко И.Т..ИнтелССГ-2011ст*
- *Корончик Д.Н.СеманММПИ*
- *Шункевич Д.В.Модели иСКПМ-2013ст*
- *Корончик Д.Н.УнифиСМПИ-2013ст*
- *Елисеева О.Е..КомпоПИОС-2013ст*
- *Шункевич Д.В..МетодКПСУЗ-2013ст*
- *Давыденко И.Т.МоделМиСРГБ-2017дс*
- *Davydenko I.T.SemanMMaToKB-2018art*
- *Samson W.T..OntolBCoPS-1995art*
- *Studer.R..OntologCPSM-1996art*
- *Benjamins V.R..KnowlSTOaPS-1999art*

Введение в Главу 5.1.

⇒ *ключевое понятие**:
- *компонентное проектирование интеллектуальных компьютерных систем*

⇒ *ключевое знание**:
- *Проблемы в реализации компонентного проектирования интеллектуальных систем*
- *Требования к реализации компонентного проектирования интеллектуальных систем*

Повторное использование готовых компонентов широко применяется во многих отраслях, связанных с разработкой различного рода компьютерных систем, поскольку позволяет уменьшить трудоемкость их создания и стоимость (путем минимизации количества труда за счет отсутствия необходимости разрабатывать какой-либо компонент), повысить качество создаваемых компьютерных систем и снизить профессиональные требования к разработчикам этих систем (см. *Житко В.А.ТехноКПСНиПвСС-2011ст*). Таким образом, осуществляется переход от программирования компонентов или целых систем к их проектированию (дизайну, сборке) на основе готовых компонентов (см. *Заливако С.С..СеманТКПИРЗ-2012ст*). ***компонентное проектирование интеллектуальных компьютерных систем*** предполагает подбор существующих компонентов, способных решить поставленную задачу целиком или декомпозицию задачи на подзадачи с выделением компонентов для каждой из них (см. *Борисов А.Н.ПострИСОнЗсП-2014ст*). Проектируемые системы по предлагаемой технологии обладают высоким уровнем гибкости, их разработка осуществляется поэтапно, переходя от одной целостной версии системы к другой. При этом стартовая версия системы может быть ядром соответствующего класса систем, входящим в ***библиотеку многократно используемых компонентов***. Технология компонентного проектирования интеллектуальных компьютерных систем включает в себя комплекс согласованных частных технологий, обеспечивающих целостное проектирование компьютерных систем. Она включает технологию компонентного проектирования баз знаний, решателей задач, интерфейсов и другие (см. *Голенков В.В..СеманТКПС-2015ст*).

Главным элементом семантической технологии *компонентного проектирования интеллектуальных компьютерных систем* является *библиотека совместимых многократно используемых компонентов*. Это позволяет проектировать интеллектуальные системы комбинируя уже существующие компоненты, выбирая нужные из соответствующих библиотек (см. *Житко В.А..СеманТКПЕЯИИВОС-2011ст*). Использование готовых компонентов предполагает, что распространяемый компонент верифицирован и документирован, а возможные ошибки и ограничения устранены либо специфицированы и известны. Создание *библиотеки многократно используемых компонентов* не означает пересоздание всех уже существующих современных продуктов информационных технологий. Технология компонентного проектирования интеллектуальных компьютерных систем предполагает использование огромного опыта в разработке современных компьютерных систем, однако обязательным является спецификация каждого компонента (как вновь созданного, так и интегрируемого существующего) для обеспечения возможности его установки и совместимости с другими компонентами и системами. Тем не менее эффективная технология компонентного проектирования появится только тогда, когда сформируется "критическая масса" разработчиков прикладных систем, участвующих в пополнении *библиотек многократно используемых компонентов* проектируемых систем (см. *Голенков В.В..ОткрыПНнСТ-2013ст*).

Проблемы реализации компонентного подхода к проектированию интеллектуальных компьютерных систем наследуют проблемы современных *технологий проектирования интеллектуальных систем*, а также включают следующие (см. *Шункевич Д.В..СредстваПКПС-2015ст*):

компонентное проектирование интеллектуальных систем
⇒ *проблемы текущего состояния**:
Проблемы в реализации компонентного проектирования интеллектуальных систем
= {• [Несовместимость компонентов, разработанных в рамках разных проектов, вследствие отсутствия унификации в принципах представления различных видов знаний в рамках одной *базы знаний*, и, как следствие, отсутствие унификации в принципах выделения и ***спецификации многократно используемых компонентов***.]
• [Невозможность автоматической интеграции компонентов в систему без ручного вмешательства пользователя.]
• [Автоматическое обновление компонентов приводит к рассогласованности как отдельных модулей компьютерных систем, так и самих систем между собой.]
• [Отсутствие классификации компонентов на различных уровнях детализации.]
• [Не проводится тестирование, верификация и анализ качества компонентов, не выделяются преимущества, недостатки, ограничения компонентов.]
• [Многие компоненты используют для идентификации язык разработчика (как правило, английский), и предполагается, что все пользователи будут использовать этот же язык. Однако для многих приложений это недопустимо — понятные только разработчику идентификаторы должны быть скрыты от конечных пользователей, которые должны быть в состоянии выбрать язык для идентификаторов, которые они видят.]
• [Отсутствие средств поиска компонентов, удовлетворяющих заданным критериям.]
}

компонентное проектирование интеллектуальных компьютерных систем возможно только в том случае, если отбор компонентов будет осуществляться на основе тщательного анализа качества этих компонентов. Одним из важнейших критериев такого анализа является уровень семантической совместимости анализируемых компонентов со всеми компонентами, имеющимися в текущей версии библиотеки.

Для того, чтобы решить возникшие проблемы при проектировании интеллектуальных систем и библиотек их многократно используемых компонентов, необходимо придерживаться общих принципов *технологии проектирования интеллектуальных компьютерных систем*, а также выполнить следующие требования:

компонентное проектирование интеллектуальных систем
⇒ *предъявляемые требования**:
Требования к реализации компонентного проектирования интеллектуальных систем
= {• [Обеспечение совместимости (интегрируемости) компонентов интеллектуальных компьютерных систем на основе унификации представления этих компонентов.]
• [Четкое разделение процесса разработки формальных описаний интеллектуальных компьютерных систем и процесса их реализации по этому описанию.]
• [Четкое разделение разработки формального описания проектируемой интеллектуальной системы от разработки различных вариантов интерпретации таких формальных описаний систем.]
• [Наличие онтологии компонентного проектирования интеллектуальных компьютерных систем, включающей (1) описание методов компонентного проектирования, (2) модель *библиотеки многократно используемых компонентов*, (3) модель *спецификации многократно используемых компонентов*, (4) полную *классификацию многократно используемых компонентов*, (5) описание средств взаимо-

действия разрабатываемой интеллектуальной компьютерной системы с *библиотеками многократно используемых компонентов*.]

- [Наличие *библиотек многократно используемых компонентов интеллектуальных компьютерных систем*, включающих спецификации компонентов.]
- [Наличие средств взаимодействия разрабатываемой интеллектуальной компьютерной системы с библиотеками многократно используемых компонентов для установки любых видов компонентов и управления ими в создаваемой системе.]
  - ⇒ *примечание**:
    [Под установкой компонента понимается его транспортировка в систему (копирование sc-элементов и/или скачивание файлов компонента), а также выполнение вспомогательных действий для того, чтобы компонент мог функционировать в создаваемой системе.]

}

Более подробно указанные требования рассмотрены в работах *Заливако С.С..СеманТКПИРЗ-2011ст*, *Голенков В.В..ОткрыПНнСТ-2013ст* и *Голенков В.В..ПроекОСТКПИСЧ1-2014ст*.

Большинство существующих систем создано как автономные программные продукты, которые не могут быть использованы в качестве компонентов других систем. Необходимо использовать либо целую систему, либо ничего. Небольшое число систем поддерживает компонентно-ориентированную архитектуру способную интегрироваться с другими системами (рассматривается в работах *Iyengar A.CompoDfRD-2021art*, *Ford B..CompoD-2019art*). Однако, их интеграция возможна при условии использования одинаковых технологий и только при проектировании одной командой разработчиков.

Многократная повторная разработка уже имеющихся технических решений обусловлена либо тем, что известные технические решения плохо интегрируются в разрабатываемую систему, либо тем, что эти технические решения трудно найти. Данная проблема актуальна как в целом в сфере разработки компьютерных систем, так и в сфере разработки систем, основанных на знаниях, поскольку в системах такого рода степень согласованности различных видов знаний влияет на возможность системы решать нетривиальные задачи.

§ 5.1.1. Анализ современных библиотек многократно используемых компонентов

⇒ *ключевой знак**:
- *pip*
- *poetry*
- *Библиотека STL*
- *WebProtege*
- *Modelica*
- *Microsoft Power Apps*
- *Платформа IACPaaS*

⇒ *ключевое понятие**:
- *пакетный менеджер*

⇒ *ключевое знание**:
- *Файл конфигурации pip*
- *Файл конфигурации poetry*
- *Структура библиотеки STL*

На данный момент не существует комплексной библиотеки многократно используемых семантически совместимых компонентов компьютерных систем в целом, не говоря об интеллектуальных. Существуют некоторые попытки создания библиотек типовых методов и программ для традиционных компьютерных систем, однако такие библиотеки не решают вышеперечисленные проблемы.

Термин "библиотека подпрограмм", одними из первых упомянули Уилкс М., Уиллер Д. и Гилл С. в качестве одной из форм организации вычислений на компьютере (рассматривается в работах *Wilkes M.V.Prepa oPfaEDC-1951bk*, *Уилкс М..СостаПдЭСМ-1953кн*). Исходя из изложенного в их книге, под библиотекой понимался набор "коротких, заранее заготовленных программ для отдельных, часто встречающихся (стандартных) вычислительных операций" (см. *Волченскова Н.И.ТехноМРиЖБ-1984ст*). Стоит отметить, что компонентами библиотек являются не только программы, но и компоненты интерфейсов и баз знаний.

К традиционным решениям относятся ***пакетные менеджеры*** языков программирования и операционных систем, а также отдельные системы и платформы с встроенными компонентами и средствами для сохранения создаваемых компонентов.

Компоненты библиотеки могут быть реализованы на разных *языках программирования* (что приводит к тому, что для каждого *языка программирования* разрабатываются свои библиотеки со своими решениями различных часто встречаемых ситуаций), а также могут располагаться в разных местах, что приводит к тому, что в библиотеке необходимо средство для поиска компонентов и их установки.

Современные *пакетные менеджеры*, такие как npm, pip, poetry, maven, apt, pacman и другие имеют преимущество в том, что они способны разрешать конфликты при установке зависимых компонентов, однако они не учитывают семантику компонентов, а только лишь устанавливают компоненты по идентификатору (см. *Blahser J..ThineAfaFTDP-2021art*). Библиотеки таких компонентов являются только лишь хранилищем компонентов, никак не учитывающим назначение компонентов, их преимущества и недостатки, сферы применения, иерархию компонентов и другую информацию, необходимую для интеллектуализации компонентного проектирования компьютерных систем. Поиск компонентов в *библиотеках компонентов*, соответствующих данных пакетным менеджерам сводится к поиску по идентификатору компонента. Современные *пакетные менеджеры* являются лишь "установщиками" без автоматической интеграции компонентов в систему. Также существенным недостатком современного подхода является платформенная зависимость компонентов. Современные библиотеки компонентов ориентированы только на какой-то определенный язык программирования, операционную систему или платформу.

Пакетный менеджер ***pip*** является системой управления пакетами, которая используется для установки пакетов из Python Package Index, который является некоторой библиотекой таких пакетов. Зачастую *pip* устанавливается вместе с *Python*. Пакетный менеджер *pip* используется только для языка программирования Python. Он имеет множество функций для работы с пакетами:

- установка пакета;
- установка пакета специализированной версии;
- удаление пакета;
- переустановка пакета;
- отображение установленных пакетов;
- поиск пакетов;
- верификация зависимостей пакетов;
- создание файла конфигурации со списком установленных пакетов и их версий;
- установка множества пакетов из файла конфигурации.

Пример файла конфигурации пакетов pip приведён на Рисунок. Файл конфигурации pip.

Рисунок. Файл конфигурации pip

=

requirements.txt

```
py==1.8.1
pip==19.0.3
Mako==1.1.1
MarkupSafe==1.1.1
six==1.14.0
attrs==19.3.0
pytest==5.3.5
pluggy==0.13.1
setuptools==40.8.0
parse==1.14.0
glob2==0.7
```

Пакетный менеджер *pip* хорошо работает с зависимостями, отображает безуспешно установленные пакеты, а также отображает информацию о требуемой версии пакета при конфликте с другим пакетом.

Альтернативой пакетного менеджера pip является пакетный менеджер ***poetry***, который также ориентирован на язык программирования *Python*. Преимущество *poetry* перед *pip* в том, что он автоматически работает с виртуальными окружениями, способен самостоятельно их находить и создавать. Файл конфигурации для пакетов *poetry* является более богатым, чем у pip, он хранит такие сведения, как имя проекта, версия проекта, его описание, лицензия, список авторов, URL проекта, его документации и сайта, список ключевых слов проекта и список *PyPI* классификаторов. *poetry* позволяет более гибко настраивать пакеты для проектов *Python*, файл конфигурации *poetry* представляет собой более богатую спецификацию проекта (см. рисунок Рисунок. Файл конфигурации poetry), од-

нако эта спецификация не позволяет достичь совместимости между компонентами даже в рамках *Python* проектов и предназначена преимущественно только для чтения разработчиком. Автоматизировать проектирование компьютерных систем с помощью пакетного менеджера *poetry* или *pip* невозможно, так как требуется вмешательство разработчика, который должен вручную совместить интерфейсы устанавливаемых пакетов. Другие пакетные менеджеры языков программирования и операционных систем устроены по такому же принципу: существует хранилище компонентов (библиотека), которая представляет собой множество пакетов этого языка программирования или операционной системы и с которым взаимодействует менеджер компонентов.

Рисунок. Файл конфигурации poetry
=

```
[tool.poetry]
name = "first"
version = "0.1.0"
description = ""
authors = ["…"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.10"

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

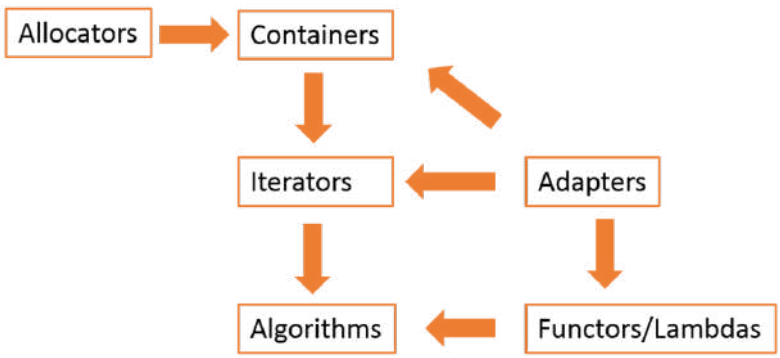
В качестве компонентного подхода к проектированию программ можно рассмотреть библиотеки подпрограмм современных *языков программирования*, например, ***Библиотеку STL*** (библиотеку стандартных шаблонов C++).

Библиотека STL — набор согласованных обобщенных алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.

В *Библиотеке STL* выделяют пять основных компонентов:
- контейнер — хранение набора объектов в памяти;
- итератор — обеспечение средств доступа к содержимому контейнера;
- алгоритм — определение вычислительной процедуры;
- адаптер — адаптация компонентов для обеспечения различного интерфейса;
- функциональный объект — сокрытие функции в объекте для использования другими компонентами.

Структура *Библиотеки STL* приведена на Рисунок. Структура библиотеки STL.

Рисунок. Структура библиотеки STL
=



Разделение позволяет уменьшить количество компонентов. Например, вместо написания отдельной функции поиска элемента для каждого типа контейнера обеспечивается единственная версия, которая работает с каждым из них, пока соблюдаются основные требования.

Совместимость компонентов (контейнеров) в *Библиотеке STL* обеспечивается общим интерфейсом использования этих компонентов.

Компонентный подход к проектированию компьютерных систем может реализовываться в рамках различных языков, платформ и приложений. Рассмотрим некоторые из них.

Онтология, реализованная на языке *OWL* (Web Ontology Language), представляет собой множество декларативных утверждений о сущностях словаря предметной области (подробнее рассматривается в работе *Hepp M.OntolSotABPaGC-2008art*). *OWL* предполагает концепцию "открытого мира", в соответствии с которой применимость описаний предметной области, помещенных в конкретном физическом документе, не ограничивается лишь рамками этого документа — содержание онтологии может быть использовано и дополнено другими документами, добавляющими новые факты о тех же сущностях или описывающими другую предметную область в терминах данной. "Открытость мира" достигается путем добавления URI каждому элементу онтологии, что позволяет воспринимать описанную на *OWL* онтологию как часть всеобщего объединенного знания.

WebProtege представляет собой многопользовательский веб-интерфейс, позволяющий редактировать и хранить онтологии в формате *OWL* в совместной среде (см. *Memduhoglu M..PossiCoSSMaTtMRSDP-2018art*). Данный проект позволяет не только создавать новые онтологии, но также загружать уже существующие онтологии, которые хранятся на сервере университета Стэнфорда. К преимуществу данного проекта можно отнести автоматическую проверку ошибок в процессе создания объектов онтологий. Данный проект является примером попытки решения проблемы накопления, систематизации и повторного использования уже существующих решений, однако, недостатком данного решения является обособленность разрабатываемых онтологий. Каждый разработанный компонент имеет свою иерархию понятий, подход к выделению классов и сущностей, которые зависят от разработчиков данных онтологий, так как в рамках данного подхода не существует универсальной модели представления знаний, а также формальной спецификации компонентов, представленных в виде онтологий. Следовательно, возникает проблема их семантической несовместимости, что, в свою очередь, приводит к невозможности повторного использования разработанных онтологий при проектировании баз знаний. Данный факт подтверждается наличием на сервере университета Стэнфорда многообразия различных онтологий на одни и те же темы.

На основе языка ***Modelica*** разработано большое число свободно доступных библиотек компонентов, одной из которых является библиотека Modelica_StateGraph2, включающая компоненты для моделирования дискретных событий, реактивных и гибридных систем с помощью иерархических диаграмм состояния (см. *Fritzson P.ModelLO-2014art*). Основным недостатком систем на базе языка *Modelica* является отсутствие совместимости компонентов и достаточной документации, а также узкая направленность разрабатываемых компонентов.

Microsoft Power Apps — это набор приложений, служб и соединителей, а также платформа данных, которая предоставляет среду разработки для эффективного создания пользовательских приложений для бизнеса. Платформа *Microsoft Power Apps* имеет средства для создания библиотеки многократно используемых компонентов графического интерфейса, а также предварительно созданные модели распознавания текста (чтение визитных карточек или чеков) и средство обнаружение объектов, которые можно подключить к разрабатываемому приложению (см. *Prakash S.P.Worki wMPA-2022art*). Библиотека компонентов *Microsoft Power Apps* представляет собой множество создаваемых пользователем компонентов, которые можно использовать в любых приложениях. Преимущество библиотеки в том, что компоненты могут настраивать свойства по умолчанию, которые можно гибко редактировать в любых приложениях, использующих компоненты. Недостаток в том, что отсутствует семантическая совместимость компонентов, спецификация компонентов, не решена проблема существования семантически эквивалентных компонентов, нету иерархии компонентов и средств поиска этих компонентов. Компоненты платформы *Microsoft Power Apps* являются многократно используемыми только для однотипных приложений, которые создаются одним и тем же разработчиком.

Платформа IACPaaS (Intelligent Applications, Control and Platform as a Service) — облачная платформа для разработки, управления и удаленного использования интеллектуальных облачных сервисов (см. *Москаленко Ф.М..ТехноРРЗИСдОПИнОРРОИ-2016ст*). Она предназначена для обеспечения поддержки разработки, управления и удаленного использования прикладных и инструментальных мультиагентных облачных сервисов (прежде всего интеллектуальных) и их компонентов для различных предметных областей. Платформа предоставляет доступ:

- прикладным пользователям (специалистам в различных предметных областях) — к прикладным сервисам;
- разработчикам прикладных и инструментальных сервисов и их компонентов — к инструментальным сервисам;
- управляющим интеллектуальными сервисами;
- к сервисам управления.

Платформа IACPaaS поддерживает:

- базовую технологию разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов с использованием базовых инструментальных сервисов платформы, поддерживающих эту технологию;
- множество специализированных технологий разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов, с использованием специализированных инструментальных сервисов платформы, поддерживающих эти технологии.

Платформа IACPaaS также не имеет средств для унифицированного представления компонентов интеллектуальных компьютерных систем и средств для их спецификации и автоматической интеграции компонентов.

Исходя из проведенного анализа можно сказать, что на текущем состоянии развития информационных технологий не существует комплексной библиотеки многократно используемых семантически совместимых компонентов компьютерных систем. Таким образом, предлагается комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем.

§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем

⇒ *ключевой знак**:
- *Библиотека Метасистемы OSTIS*
- *Пример интеграции многократно используемого компонента ostis-систем*

⇒ *ключевое понятие**:
- *библиотека многократно используемых компонентов ostis-систем*
- *материнская ostis-система*
- *дочерняя ostis-система*

⇒ *ключевое знание**:
- *Архитектура Экосистемы OSTIS с точки зрения библиотек многократно используемых компонентов*
- *Функциональные возможности библиотеки многократно используемых компонентов ostis-систем*
- *Декомпозиция библиотеки многократно используемых компонентов ostis-систем*

Основным требованием, предъявляемым к *Технологии OSTIS*, является обеспечение возможности совместного использования в рамках *ostis-систем* различных *видов знаний* и различных *моделей решения задач* с возможностью неограниченного расширения перечня используемых в *ostis-системе видов знаний* и *моделей решения задач* без существенных трудозатрат, а также различных видов интерфейсов. Следствием данного требования является необходимость реализации компонентного подхода на всех уровнях, от простых компонентов баз знаний, решателей задач и интерфейсов до целых *встраиваемых ostis-систем* (понятие компонента рассматривается далее в *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*). Интеллектуальная система, спроектированная по *Технологии OSTIS*, представляет собой интеграцию *многократно используемых компонентов баз знаний*, *многократно используемых компонентов решателей задач* и *многократно используемых компонентов интерфейсов* (см. *Пивоварчик О.В.КомпоАИСК-2015ст*). Различные *многократно используемые компоненты ostis-систем* объединяются в *библиотеки многократно используемых компонентов ostis-систем*. Разработчики любой ostis-системы могут включить в ее состав библиотеку, которая позволит им накапливать и распространять результаты своей деятельности среди других участников *Экосистемы OSTIS* в виде *многократно используемых компонентов*. Решение о включении компонента в библиотеку принимается экспертным сообществом разработчиков, ответственным за качество этой библиотеки. Верификацию компонентов можно автоматизировать путем проверки наличия обязательной части их спецификации, а также тестированием корректности автоматической установки и интеграции компонентов.

В рамках *Экосистемы OSTIS* существует множество библиотек многократно используемых компонентов ostis-систем, являющихся подсистемами соответствующих *материнских ostis-систем*. Все библиотеки в рамках *Экосистемы OSTIS* объединяются в *Библиотеку Экосистемы OSTIS*. Материнская ostis-система отвечает за какой-то класс компонентов и является САПРом для этого класса, содержит методики разработки таких компонентов, их классификацию, подробные пояснения ко всем подклассам компонентов. Таким образом, формируется иерархия ***материнских ostis-систем***. *материнская ostis-система* в свою очередь может являться дочерней *ostis-системой* для какой-либо другой *ostis-системы*, заимствуя компоненты из библиотеки, входящей в состав этой другой *ostis-системы*.

ostis-система
⊃ *материнская ostis-система*
≔ [ostis-система, имеющая в своем составе библиотеку многократно используемых компонентов.]

∋ *Метасистема OSTIS*

⊃ *дочерняя ostis-система*

:= [ostis-система, в составе которой имеется компонент, заимствованный из какой-либо библиотеки многократно используемых компонентов.]

Библиотека многократно используемых компонентов ostis-систем позволяет использовать проектный опыт по разработке и модернизации ostis-систем различного назначения.

библиотека многократно используемых компонентов ostis-систем

⇒ *часто используемый sc-идентификатор**:
[библиотека компонентов ostis-систем]

⇒ *часто используемый sc-идентификатор**:
[библиотека компонентов]

:= [библиотека совместимых многократно используемых компонентов]

:= [комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем]

:= [библиотека многократно используемых и совместимых компонентов интеллектуальных компьютерных систем нового поколения]

:= [библиотека многократно используемых и совместимых компонентов интеллектуальных компьютерных систем нового поколения]

:= [библиотека типовых компонентов ostis-систем]

:= [библиотека многократно используемых компонентов OSTIS]

:= [библиотека повторно используемых компонентов OSTIS]

:= [библиотека intelligent property компонентов ostis-систем]

⇒ *сокращение**:
[библиотека ip-компонентов ostis-систем]

∋ *типичный пример'*:
Библиотека Метасистемы OSTIS

:= [Распределенная библиотека типовых (многократно используемых) компонентов ostis-систем в составе Метасистемы OSTIS]

:= [Библиотека многократно используемых компонентов ostis-систем в составе *Метасистемы OSTIS*]

∋ *типичный пример'*:
Библиотека Экосистемы OSTIS

⇒ *часто используемый sc-идентификатор**:
[Библиотека OSTIS]

:= [Библиотека многократно используемых и совместимых компонентов интеллектуальных компьютерных систем нового поколения]

:= [Библиотека типовых компонентов интеллектуальных компьютерных систем нового поколения]

:= [Распределенная библиотека типовых (многократно используемых) компонентов ostis-систем в составе Экосистемы OSTIS]

:= [Библиотека многократно используемых компонентов ostis-систем в составе *Экосистемы OSTIS*]

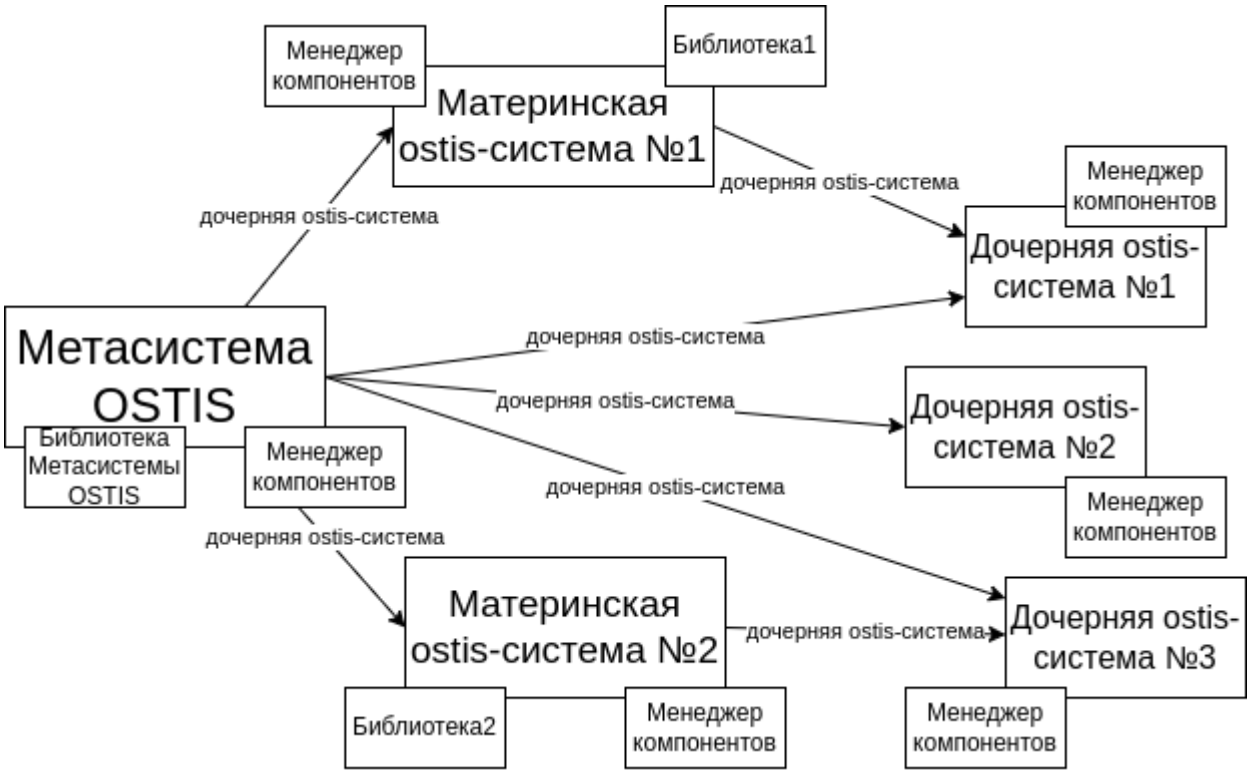
⇐ *объединение**:

{• *библиотека многократно используемых компонентов баз знаний ostis-систем (см. § 5.2.6. Многократно используемые компоненты баз знаний ostis-систем)*
• *библиотека многократно используемых компонентов решателей задач ostis-систем (см. § 5.3.3. Многократно используемые компоненты решателей задач ostis-систем)*
• *библиотека многократно используемых компонентов интерфейсов ostis-систем (см. § 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем)*
• *библиотека встраиваемых ostis-систем*
• *библиотека ostis-платформ*
}

Главной *библиотекой многократно используемых компонентов ostis-систем* является *Библиотека Метасистемы OSTIS. Метасистема OSTIS* (см. *Главу 7.2. Метасистема OSTIS*) выступает ***материнской системой*** для всех разрабатываемых ostis-систем, поскольку содержит все базовые компоненты (см. *Рисунок. Архитектура Экосистемы OSTIS с точки зрения библиотек многократно используемых компонентов*).

Рисунок. Архитектура Экосистемы OSTIS с точки зрения библиотек многократно используемых компонентов

=



Основу для реализации компонентного подхода в рамках *Технологии OSTIS* составляет ***Библиотека Метасистемы OSTIS***. *Метасистема OSTIS* ориентирована на разработку и практическое внедрение методов и средств компонентного проектирования семантически совместимых интеллектуальных систем, которая предоставляет возможность быстрого создания интеллектуальных систем различного назначения. В состав *Метасистемы OSTIS* входит ***Библиотека Метасистемы OSTIS***. Сферы практического применения технологии компонентного проектирования семантически совместимых интеллектуальных систем ничем не ограничены.

Основное назначение *Библиотеки Экосистемы OSTIS* — создание условий для эффективного, осмысленного и массового проектирования ostis-систем и их компонентов путем создания среды для накопления и совместного использования компонентов ostis-систем. Такие условия осуществляются путем неограниченного расширения постоянно эволюционируемых семантически совместимых ostis-систем и их компонентов, входящих в *Экосистему OSTIS*.

Постоянно расширяемая *Библиотека Экосистемы OSTIS* существенно сокращает сроки разработки новых интеллектуальных компьютерных систем. *библиотека многократно используемых компонентов ostis-систем* позволяет избавиться от дублирования семантически эквивалентных информационных компонентов. А также от многообразия форм технической реализации используемых моделей решения задач.

Функциональные возможности библиотеки многократно используемых компонентов ostis-систем (см. *Корончик Д.Н.СеманТКПП-2011ст*):

библиотека многократно используемых компонентов ostis-систем

⇒ *функциональные возможности**:

{• **[**Хранение многократно используемых компонентов ostis-систем и их спецификаций. При этом часть компонентов, специфицированных в рамках библиотеки, могут физически храниться в другом месте ввиду особенностей их технической реализации (например, исходные тексты *ostis-платформы* могут физически храниться в каком-либо отдельном репозитории, но специфицированы как компонент будут в соответствующей библиотеке). В этом случае спецификация компонента в рамках библиотеки должна также включать описание (1) того где располагается компонент и (2) сценария его автоматической установки в дочернюю ostis-систему.**]**

• **[**Просмотр имеющихся компонентов и их спецификаций, а также поиск компонентов по фрагментам их спецификации.**]**

- [Хранение сведений о статистике использования компонентов. Например, в каких дочерних ostis-системах какие из компонентов библиотеки и какой версии используются (были скачаны). Это необходимо для учета востребованности того или иного компонента, оценки его важности и популярности.]
- [Систематизация многократно используемых компонентов ostis-систем.]
- [Обеспечение версионирования многократно используемых компонентов ostis-систем.]
- [Поиск зависимостей между многократно используемыми компонентами в рамках библиотеки компонентов.]
- [Обеспечение автоматического обновления компонентов, заимствованных в дочерние ostis-системы. Данная функция может включаться и отключаться по желанию разработчиков дочерней ostis-системы. Одновременное обновление одних и тех же компонентов во всех системах, его использующих, не должно ни в каком контексте приводить к несогласованности между этими системами. Это требование может оказаться довольно сложным, но без него работа *Экосистемы OSTIS* невозможна.]

}

библиотека многократно используемых компонентов ostis-систем является подсистемой ostis-систем, которая имеет свою базу знаний, свой решатель задач и свой интерфейс. Однако не каждая ostis-система обязана иметь библиотеку компонентов.

библиотека многократно используемых компонентов ostis-систем
⇒ *обобщенная декомпозиция**:
{• *база знаний библиотеки многократно используемых компонентов ostis-систем*
• *решатель задач библиотеки многократно используемых компонентов ostis-систем*
• *интерфейс библиотеки многократно используемых компонентов ostis-систем*
⇒ *обобщенная декомпозиция**:
{• *минимальный интерфейс библиотеки многократно используемых компонентов ostis-систем*
• *расширенный интерфейс библиотеки многократно используемых компонентов ostis-систем*
:= [графический интерфейс библиотеки многократно используемых компонентов ostis-систем]
⇒ *включение**:
SCg-интерфейс библиотеки многократно используемых компонентов ostis-систем
}
}

решатель задач библиотеки многократно используемых компонентов ostis-систем реализует функциональные возможности библиотеки ostis-систем.

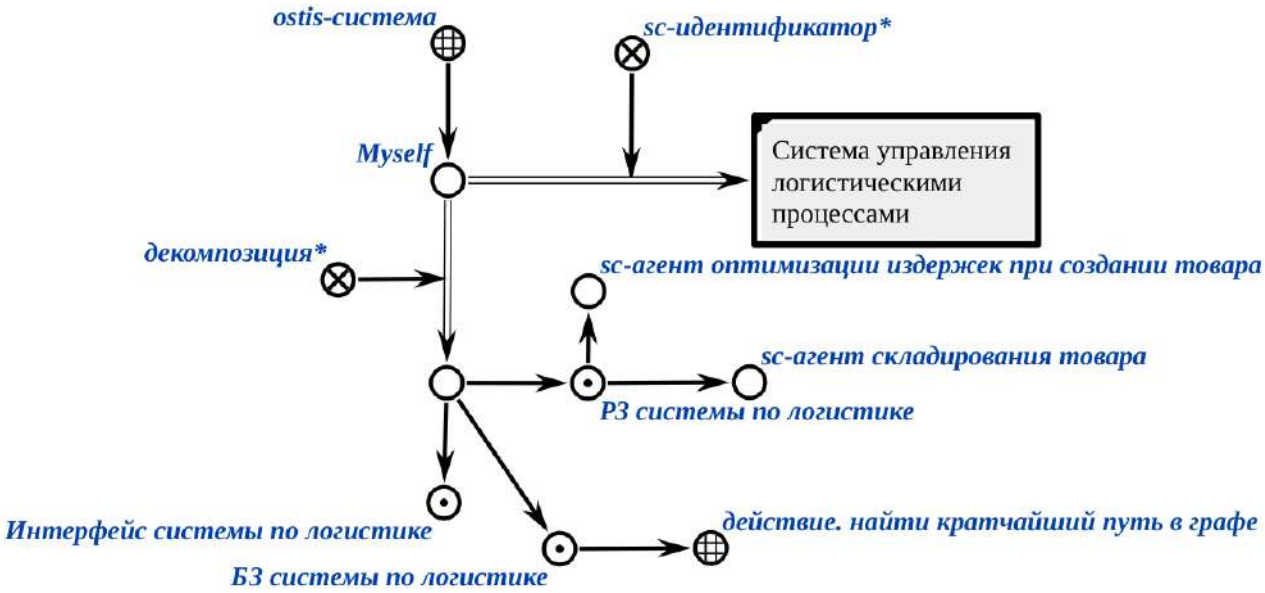
база знаний библиотеки многократно используемых компонентов ostis-систем представляет собой иерархию многократно используемых компонентов ostis-систем и их спецификаций.

интерфейс библиотеки многократно используемых компонентов ostis-систем обеспечивает доступ к многократно используемым компонентам и возможностям библиотеки ostis-систем для пользователя и других систем (см. *Корончик Д.Н.СеманТКПП-2011ст*). Существует минимальный и расширенный интерфейс библиотеки многократно используемых компонентов ostis-систем. *минимальный интерфейс библиотеки компонентов* позволяет *менеджеру многократно используемых компонентов ostis-систем*, входящему в состав какой-либо дочерней ostis-системы, подключиться к библиотеке многократно используемых компонентов ostis-систем и использовать ее функциональные возможности, то есть, например, получить доступ к спецификации компонентов и установить выбранные компоненты в *дочернюю ostis-систему*, получить сведения о доступных версиях компонента, его зависимостях и так далее. *расширенный интерфейс библиотеки компонентов*, в отличие от минимального интерфейса, позволяет не только получить доступ к компонентам для дальнейшей работы с ними, но и просматривать существующую структуру библиотеки, а также компоненты и их элементы в удобном и интуитивно понятном для пользователя виде.

Проблема интеграции многократно используемых компонентов ostis-систем решается путем взаимодействия компонентов через общую базу знаний. Компоненты могут лишь использовать общие ключевые узлы (понятия) в базе знаний (см. *Давыденко И.Т.ТехноКПБЗнО-2013ст* , *Ивашенко В.П.УнифиПиИЗ-2013ст* и *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*). Интеграция многократно используемых компонентов ostis-систем сводится к отождествлению ключевых узлов (см. *Пункт 2.2.1.4. Онтологическая формализация Базовой денотационной семантики SC-кода*) и устранению возможных дублирований и противоречий исходя из спецификации компонента и его содержания. Такой способ интеграции компонентов позволяет разрабатывать их параллельно и независимо друг от друга, что значительно сокращает сроки проектирования. Отождествление sc-элементов происходит в ходе выполнения *действие. отождествить два указанных sc-элемента*, которое рассматривается в *Главе 3.3.* Автоматическая интеграция компонентов *интеллектуальных систем* представляет широкие возможности для существенного сокращения сроков проектирования *интеллектуальных систем*, поскольку позволяет использовать опыт прошлых разработок.

Рассмотрим пример интеграции многократно используемого компонента *решателей задач*, который представляет собой sc-агент поиска кратчайшего пути в графе, в систему управления логистическими процессами. Допустим, система управления логистическими процессами умеет решать задачи, связанные с оптимизацией издержек в процессе создания товара и со складированием товаров.

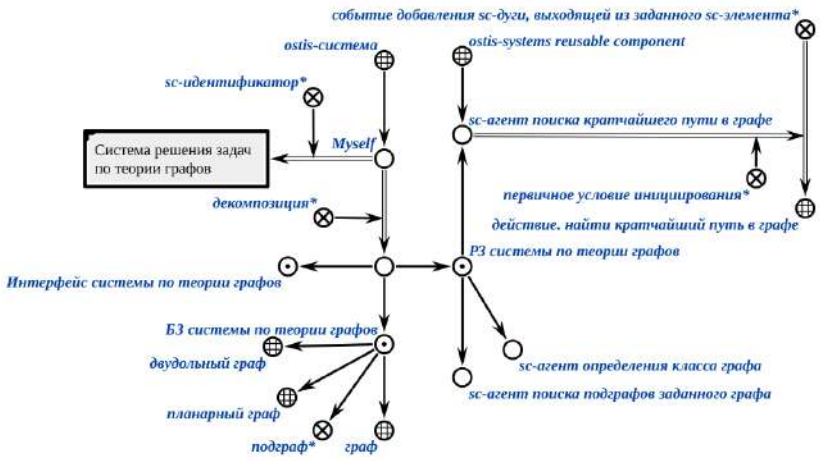
SCg-текст. Структура системы управления логистическими процессами
=



При этом в базе знаний системы описано, какие задачи должна решать система и соответствующие им действия. Например, действие. найти кротчайший путь в графе, которое система пока что не умеет выполнять.

Система решения задач по теории графов имеет богатую *базу знаний* и *решатель задач*, в том числе имеет sc-агент поиска кратчайшего пути в графе, который специфицирован как многократно используемый компонент и может быть использован в системе по логистике.

SCg-текст. Структура системы решения задач по теории графов
=



В результате установки *многократно используемого компонента* в виде sc-агента поиска кратчайшего пути в графе вся его sc-модель погружается в систему решения задач по логистике. При интеграции многократно используемого компонента, который представляет собой sc-агент поиска кратчайшего пути в графе, в систему по логистике ключевой узел *действие. найти кратчайший путь в графе* системы по логистике отождествляется с таким же узлом из установленного компонента из системы по теории графов. Таким образом, при выполнении логистических

задач, система сможет интерпретировать действие по поиску кратчайших путей с помощью интегрированного компонента.

Интеграция любых компонентов ostis-систем происходит автоматически, без вмешательства разработчика. Это достигается за счет использования *SC-кода* и его преимуществ, унификации спецификации многократно используемых компонентов и тщательного отбора компонентов в библиотеках экспертным сообществом, ответственным за эту библиотеку.

§ 5.1.3. Понятие многократно используемого компонента ostis-систем

⇒ *ключевой знак**:
- *Пример многократно используемого компонента баз знаний*
- *Пример шаблона семантической окрестности отношения*
- *Пример обязательной части спецификации многократно используемого компонента ostis-систем*
- *Пример необязательной части спецификации многократно используемого компонента ostis-систем*

⇒ *ключевое понятие**:
- *многократно используемый компонент ostis-систем*
- *компонент ostis-системы*
- *отношение, специфицирующее многократно используемый компонент ostis-систем*
- *встраиваемая ostis-система*

⇒ *ключевое знание**:
- *Отличие многократно используемого компонента ostis-систем от компонента ostis-системы*
- *Требования, предъявляемые к многократно используемым компонентам ostis-систем*
- *Классификация многократно используемых компонентов ostis-систем*
- *спецификация многократно используемых компонентов ostis-систем*
- *Метод установки динамически устанавливаемого многократно используемого компонента ostis-систем*
- *Метод установки многократно используемого компонента, при установке которого система требует перезапуска*

Рассмотрим понятие ***многократно используемого компонента ostis-систем***. Под многократно используемым компонентом ostis-систем понимается компонент некоторой ostis-системы, который может быть использован в рамках другой ostis-системы (см. *Шункевич Д.В..СредстваПКПС-2015ст*). Это компонент ostis-системы, который может быть использован в других ostis-системах (***дочерних ostis-системах***) и содержит все те и только те sc-элементы, которые необходимы для функционирования компонента в дочерней ostis-системе. Другими словами это компонент некоторой ***материнской ostis-системы***, который может быть использован в некоторой дочерней ostis-системе. Для включения многократно используемого компонента в некоторую систему, его необходимо установить в эту систему, то есть скопировать в нее все sc-элементы компонента и, при необходимости, вспомогательные файлы, такие как исходные или скомпилированные файлы компонента. *многократно используемые компоненты* должны иметь унифицированную спецификацию и иерархию для поддержки совместимости с другими компонентами. Совместимость многократно используемых компонентов приводит систему к новому качеству, к дополнительному расширению множества решаемых задач при интеграции компонентов.

многократно используемый компонент ostis-систем
:= [типовой компонент ostis-систем]
:= [повторно используемый компонент ostis-систем]
:= [многократно используемый компонент OSTIS]
:= [ip-компонент ostis-систем]
:= *часто используемый sc-идентификатор**:
[многократно используемый компонент]
⊂ *компонент ostis-системы*
⊂ *sc-структура*

компонент ostis-системы — это целостная часть *ostis-системы*, которая содержит все те (и только те) sc-элементы, которые необходимы для ее функционирования в ostis-системе. *многократно используемый компонент ostis-систем* — это общий компонент (общая часть) для некоторого множества ostis-систем, который многократно используется, дублируется и входит в состав некоторого множества ostis-систем. ***Отличие многократно используемого компонента ostis-систем от компонента ostis-системы*** в том, что многократно используемый компонент имеет спецификацию, достаточную для установки этого компонента в *дочернюю ostis-систему*.

Спецификация является частью *базы знаний библиотеки многократно используемых компонентов* соответствующей материнской ostis-системы. Есть техническая возможность встроить *многократно используемый компонент ostis-системы* в *дочернюю ostis-систему*.

Требования, предъявляемые к многократно используемым компонентам ostis-систем, наследуют общие требования к проектированию программных компонентов, а также включают следующие:

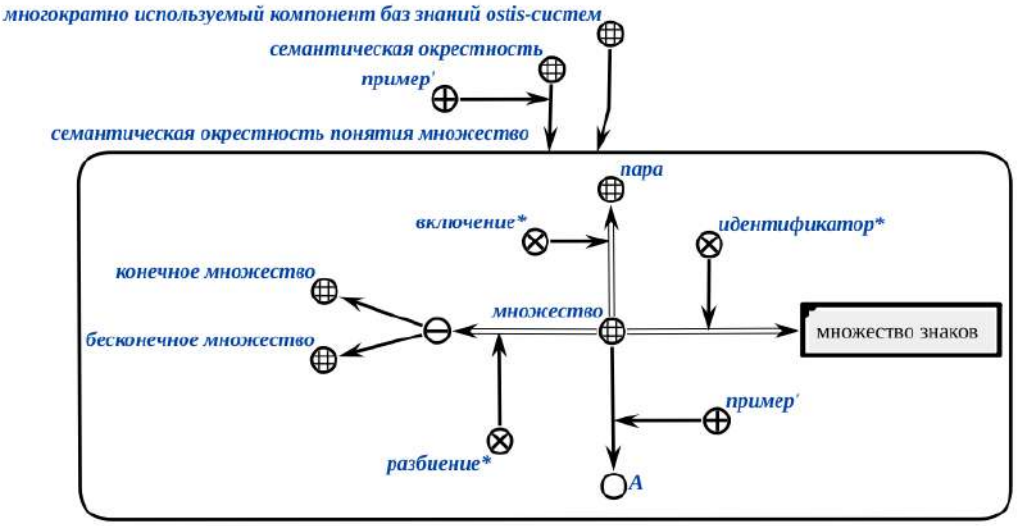
- существует техническая возможность встроить многократно используемый компонент в дочернюю ostis-систему;
- многократно используемый компонент должен выполнять свои функции наиболее общим образом, чтобы круг возможных систем, в которые он может быть встроен, был наиболее широким;
- совместимость многократно используемого компонента: компонент должен стремиться повышать уровень договороспособности ostis-систем, в которые он встроен и иметь возможность автоматической интеграции в другие системы;
- самодостаточность компонентов (или групп компонентов) технологии, то есть способности их функционировать отдельно от других компонентов без утраты целесообразности их использования.

Рассмотрим *классификацию многократно используемых компонентов ostis-систем* (см. *Orlov M.K.ComprLoRSC-2022art* и *Шункевич Д.В..СредстваПКПС-2015ст*). Класс многократно используемого компонента ostis-систем является важной частью спецификации компонента, позволяющей лучше понять назначение и область применения данного компонента, а также класс многократно используемого компонента является важнейшим критерием поиска компонентов в библиотеке ostis-систем. Основной признак классификации многократно используемых компонентов является признак *предметной области*, к которой относится компонент. Здесь структура может быть довольно богатой в соответствии с иерархией областей человеческой деятельности. Существует также множество предметно-независимых многократно используемых компонентов, которые могут использоваться в любой предметной области.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
{•
многократно используемый компонент баз знаний ostis-систем
:= [многократно используемый компонент баз знаний]
⊃ *семантическая окрестность*
∋ *Семантическая окрестность города Минска*
∋ *Семантическая окрестность понятия множество*
⊃ *предметная область и онтология*
∋ *Предметная область и онтология треугольников*
⊃ *шаблон многократно используемого компонента ostis-систем*
∋ *Шаблон описания предметной области*
∋ *Шаблон описания отношения*
• *многократно используемый компонент решателей задач ostis-систем*
:= [многократно используемый компонент решателей задач]
⊃ *атомарный абстрактный sc-агент*
∋ *Абстрактный sc-агент подсчета мощности множества*
⊃ *программа обработки знаний*
⊃ *scp-машина*
⊃ *scl-машина*
• *многократно используемый компонент интерфейсов ostis-систем*
:= [многократно используемый компонент интерфейсов]
⊃ *многократно используемый компонент пользовательских интерфейсов ostis-систем*
}

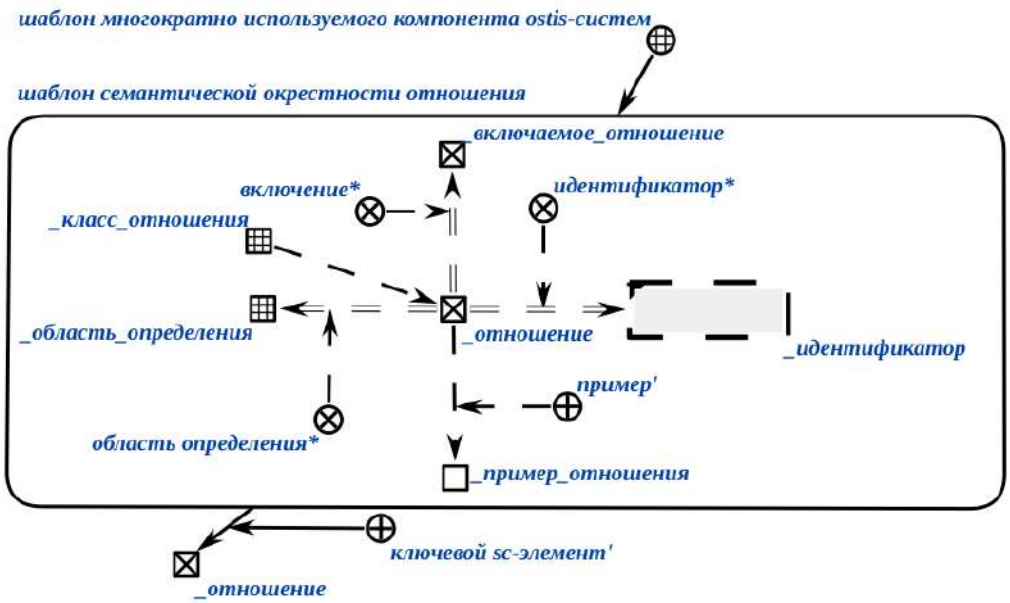
На рисунке *SCg-текст. Пример многократно используемого компонента баз знаний* приведен фрагмент *многократно используемого компонента баз знаний* на примере семантической окрестности понятия *множество*.

SCg-текст. Пример многократно используемого компонента баз знаний
=



На рисунке *SCg-текст. Пример шаблона семантической окрестности отношения* приведен фрагмент *шаблона многократно используемого компонента ostis-систем* на примере описания семантической окрестности отношения. Шаблоны многократно используемых компонентов также можно использовать для описания платформенно-независимых компонентов решателей задач и интерфейсов.

SCg-текст. Пример шаблона семантической окрестности отношения
=



Для компонентов баз знаний важнейшим признаком классификации многократно используемых компонентов является *вид используемых знаний*. Для компонентов решателей задач — *модель решения задач*. Для компонентов интерфейсов — *вид интерфейса* в соответствии с классификацией компонентов интерфейсов.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
Типология компонентов ostis-систем по атомарности^
= {• *атомарный многократно используемый компонент ostis-систем*
∋ *Абстрактный sc-агент подсчета мощности множества*
• *неатомарный многократно используемый компонент ostis-систем*

:= [составной многократно используемый компонент ostis-систем]
∋ *Решатель задач по геометрии*
}

атомарный многократно используемый компонент ostis-систем — это *многократно используемый компонент ostis-систем*, который в текущем состоянии библиотеки ostis-систем рассматривается как неделимый, то есть не содержит в своем составе других компонентов. Принадлежность *многократно используемого компонента ostis-систем* классу атомарных компонентов зависит от того, как специфицирован этот компонент и от существующих на данный момент компонентов в библиотеке. *неатомарный многократно используемый компонент* в текущем состоянии библиотеки ostis-систем содержит в своем составе другие атомарные или неатомарные компоненты, он не зависит от своих частей. Без какой-либо части неатомарного компонента его назначение сужается. В общем случае атомарный компонент может перейти в разряд неатомарных в случае, если будет принято решение выделить какую-то из его частей в качестве отдельного компонента. Все вышесказанное, однако, не означает, что даже в случае его платформенной независимости, атомарный компонент всегда хранится в sc-памяти как сформированная sc-структура. Например, платформенно-независимая реализация sc-агента всегда будет представлена набором *scp-программ*, но будет *атомарным многократно используемым компонентом OSTIS* в случае, если ни одна из этих программ не будет представлять интереса как самостоятельный компонент. В общем случае неатомарный компонент может перейти в разряд атомарных в случае, если будет принято решение по каким-либо причинам исключить все его части из рассмотрения в качестве отдельных компонентов. Следует отметить, что неатомарный компонент необязательно должен складываться полностью из других компонентов, в его состав могут также входить и части, не являющиеся самостоятельными компонентами. Например, в состав реализованного на *Языке SCP sc-агента*, являющего *неатомарным многократно используемым компонентом* могут входить как *scp-программы*, которые могут являться многократно используемыми компонентами (а могут и не являться), а также агентная *scp-программа*, которая не имеет смысла как многократно используемый компонент.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
Типология компонентов ostis-систем по зависимости^
= {• *зависимый многократно используемый компонент ostis-систем*
∋ *визуальный редактор системы по химии*
• *независимый многократно используемый компонент ostis-систем*
∋ *предметная область чисел*
}

зависимый многократно используемый компонент ostis-систем зависит хотя бы от одного другого компонента библиотеки ostis-систем и не может функционировать в дочерней ostis-системе без компонентов, от которых он зависит. *независимый многократно используемый компонент ostis-систем* не зависит ни от одного другого компонента библиотеки ostis-систем.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
Типология компонентов ostis-систем по способу их хранения^
= {• *многократно используемый компонент ostis-систем, хранящийся в виде внешних файлов*
⇒ *разбиение**:
{• *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
• *многократно используемый компонент ostis-систем, хранящийся в виде скомпилированных файлов*
}
• *многократно используемый компонент, хранящийся в виде sc-структуры*
}

На данном этапе развития *Технологии OSTIS* более удобным является хранение компонентов в виде исходных текстов.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
Типология компонентов ostis-систем по зависимости от ostis-платформы^
= {• *платформенно-зависимый многократно используемый компонент ostis-систем*
⊃ *ostis-платформа*
⊃ *абстрактный sc-агент, не реализуемый на Языке SCP*
• *платформенно-независимый многократно используемый компонент ostis-систем*

⊃ *многократно используемый компонент баз знаний*
⊃ *платформенно-независимый абстрактный sc-агент*
⊃ *scp-программа*
}

Под ***платформенно-зависимым многократно используемым компонентом ostis-систем*** понимается компонент, частично или полностью реализованный при помощи каких-либо сторонних с точки зрения *Технологии OSTIS* средств. Недостаток таких компонентов в том, что интеграция таких компонентов в интеллектуальные системы может сопровождаться дополнительными трудностями, зависящими от конкретных средств реализации компонента. В качестве возможного преимущества платформенно-зависимых многократно используемых компонентов ostis-систем можно выделить их, как правило, более высокую производительность за счет реализации их на более приближенном к платформе уровне. В общем случае платформенно-зависимый многократно используемый компонент ostis-систем может поставляться как в виде набора исходных кодов, так и в скомпилированном виде. Процесс интеграции платформенно-зависимого многократно используемого компонента ostis-систем в дочернюю систему, разработанную по *Технологии OSTIS*, сильно зависит от технологий реализации данного компонента и в каждом конкретном случае может состоять из различных этапов. Каждый платформенно-зависимый многократно используемый компонент ostis-систем должен иметь соответствующую подробную, корректную и понятную инструкцию по его установке и внедрению в дочернюю систему.

Под ***платформенно-независимым многократно используемым компонентом ostis-систем*** понимается компонент, который целиком и полностью представлен на *SC-коде*. В случае *неатомарного многократно используемого компонента* это означает, что все более простые компоненты, входящие в его состав также обязаны быть платформенно-независимыми многократно используемыми компонентами ostis-систем. Процесс интеграции платформенно-зависимого многократно используемого компонента ostis-систем в дочернюю систему, разработанную по Технологии OSTIS, существенно упрощается за счет использования общей унифицированной формальной основы представления и обработки знаний.

Наиболее ценными являются *платформенно-независимые многократно используемые компоненты ostis-систем*.

многократно используемый компонент ostis-систем
⇒ *разбиение**:
Типология компонентов ostis-систем по динамичности их установки^
= {• *динамически устанавливаемый многократно используемый компонент ostis-систем*
:= [многократно используемый компонент, при установке которого система не требует перезапуска]
⇒ *декомпозиция**:
{• *многократно используемый компонент, хранящийся в виде скомпилированных файлов*
• *многократно используемый компонент баз знаний*
}
• *многократно используемый компонент, при установке которого система требует перезапуска*
}

Процесс интеграции компонентов разных видов на разных этапах жизненного цикла *osits-систем* бывает разным. Наиболее ценными являются компоненты, которые могут быть интегрированы в работающую систему без прекращения ее функционирования. Некоторые системы, особенно системы управления, нельзя останавливать, а устанавливать и обновлять компоненты нужно.

встраиваемая ostis-система — это *неатомарный многократно используемый компонент*, который состоит из *базы знаний*, *решателя задач* и *интерфейса*.

встраиваемая ostis-система
⊂ *ostis-система*
⊂ *неатомарный многократно используемый компонент ostis-систем*
⇒ *декомпозиция**:
{• *многократно используемый компонент баз знаний ostis-систем*
• *многократно используемый компонент решателей задач ostis-систем*
• *многократно используемый компонент интерфейсов ostis-систем*
}

Такими системами могут быть, например, интеллектуальный интерфейс (в том числе естественно-языковой интерфейс), *средство коллективного проектирования баз знаний*, *менеджер многократно используемых компонентов ostis-систем*, *интеллектуальная обучающая система*, *система тестирования и верификации интеллектуальных систем*, *визуальный web-ориентированный редактор sc.g-текстов* и другие.

Особенность *встраиваемых ostis-систем* в том, что интеграция целых интеллектуальных систем предполагает интеграцию баз знаний этих систем, интеграцию их решателей задач и интеграцию их интеллектуальных интерфейсов. При интеграции встраиваемых ostis-систем база знаний встраиваемой системы становится частью базы знаний системы, в которую она встраивается. Решатель задач встраиваемой ostis-системы становится частью решателя задач системы, в которую она встраивается. И интерфейс встраиваемой ostis-системы становится частью интерфейса системы, в которую она встраивается. При этом встраиваемая система является целостной и может функционировать отдельно от других ostis-систем, в отличие от других многократно используемых компонентов.

встраиваемые ostis-системы зачастую являются предметно-независимыми многократно используемыми компонентами. Таким образом, например, встраиваемая ostis-система в виде среды проектирования баз знаний может быть встроена как в систему из предметной области по геометрии, так и в систему управления аграрными объектами.

встраиваемая ostis-система, как и любой другой многократно используемый компонент ostis-систем, должна поддерживать семантическую совместимость ostis-систем. Как сама встраиваемая ostis-система, так и все ее компоненты должны быть специфицированы и согласованы. Компоненты встраиваемых ostis-систем могут быть заменены на другие, имеющие то же назначение, например, естественно-языковой интерфейс может иметь различные варианты базы знаний в зависимости от естественного языка, поддерживаемого системой, различные варианты интерфейса, в зависимости от требований и удобства пользователей и также различные варианты реализации решателя задач для обработки естественного языка, которые могут использовать различные модели, однако решать одну и ту же задачу. Встраиваемая ostis-система связывается с системой, в которую она встроена с помощью отношения ***встроенная ostis-система****, которое является подмножеством отношения *встроенная кибернетическая система**.

Для того, чтобы хранить *многократно используемые компоненты ostis-систем*, необходимо некоторое хранилище. Таким хранилищем может выступать как какая-либо ostis-система, так и стороннее хранилище, например, облачное. Помимо внешних файлов компонента в хранилище должна находиться его спецификация.

Каждый *многократно используемый компонент ostis-систем* должен быть специфицирован в рамках библиотеки (см. *Давыденко И.Т.ТехноКПБЗнО-2013ст*). Данные спецификации включают в себя основные знания о компоненте, которые позволяют обеспечить построение полной иерархии компонентов и их зависимостей, а также обеспечивают беспрепятственную интеграцию компонентов в дочерние ostis-системы (см. *Orlov M.K.ComprLoRSC-2022art*). Для спецификации компонента используются, как отношения, так и классы компонента.

Чтобы многократно используемый компонент мог быть принят в библиотеку, нужно специфицировать его, используя отношения класса *необходимое для установки отношение, специфицирующее многократно используемый компонент ostis-систем*. Здесь описана спецификация, общая для любых типов компонентов, однако в зависимости от типа компонента, спецификация может расширяться (см. *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем, § 5.3.3. Многократно используемые компоненты решателей задач ostis-систем, § 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем*). Отношения класса *необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем* помогают лучше понять суть компонента, упрощают поиск, но не являются обязательными для того, чтобы компонент мог быть установлен в ostis-систему. Спецификация многократно используемого компонента ostis-систем также включает класс (тип) компонента. Указание класса *многократно используемый компонент ostis-систем* является обязательным. Сам многократно используемый компонент в рамках спецификации является *ключевым sc-элементом*, а также может иметь множество своих ключевых sc-элементов. Для уточнения типа компонента могут использоваться другие классы, например дата публикации первой и последней версии компонента, качественно-количественные характеристики, такие как уровень семантической совместимости компонентов, сложность реализации компонента, уровень производительности компонента (для программ можно использовать О-нотацию), количество sc-элементов, входящих в состав многократно используемого компонента, количество ключевых узлов компонента, рейтинг компонента в рамках *Экосистемы OSTIS*, количество скачиваний компонента и другие (см. *Давыденко И.Т.ТехноКПБЗнО-2013ст*). Параметр *лицензия многократно используемого компонента* используется для обозначения условий использования и распространения компонента. По умолчанию лицензия компонента открытая, если не указано иное.

отношение, специфицирующее многократно используемый компонент ostis-систем

:= [отношение, которое используется при спецификации многократно используемого компонента ostis-систем]

⇒ *разбиение**:

- {• *необходимое для установки отношение, специфицирующее многократно используемый компонент ostis-систем*
  - ∋ *метод установки**
  - ∋ *адрес хранилища**
  - ∋ *зависимости компонента**
- • *необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем*

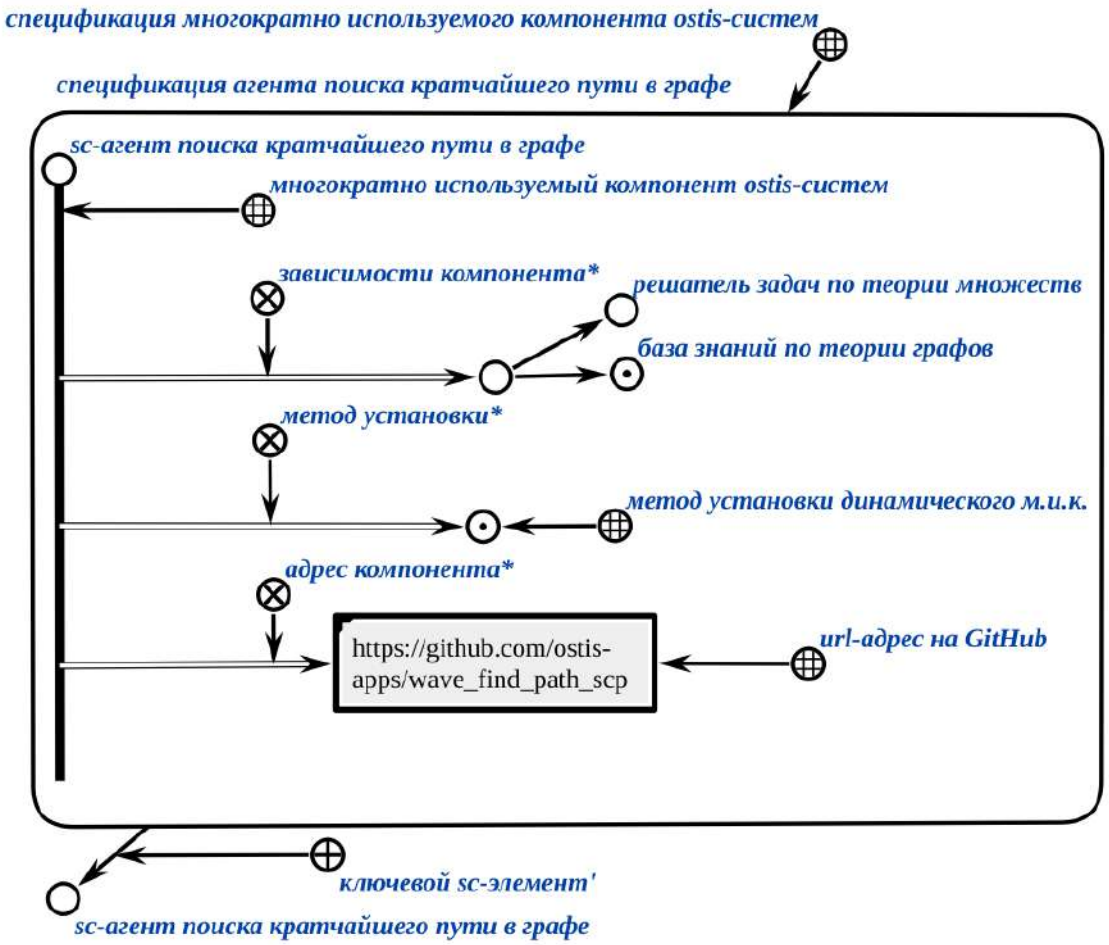
- ∋ *сопутствующие компоненты**
- ∋ *история изменений**
- ∋ *модификации компонентов**
- ∋ *авторы**
- ∋ *примечание**
- ∋ *пояснение**
- ∋ *идентификатор**
- ∋ *ключевой sc-элемент**
- ∋ *назначение**
- ∋ *требования полноты**
- ∋ *требования безошибочности**
- ∋ *преимущества**
- ∋ *недостатки**

}

На рисунке *SCg-текст. Пример обязательной части спецификации многократно используемого компонента ostis-систем* изображена обязательная часть спецификации, необходимой для установки sc-агента поиска кратчайшего пути в графе.

SCg-текст. Пример обязательной части спецификации многократно используемого компонента ostis-систем

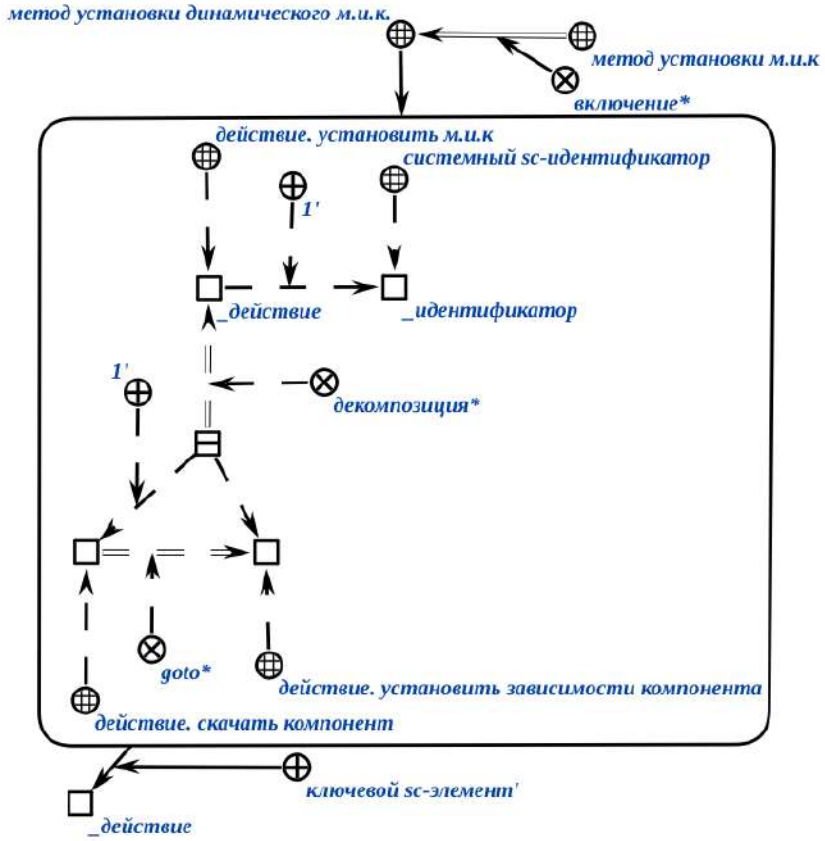
=



Метод установки позволяет пользователю установить компонент вручную, а ***менеджеру многократно используемых компонентов ostis-систем*** — автоматически. Основные два метода установки многократно используемых компонентов — метод установки динамически устанавливаемого многократно используемого компонента ostis-систем и метод установки многократно используемого компонента, при установке которого система требует перезапуска. При динамической установке необходимо только скачать компонент и, при необходимости, его зависимые компоненты, и он сразу же функционирует в системе.

SCg-текст. Метод установки динамически устанавливаемого многократно используемого компонента ostis-систем

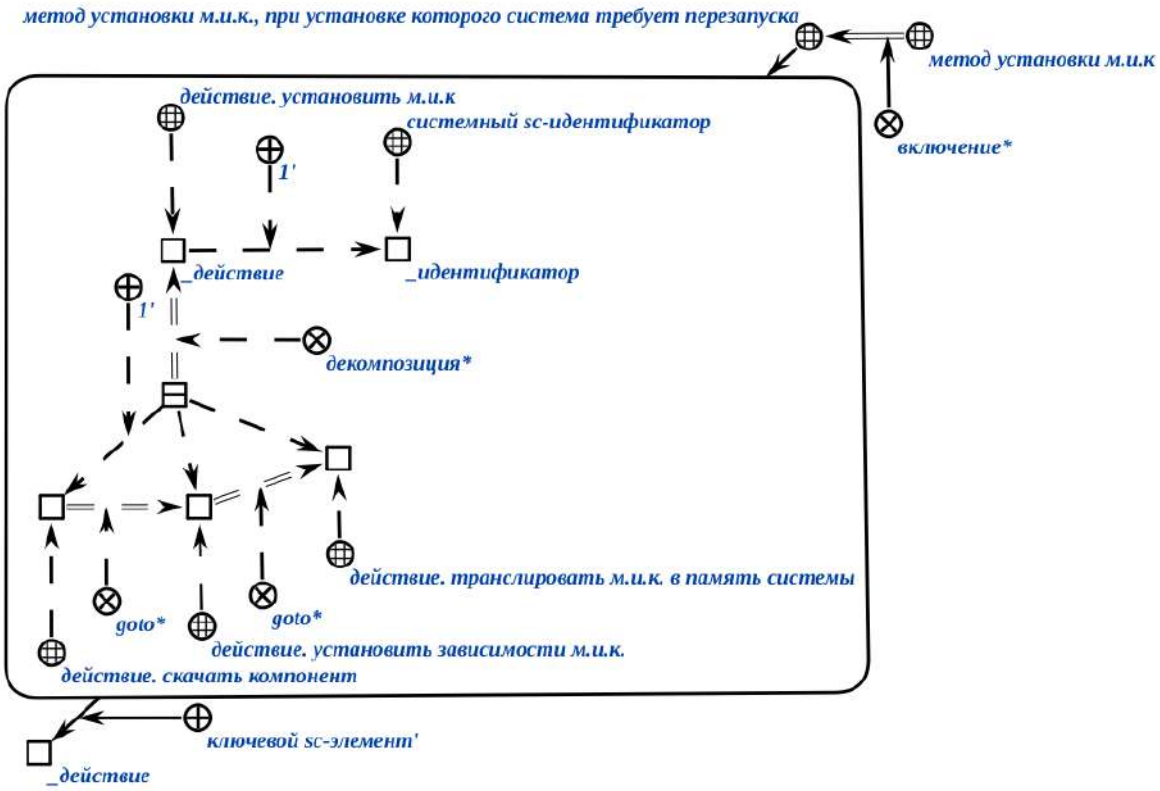
=



При установке компонента, при установке которого система требует перезапуска, необходимо помимо скачивания компонента транслировать его в память системы.

SCg-текст. Метод установки многократно используемого компонента, при установке которого система требует перезапуска

=



Связки отношения *адрес хранилища** связывают многократно используемый компонент, хранящийся в виде внешних файлов и файл, содержащий url-адрес *многократно используемого компонента ostis-систем*. Таким файлом может быть файл, содержащий url-адрес на GitHub *многократно используемого компонента ostis-систем*, файл, содержащий url-адрес на Google Drive многократно используемого компонента ostis-систем, файл, содержащий url-адрес на Docker Hub многократно используемого компонента ostis-систем, и другие.

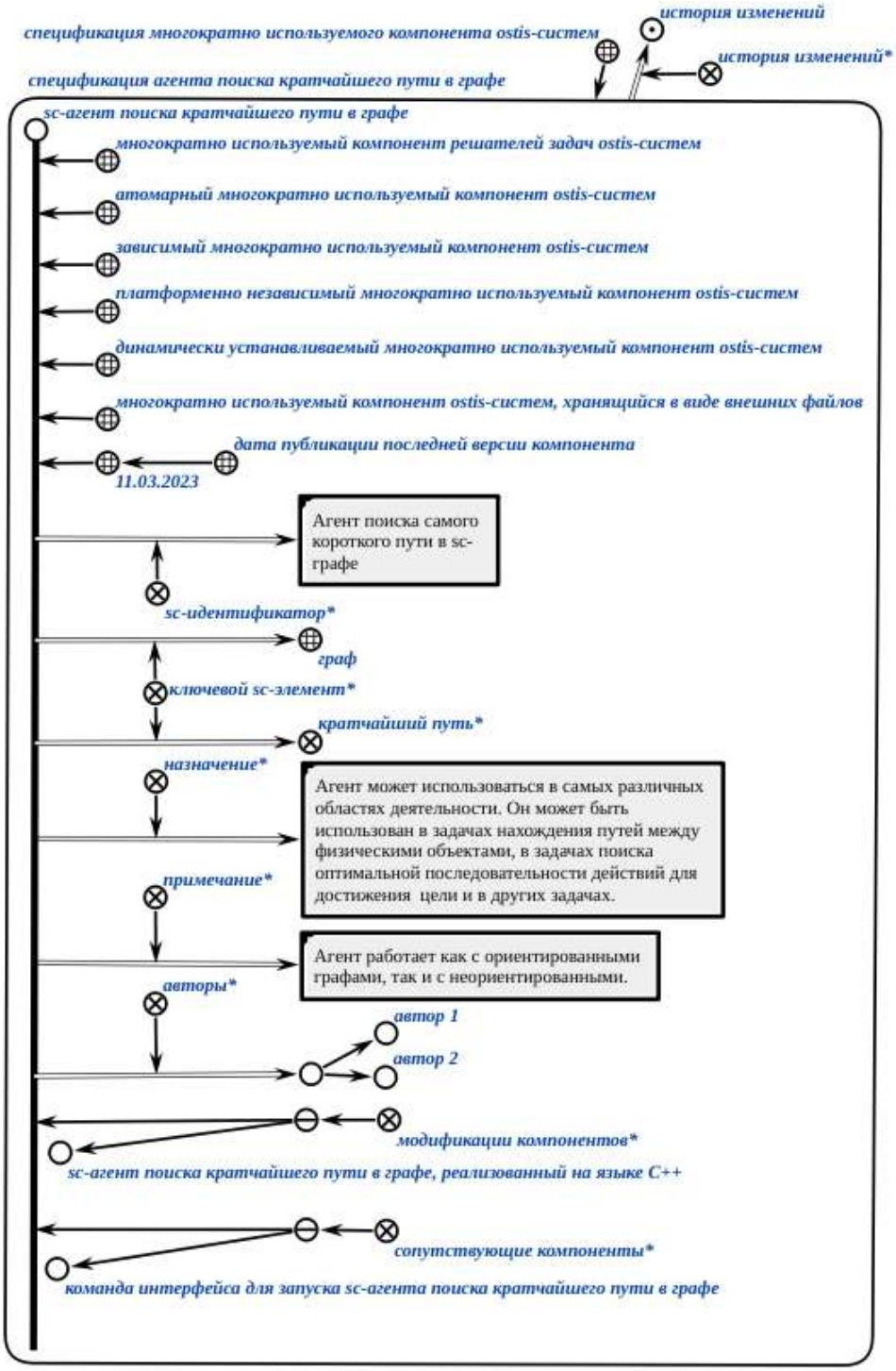
Связки отношения *зависимости компонента** связывают многократно используемый компонент и множество компонентов, без которых устанавливаемый компонент не может быть встроен в дочернюю ostis-систему.

Спецификация неатомарного многократно используемого компонента должна содержать информацию о том, из каких компонентов он состоит, используя отношение *декомпозиция**. При этом sc-структура, обозначающая неатомарный компонент не обязана содержать все sc-элементы компонентов, на которые она декомпозируется, достаточно, чтобы неатомарному компоненту принадлежали знаки всех тех компонентов, из которых он состоит (должно быть полное перечисление составных компонентов).

На рисунке *SCg-текст. Пример необязательной части спецификации многократно используемого компонента ostis-систем* изображена необязательная часть спецификации, необходимой для установки sc-агента поиска кратчайшего пути в графе. Полная спецификация компонента является результатом объединения обязательной и необязательной частей. Другие примеры спецификаций многократно используемых компонентов ostis-систем описаны в *§ 7.8.6. Многократно используемые компоненты интеллектуальных геоинформационных ostis-систем*.

SCg-текст. Пример необязательной части спецификации многократно используемого компонента ostis-систем

=



В некоторых случаях может оказаться, что для использования одного многократно используемого компонента OSTIS целесообразно или даже необходимо дополнительно использовать несколько других *многократно используемых компонентов OSTIS*. Например, может оказаться целесообразным вместе с каким либо sc-агентом информационного поиска использовать соответствующую команду интерфейса, которая представлена отдельным компонентом и позволит пользователю задавать вопрос для указанного sc-агента через интерфейс системы. В таких случаях для связи компонентов используется отношение *сопутствующие компоненты**. Наличие таких связей позволяет устранить возможные проблемы неполноты знаний и навыков в дочерней системе, из-за которых какие-либо из компонентов могут не выполнять свои функции. Связки отношения *сопутствующий компонент** связывают многократно используемые компоненты ostis-систем, которые целесообразно использовать в дочерней системе вместе. Каждая такая связка может дополнительно быть снабжена sc-комментарием или sc-пояснением,

отражающим суть указываемой зависимости. Отношение *история изменений** позволяет специфицировать различные версии компонента и, при необходимости, устанавливать выбранную пользователем версию. Различные версии, как правило, отражают какие-либо улучшения или исправления ошибок. *модификации компонентов** — это функционально эквивалентные реализации одного и того же компонента, которые могут быть синтаксически эквивалентны (например, реализация одного и того же sc-агента на платформенно-зависимом и платформенно-независимом уровнях). Развитие *Библиотеки Экосистемы OSTIS* происходит не только за счет ее пополнения новыми компонентами, но и за счет появления новых версий и модификаций уже существующих компонентов. Связки отношения *авторы** связывают многократно используемый компонент со множеством авторов этого компонента. Спецификация может также содержать дополнительную информацию об авторах при необходимости. Отношения *примечание** и *пояснение** связывают многократно используемый компонент и некоторый текст (как естественно-языковой, так и текст на SC-коде), который является примечанием или пояснением к этому компоненту. Идентификаторы компонента позволяют пользователям и разработчикам обозначать компонент, используя внешний язык. Ключевые sc-элементы компонента позволяют указать на наиболее важные понятия, связанные с описываемым компонентом. Отношение *назначение** позволяет описать ожидаемый сценарий, выделить рекомендации использования многократно используемого компонента. Требования полноты и безошибочности специфицируют возможные ограничения и ошибки компонента, область его использования.

§ 5.1.4. Менеджер многократно используемых компонентов ostis-систем

⇒ *ключевой знак**:
- *Реализация менеджера многократно используемых компонентов ostis-систем*
- *Пример структуры хранилища адресов спецификаций компонентов*

⇒ *ключевое понятие**:
- *менеджер многократно используемых компонентов ostis-систем*

⇒ *ключевое знание**:
- *Минимальные функциональные возможности менеджера компонентов*
- *Расширенные функциональные возможности менеджера компонентов*

менеджер многократно используемых компонентов ostis-систем — подсистема ostis-системы, с помощью которой происходит взаимодействие с библиотекой компонентов ostis-систем.

менеджер многократно используемых компонентов ostis-систем
⊂ *встраиваемая ostis-система*
⊂ *платформенно-зависимый многократно используемый компонент ostis-систем*
:= *часто используемый sc-идентификатор**:
[менеджер многократно используемых компонентов]
:= *часто используемый sc-идентификатор**:
[менеджер компонентов]
⇒ *обобщенная декомпозиция**:
{• *база знаний менеджера многократно используемых компонентов ostis-систем*
• *решатель задач менеджера многократно используемых компонентов ostis-систем*
• *интерфейс менеджера многократно используемых компонентов ostis-систем*
}
∋ *Реализация менеджера многократно используемых компонентов ostis-систем*
⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-component-manager]

база знаний менеджера многократно используемых компонентов ostis-систем содержит все те знания, которые необходимы для установки многократно используемого компонента в дочернюю ostis-систему. К таким знаниям относятся знания о спецификации многократно используемых компонентов, методы установки компонентов, знание о библиотеках ostis-систем, с которыми происходит взаимодействие, *Классификация компонентов* и другие. ***решатель задач менеджера многократно используемых компонентов ostis-систем*** взаимодействует с *библиотекой ostis-систем* и позволяет установить и интегрировать многократно используемые компоненты в дочернюю ostis-систему, также выполнять поиск, обновление, публикацию, удаление компонентов и другие операции с ними. ***интерфейс менеджера многократно используемых компонентов ostis-систем*** обеспечивает удобное для пользователя и других систем использование менеджера компонентов.

решатель задач менеджера многократно используемых компонентов ostis-систем как минимум должен обеспечивать следующие функциональные возможности:

менеджер многократно используемых компонентов ostis-систем
⇒ *минимальные функциональные возможности**:
Минимальные функциональные возможности менеджера компонентов
= {• **[Поиск многократно используемых компонентов ostis-систем.** Множество возможных критериев поиска соответствует спецификации многократно используемых компонентов. Такими критериями могут быть классы компонента, его авторы, идентификатор, фрагмент примечания, назначение, принадлежность какой-либо предметной области, вид знаний компонента и другие.**]**
• **[Установка многократно используемого компонента ostis-систем.** Установка многократно используемого компонента происходит вне зависимости от типологии, способа установки и местоположения компонента. Необходимое условие для возможности установки многократно используемого компонента — наличие ***спецификации многократно используемого компонента ostis-систем***. Перед установкой многократно используемого компонента в дочернюю систему необходимо установить все зависимые компоненты. Также для платформенно-зависимых компонентов может быть необходимо установить иные зависимости, которые не являются компонентами какой-либо библиотеки ostis-систем. После успешной установки компонента в базе знаний дочерней системы генерируется информационная конструкция, обозначающая факт установки компонента в систему с помощью отношения *установленные компоненты**.**]**
• **[Добавление и удаление отслеживаемых менеджером компонентов библиотек.** Менеджер компонентов содержит информацию о множестве источников для установки компонентов, перечень которых можно дополнять. По умолчанию менеджер компонентов отслеживает *Библиотеку Метасистемы OSTIS*, однако можно создавать и добавлять дополнительные библиотеки ostis-систем.**]**
}

Исходя из указанных минимальных функциональных возможностей *решатель задач менеджера многократно используемых компонентов ostis-систем* представляет собой следующую иерархию абстрактных sc-агентов:

решатель задач менеджера многократно используемых компонентов ostis-систем
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент поиска многократно используемых компонентов ostis-систем*
• *Абстрактный sc-агент установки многократно используемых компонентов ostis-систем*
• *Абстрактный sc-агент управления отслеживаемых менеджером компонентов библиотек*
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент добавления отслеживаемой менеджером компонентов библиотеки*
• *Абстрактный sc-агент удаления отслеживаемой менеджером компонентов библиотеки*
}
}

Используя минимальные функциональные возможности менеджер компонентов может установить компоненты, которые будут расширять его же функционал. Компоненты, реализующие расширенный функционал менеджера компонентов являются частью *Библиотеки Метасистемы OSTIS*. К расширенному функционалу относится:

менеджер многократно используемых компонентов ostis-систем
⇒ *расширенные функциональные возможности**:
Расширенные функциональные возможности менеджера компонентов
= {• **[Спецификация** многократно используемого компонента ostis-систем. Менеджер компонентов позволяет специфицировать компоненты, входящие в состав библиотеки ostis-систем, а также специфицировать новые создаваемые компоненты, которые будут публиковаться в библиотеку ostis-систем. При этом спецификация может происходить как автоматически, так и вручную. Например, менеджер компонентов может обновить спецификацию используемого компонента в соответствии с тем, в какие новые ostis-системы его установили, обновить спецификацию авторства компонента при его редактировании в библиотеке ostis-систем, спецификацию ошибок, выявленных в процессе эксплуатации компонента и так далее.**]**
• **[Формирование** многократно используемого компонента ostis-систем по шаблону с заданными параметрами. При установке шаблона многократно используемого компонента ostis-систем менеджер компонентов позволяет сформировать по нему конкретный компонент. Для этого пользователю предлагается определить значения всех sc-переменных в шаблоне для формирования конкретного компонента из некоторой предметной области. Например, для формирования многократно используемого компонента баз знаний, представляющего собой семантическую окрестность некоторого отношения (см. рисунок *SCg-текст. Пример шаблона семантической окрестности отношения*), нужно определить значения всех переменных, кроме переменной, являющейся ключевым sc-элементом данной структуры.**]**

- **[Публикация** многократно используемого компонента ostis-систем в библиотеку ostis-систем. При публикации компонента в библиотеку ostis-систем происходит верификация на основе спецификации компонента. Публикация компонента может сопровождаться сборкой неатомарного компонента из существующих атомарных. Также существует возможность обновления версии опубликованного компонента сообществом его разработчиков.**]**
- **[Обновление** установленного многократно используемого компонента ostis-систем.**]**
- **[Удаление** установленного многократно используемого компонента. Как и в случае установки после удаления многократно используемого компонента из ostis-системы в базе знаний системы устанавливается факт удаления компонента. Эта информация является важной частью истории эксплуатации ostis-системы.**]**
- **[Редактирование** многократно используемого компонента в библиотеке ostis-систем.**]**
- **[Сравнение** многократно используемых компонентов ostis-систем.**]**

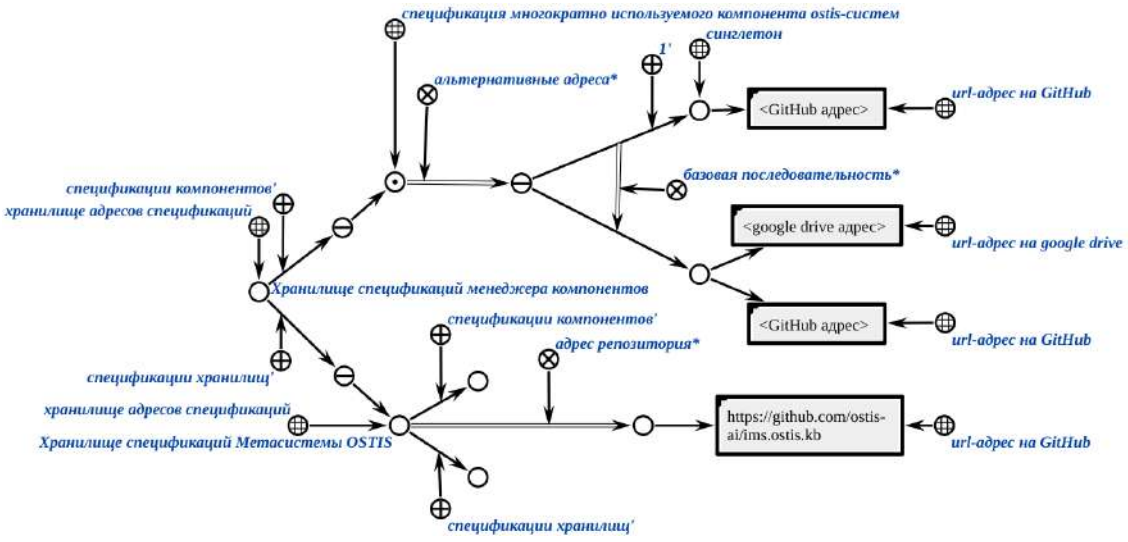
}

Для того, чтобы создать новую ostis-систему "с нуля", используя *ostis-платформу* (см. *§ 6.1.3. Уточнение понятия ostis-платформы*), необходимо установить некоторый *Программный вариант реализации ostis-платформы* (см. *Главу 6.3. Программная платформа ostis-систем*) с помощью сторонних средств. Такими средствами могут быть (1) хранилища исходного кода платформы, например, облачные хранилища, такие как GitHub репозиторий, с соответствующим набором инструкций по установке платформы или (2) средства установки заранее скомпилированной программной реализации платформы, например, средство установки программного обеспечения apt. Далее установка многократно используемых компонентов в ostis-систему (независимо от типа компонентов) осуществляется с помощью менеджера компонентов. При установке платформенно-зависимых компонентов менеджер компонентов должен управлять соответствующими средствами сборки таких компонентов (CMake, Ninja, npm, grunt и другие).

Компонент находится в некотором хранилище — (1) *библиотеке компонентов ostis-систем* или (2) в виде файлов в некотором облачном хранилище. В случае, когда компонент хранится в библиотеке, для его установки менеджер компонентов копирует все sc-элементы, которые представляют собой компонент, в дочернюю ostis-систему. В случае, когда компонент хранится в виде файлов в облачном хранилище, менеджер компонентов скачивает файлы компонента и устанавливает их в соответствии со спецификацией. Адреса хранилищ спецификаций компонентов должны храниться в базе знаний менеджера компонентов, чтобы иметь доступ к спецификациям компонентов для их последующего использования (поиска, установки и так далее). На рисунке *SCg-текст. Пример структуры хранилища адресов спецификаций компонентов* приведен пример фрагмента базы знаний менеджера компонентов, который описывает то, где хранятся спецификации компонентов, доступных для установки. Такое хранилище представляет собой множество, состоящее из двух множеств: (1) множество адресов спецификаций компонентов и (2) множество адресов спецификаций других хранилищ. Таким образом, образуется древовидная структура в соответствии с иерархией материнских ostis-систем и соответствующих им библиотек.

SCg-текст. Пример структуры хранилища адресов спецификаций компонентов

=



Если указать адрес корня дерева хранилищ адресов спецификаций, то менеджер компонентов получает доступ ко всем спецификациям дочерних хранилищ. При обработке такого дерева спецификаций менеджер компонентов погружает в sc-память спецификации компонентов, доступных для установки, однако не сами эти компоненты.

менеджер многократно используемых компонентов ostis-систем является необязательной подсистемой *ostis-платформы*. Однако система, имеющая менеджер компонентов, может устанавливать компоненты не только сама в себя, но и в другие системы при наличии доступа. Таким образом, одна система может заменить *ostis-платформу* другой системы, оставив при этом *sc-модель кибернетической системы* (см. *§ 6.1.2. Методы и средства реализации ostis-систем*). Таким же образом некоторая ostis-система может порождать другие ostis-системы, используя компонентный подход.

Включение компонента в *дочернюю ostis-систему* в общем случае состоит из следующих этапов:
- поиск подходящего компонента во множестве доступных библиотек;
- выделение компонента в виде, удобном для транспортировки в дочернюю ostis-систему с указанием версии и модификации при необходимости (например, выбор доступного хранилища компонента, выбор оптимального варианта реализации компонента с учетом состава дочерней системы);
- установка многократно используемого компонента и его зависимостей (с указанием версии и модификации при необходимости);
- интеграция компонента в дочернюю систему;
- поиск и устранение ошибок и противоречий в дочерней системе.

Данный процесс с точки зрения пользователя не зависит от типа компонента и особенностей его реализации.

Заключение к Главе 5.1.

Компонентный подход является ключевым в технологии проектирования интеллектуальных компьютерных систем. Вместе с этим, технология компонентного проектирования тесно связана с остальными составляющими *технологии проектирования интеллектуальных компьютерных систем* и обеспечивает их совместимость, производя мощнейший синергетический эффект при использовании всего комплекса частных технологий проектирования интеллектуальных систем. Важнейшим принципом в реализации компонентного подхода является семантическая совместимость многократно используемых компонентов, что позволяет минимизировать участие программистов в создании новых компьютерных систем и в совершенствовании существующих.

Для реализации компонентного подхода в работе предложена *библиотека многократно используемых совместимых компонентов интеллектуальных компьютерных систем на основе Технологии OSTIS*, введена классификация и спецификация многократно используемых компонентов ostis-систем, предложена модель менеджера компонентов, позволяющего взаимодействовать *ostis-системам* с *библиотеками многократно используемых компонентов* и управлять компонентами в системе, рассмотрена архитектура экосистемы *интеллектуальных компьютерных систем* с точки зрения использования библиотеки многократно используемых компонентов.

Полученные результаты позволят повысить эффективность проектирования интеллектуальных систем и средств автоматизации разработки таких систем, а также обеспечить возможность не только разработчику, но и интеллектуальной системе автоматически дополнять систему новыми знаниями и навыками.

Глава 5.2.

Методика и средства проектирования и анализа качества баз знаний ostis-систем

⇒ *эпиграф**:
[Дырявая и запутанная сеть хорошего улова не принесет.]

⇒ *автор**:
- *Бутрин С.В.*
- *Шункевич Д.В.*
- *Банцевич К.А.*

⇒ *аннотация**:
[В главе рассматриваются актуальные проблемы текущего состояния средств проектирования и анализа качества *баз знаний*, предложен подход к их решению на основе *Технологии OSTIS*. Сформулированы принципы коллективного проектирования и разработки *баз знаний*. Сформулированы принципы верификации *баз знаний*.]

⇒ *подраздел**:
- *§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем*
- *§ 5.2.2. Индивидуальный аспект проектирования и разработки баз знаний ostis-систем*
- *§ 5.2.3. Коллективный аспект проектирования и разработки баз знаний ostis-систем*
- *§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы*
- *§ 5.2.5. Логико-семантическая модель ostis-системы автоматизации управления взаимодействием разработчиков различных категорий в процессе проектирования базы знаний ostis-системы*
- *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*

⇒ *ключевое понятие**:
- *база знаний*
- *редактор баз знаний*
- *разработка баз знаний*
- *разработчик*
- *противоречие*
- *информационная дыра*
- *многократно используемые компоненты*

⇒ *библиографическая ссылка**:
- *Давыденко.И.Т.СеманМКПБЗ-2016ст*
- *Давыденко И.Т.МоделМиСРГБ-2017дс*
- *Davydenko I.T.SemanMMaToKB-2018art*
- *KnowlBEST-el*
- *Аршинский Л.В..КомплВПБЗсИ-2020ст*
- *Rybina G..Metho aAfVoKB-2007art*
- *Zhang D.KnowlBVIaA-2023art*
- *Нариньяни А.С.нФактоКВ-2004ст*
- *Ивашенко В.П.СеманТКПБЗ-2011ст*
- *Давыденко И.Т.ТехноКПБЗнО-2013ст*

Введение в Главу 5.2.

Разработка *базы знаний* является трудоемким и продолжительным процессом, требующим высокого уровня квалификации *разработчиков баз знаний*. Данный факт приводит к высокой себестоимости как самих *баз знаний*, так и соответствующих им *интеллектуальных систем*, а также к дефициту специалистов в области *инженерии знаний*.

Расширение областей применения *интеллектуальных систем* требует поддержки решения комплексных задач. Решение каждой такой задачи предполагает совместное использование различных видов знаний и моделей их представления, что приводит к компенсации недостатков одних моделей возможностями и достоинствами других.

Существующие *средства создания баз знаний* предполагают, что процессы разработки и модификации базы знаний осуществляются отдельно от процесса ее использования, что приводит к дополнительному усложнению решения задачи обеспечения совместимости различного вида знаний. Отсутствие удовлетворительного решения этой задачи приводит к несовместимости *компонентов баз знаний*, разрабатываемых для разных систем, и невозможности их повторного использования в других системах. Данный факт приводит к многократной повторной разработке содержательно одних и тех же компонентов для разных баз знаний.

Таким образом, актуальной является задача разработки модели *баз знаний*, которая, с одной стороны, обеспечит общий унифицированный формальный фундамент для представления различных видов знаний в рамках одной *базы знаний* и их совместного использования при решении комплексных задач, а с другой стороны, обеспечит возможность расширения числа видов знаний, используемых *интеллектуальной системой* (см. *Давыденко.И.Т.СеманМКПБЗ-2016ст*).

§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем

База знаний является ключевым компонентом *интеллектуальной системы*, которая в систематизированном виде включает в себя все знания, необходимые *интеллектуальной системе* для ее функционирования.

Для задач, решаемых *интеллектуальными системами*, в общем случае неизвестно, какие данные и знания должны быть использованы для решения этих задач, какие должны быть использованы методы их решения. При решении таких задач необходима локализация *фрагмента базы знаний*, содержащего данные и знания, достаточные для решения задачи, и исключающего те данные и знания, которые заведомо для этого не нужны, а также выделение из имеющегося многообразия методов решения задач тех *методов*, которых достаточно для решения данной задачи.

Однако для обеспечения совместного использования различных видов знаний в единой *базе знаний* необходимо обеспечить совместимость этих видов знаний и, как следствие, совместимость *компонентов баз знаний*, которая включает два аспекта: обеспечение синтаксической совместимости, что подразумевает унификацию формы представления знаний, и обеспечение семантической совместимости, что подразумевает однозначную и единую для всех компонентов трактовку используемых понятий.

Существующие подходы к разработке *баз знаний*, как правило, предполагают решение задачи обеспечения синтаксической совместимости знаний путем соединения разнородных моделей представления знаний, а также разработки новых интегрированных моделей и новых языков представления знаний. Разработка *базы знаний* таким методом приводит к дополнительным накладным расходам при интеграции и обработке разнородных знаний, и, как следствие, к резкому увеличению трудозатрат при модификации таких *баз знаний* и добавлении новых видов знаний. Таким образом, *интеллектуальная система*, способная решать комплексные задачи, должна обладать способностью приобретать новые знания и навыки в процессе ее эксплуатации, сохраняя при этом *корректность* и *целостность базы знаний*.

В свою очередь, это обуславливает требование модицифируемости к такой *базе знаний*, то есть снижения трудоемкости внесения изменений в базу знаний. Таким образом, можно сформулировать требования к *базе знаний систем, способных решать комплексные задачи*:

- возможность согласованного использования различных видов знаний, хранимых в одной *базе знаний*, при решении каждой комплексной задачи;
- возможность реализации различных аспектов спецификации *сущностей*, описываемых в *базе знаний*;
- *модифицируемость базы знаний*, позволяющая непосредственно в процессе эксплуатации *интеллектуальной системы* добавлять в базу знаний новые фрагменты, в том числе новые виды знаний без внесения изменений в существующую структуру *базы знаний*;
- возможность неограниченного перехода в рамках каждой *базы знаний* от знаний к *метазнаниям*, от метазнаний к *метаметазнаниям* и так далее, что, в частности, предоставляет неограниченные возможности типологии и систематизации знаний, хранимых в составе *базы знаний*, и неограниченные возможности *декомпозиции* и *структуризации* самой базы знаний;
- наличие языковых средств, позволяющих в рамках базы знаний представлять метазнания, описывающие *качество базы знаний* (противоречия, неполноту, избыточность);
- наличие языковых средств, позволяющих в рамках базы знаний представлять метазнания, описывающие *историю эволюции* и *планы дальнейшей эволюции* базы знаний;
- возможность для каждой решаемой задачи явно задавать и уточнять в ходе решения задачи *область решения*, то есть такой фрагмент базы знаний, использование которого является достаточным для решения этой задачи.

А также следующие дополнительные требования к функциональности *системы поддержки коллективной разработки гибридных баз знаний*, учитывающие недостатки рассмотренных аналогов:

- обеспечение возможности как ручного, так и автоматического *редактирования баз знаний*;

- обеспечение возможности *автоматической верификации базы знаний*, в том числе анализ *корректности* и *полноты базы знаний*;
- обеспечение возможности создания *базы знаний* распределенным *коллективом разработчиков*, включая механизм согласования вносимых в *базу знаний* изменений, а также механизм хранения истории вносимых изменений с указанием авторства.

Реализация перечисленных возможностей подразумевает отказ от работы с *файлами исходных текстов базы знаний*. В данном случае предполагается, что все изменения осуществляются непосредственно в памяти системы, где хранится вся база знаний, что позволяет осуществлять разработку базы знаний компьютерной системы в процессе ее эксплуатации (см. *Давыденко И.Т.МоделМиСРГБ-2017дс*).

Проектирование базы знаний включает в себя два аспекта: индивидуальный и коллективный. Индивидуальный представляет собой наполнение изолировнной части *базы знаний* конкретным пользователем. Коллективный представляет собой согласование всей *базы знаний* и включает в себя процесс внесения предложений и внедрения изменений.

Таким образом индивидуальный аспект включает в себя средства и методы позволяющие пользователю непосредственно наполнять *базу знаний* посредством *редакторов*, *трансляторов* и так далее. При этом *процесс редактирования базы знаний* должен быть максимально удобным и понятным для пользователя.

Коллективный же аспект включает в себя средства и методы для автоматического или полуавтоматического *согласования общей базы знаний* с индивидуальными *базами знаний* пользователей. То есть он включает в себя процессы создания предложений по изменению базы знаний, их рассмотрение и внесение в общую базу знаний.

Так как в основе любой современной базы знаний лежат *онтологии*, то методы и средства разработки онтологий являются важнейшей частью технологий разработки баз знаний. Методология разработки онтологий представляет собой набор инструкций и руководств, описывающих процесс выполнения сложных процедур разработки онтологий. Она детализирует различные задачи, как они должны быть выполнены, в каком порядке и каким образом осуществлять документирование работы по созданию онтологий. Существующие методологии можно условно разделить на представленные ниже группы.

методология разработки онтологий
⇒ *разбиение**:
Типология методологий по поддержке коллективной разработки^
= {• *методология, поддерживающая совместную коллективную разработку онтологии*
• *методология, не поддерживающая совместную коллективную разработку онтологии*
}
⇒ *разбиение**:
Типология методологий по степени зависимости от инструментария^
= {• *методология, зависимая от инструментария*
• *методология, полузависимая от инструментария*
• *методология, независимая от инструментария*
}
⇒ *разбиение**:
Типология методологий по типу используемой модели жизненного цикла онтологии^
= {• *методология без указания модели жизненного цикла онтологии*
• *методология с итеративной моделью жизненного цикла онтологии*
• *методология с моделью жизненного цикла онтологии на основе эволюционного прототипирования*
• *методология с моделью жизненного цикла приложения*
}
⇒ *разбиение**:
Типология методологий по возможности формализации^
= {• *методология, предусматривающая методы формализации*
• *методология, не предусматривающая формализации*
}
⇒ *разбиение**:
Типология методологий по возможности повторного использования разрабатываемых онтологий^
= {• *методология, поддерживающая повторное использование*
• *методология, не поддерживающая повторного использования*
}
⇒ *разбиение**:
Типология методологий по стратегии выделения концептов предметной области^
= {• *методология снизу вверх (bottom-up)*

- *методология сверху вниз (top-down)*
- *методология от середины (middle-out)*
- *методология, сочетающая различные стратегии*

}

⇒ *разбиение**:

Типология методологий по возможности поддержки совместимости разрабатываемых онтологий^

= {• *методология, поддерживающая совместимость*
• *методология, не поддерживающая совместимость*
}

Большинство методологий не поддерживают совместную разработку *баз знаний*, поддержку совместимости разрабатываемых *баз знаний* и, как следствие, поддержку повторного использования уже разработанных *баз знаний* и их *компонентов*.

Кроме того, подавляющее большинство *методологий разработки баз знаний* описывают процесс разработки в общих чертах, не регламентируя действия участников на каждом этапе разработки *онтологии*, не уточняя принципы согласования новых *понятий* с уже существующими, высоким оказывается субъективное влияние разработчиков.

Отличительной особенностью предлагаемой методики от существующих *методологий разработки баз знаний* является совершенствование *базы знаний коллективом разработчиков* непосредственно в процессе ее использования, а также создание новых и использование уже имеющихся *компонентов баз знаний* в процессе разработки каждой *базы знаний*.

Данная методика предполагает два основных этапа — этап создания стартовой версии разрабатываемой *ostis-системы*, *база знаний* которой синтезируется из *компонентов*, входящих в *библиотеку многократно используемых компонентов баз знаний ostis-систем*(см. *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*), и этап расширения и совершенствования *базы знаний* разрабатываемой *ostis-системы*, осуществляемый в рамках этой системы.

Стартовая версия *ostis-системы* содержит набор знаний и средств решения задач, достаточный для дальнейшего развития системы.

Процесс создания стартовой версии ostis-системы можно разделить на пять основных этапов:

- выбор и установка *ostis-платформы* для интерпретации *sc-модели ostis-системы*(см. *Главу 6.3. Программная платформа ostis-систем*);
- установка *Ядра sc-модели базы знаний ostis-системы* из *библиотеки многократно используемых компонентов баз знаний*(см. *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*);
- установка *Ядра решателей задач из библиотеки многократно используемых компонентов решателей задач*, то есть набора базовых *многократно используемых компонентов решателей задач*, необходимых для работы стартовой версии *ostis-системы*(см. *§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем*);
- установка *Ядра sc-моделей интерфейсов*, то есть набора базовых *многократно используемых компонентов пользовательского интерфейса ostis-систем*, необходимых для работы стартовой версии *ostis-системы*(см. *§ 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем*);
- установка *системы поддержки коллективной разработки гибридных баз знаний*.

Процесс разработки базы знаний включает в себя следующие стадии:

- Формирование начальной структуры *гибридной базы знаний*, которая предполагает:
  - формирование структуры разделов базы знаний, соответствующей варианту структуризации *базы знаний* с точки зрения разработчиков;
  - выявление описываемых *предметных областей*;
  - построение иерархической системы описываемых *предметных областей*;
  - построение иерархии разделов *базы знаний* в рамках предметной части базы знаний, учитывающей построенную на предыдущем этапе иерархию *предметных областей*.
- Выявление *компонентов базы знаний*, которые могут быть заимствованы из *библиотеки многократно используемых компонентов баз знаний*, и включение их в состав разрабатываемой *базы знаний*;
- Формирование проектных заданий на разработку недостающих *фрагментов базы знаний* и распределение заданий между разработчиками;
- Разработка и согласование *фрагментов базы знаний*, которые, в свою очередь, могут в дальнейшем быть включены в состав *библиотеки многократно используемых компонентов баз знаний*;
- Верификация и отладка базы знаний.

Следует отметить, что в процессе совершенствования базы знаний этапы 3 - 5 выполняются циклически.

Для обеспечения свойства рефлексивности *интеллектуальной системы*, в частности, возможности автоматизации анализа истории эволюции базы знаний и генерации планов по ее развитию, вся деятельность, связанная с разработкой базы знаний, должна специфицироваться в самой этой базе знаний теми же средствами, что и предметная часть (см. *Davydenko I.T.SemanMMaToKB-2018art*).

§ 5.2.2. Индивидуальный аспект проектирования и разработки баз знаний ostis-систем

Для решения задачи индивудуального наполения базы знаний предлагается использовать специализированный инструментарий, который включает в себя различного рода редакторы и трансляторы.

Текущая реализация *ostis-платформы* и решателя задач поддерживает работу с файлами исходных текстов базы знаний. Для создания таких файлов исходных текстов на *SCs-коде* можно воспользоваться любым текстовым редактором.

Для создания файлов исходных текстов в *SCg-коде* может быть использован редактор ***KBE*** (Knowledge Base source Editor, см. *KnowlBEST-el*). *KBE* является приложением, которое направлено на помощь в создании и редактировании фрагментов баз знаний интеллектуальных систем, проектирование которых основано на *Технологии OSTIS*. В основу данного редактора положен принцип визуализации данных, хранящихся в базе знаний, что намного упрощает процесс их редактирования и ускоряет процесс проектирования баз знаний.

пользовательский интерфейс инструмента представляет собой главное окно, в котором пользователь может создавать вкладки. В каждой вкладке может происходить редактирование различных файлов исходных текстов баз знаний, представленных с помощью *SCg-кода*.

В рамках главного окна имеется панель инструментов и меню приложения.

Меню приложения представляет собой некоторый набор команд. Команды, которые отображаются в меню делятся на два типа:
- команды, которые являются общими для всех вкладок. В частности к ним относятся команды сохранения, загрузки, помощи и так далее;
- команды, которые специфичны для активной вкладки. Зависят от типа активной вкладки.

На панель инструментов, как и в пользовательских интерфейсах большинства приложений, вынесены наиболее часто используемые команды:
- Создать новый файл;
- Открыть файл;
- Сохранить;
- Сохранить как;
- Закрыть.

Основная идея, которая преследуется в данном редакторе SCg-кода — это упрощение и ускорение процесса редактирования sc.g-текстов.

В процессе редактирования пользователю доступны различные режимы редактирования.

Всего выделено 4 режима:
- ***Режим выделения и создания узлов***. В данном режиме пользователь может работать со всеми объектами выделяя и перемещая их, вызывая контекстное меню с командами. Отличительной особенностью данного режима является то, что в нем можно создавать sc.g-узлы;
- ***Режим создания sc.g-дуг***. Создание sc.g-дуги начинается с того, что пользователь указывает объект из которого она будет выходить, далее он может указать точки излома дуги, завершается создание указанием конечного объекта. В процессе создания пользователь может отменять последнее действие (указание начального объекта, точки излома);
- ***Режим создания sc.g-шин***. sc.g-шины используются для увеличения контактной площади узла, поэтому они могут создаваться лишь для sc.g-узлов. Создание шины начинается с указания sc.g-узла, далее как и при создании sc.g-дуг указываются точки излома. Как и при создании дуг пользователь может отменять последнее действие нажатием правой клавиши мыши;
- ***Режим создания sc.g-контуров***. Создание sc.g-контура начинается с указаний первой его точки. Далее, как и в случае с sc.g-дугами и sc.g-шинами, указываются точки. Стоит отметить, что все объекты, которые попадут внутрь созданного контура, будут добавлены в него автоматически. Как и при создании дуг и шин пользователь может отменять последнее действие.

Кроме перечисленных выше команд существует еще целый ряд команд редактирования:
- Команда изменения основного текстового идентификатора элемента;
- Команда изменения типа элемента;

- Команда установки содержимого.

Полученные файлы исходных текстов в дальнейшем могут быть погружены в *базу знаний* ostis-системы с помощью *Реализации транслятора файлов исходных текстов базы знаний в sc-память ostis-платформы*.

Реализация транслятора файлов исходных текстов базы знаний в sc-память ostis-платформы
:= [sc-builder]
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
⇒ *используемый язык**:
SCs-код
⇒ *зависимости компонента**:
{• *Библиотека методов и структура данных C++ Standard Library*
}
⇐ *программный компонент**:
Программный вариант реализации ostis-платформы

Реализация транслятора файлов исходных текстов базы знаний в sc-память ostis-платформы позволяет осуществить сборку *базы знаний* из набора файлов исходных текстов, записанных в SCs-коде с ограничениями в бинарный формат, воспринимаемый *Программной моделью sc-памяти* (см. *Пункт 6.3.4.4. Общее описание процесса и Пример трансляции sc-текста в sc-память ostis-платформы*). При этом возможна как сборка "с нуля" (с уничтожением ранее созданного слепка памяти), так и аддитивная сборка, когда информация, содержащаяся в заданном множестве файлов, добавляется к уже имеющемуся слепку состояния памяти. В текущей реализации сборщик осуществляет "склеивание" ("слияние") sc-элементов, имеющих на уровне файлов исходных текстов одинаковые *системные sc-идентификаторы*.

Кроме *KBE* существует редактор текстов базы знаний, являющийся частью *Реализации интерпретатора sc-моделей пользовательских интерфейсов*, обладающий схожим с *KBE* функционалом, но при этом позволяющий редактировать базу знаний в режиме реального времени и без создания файлов исходных текстов базы знаний, именно им рекомендуется пользоваться для редактирования базы знаний.

§ 5.2.3. Коллективный аспект проектирования и разработки баз знаний ostis-систем

Для решения задачи *коллективной разработки баз знаний* разработана модель деятельности, направленной на создание *гибридных баз знаний* коллективом разработчиков.

Данная модель базируется на модели деятельности различных субъектов и реализована в виде онтологии предметной области деятельности разработчиков, направленной на разработку и модификацию гибридных баз знаний.

Процесс создания и редактирования *базы знаний ostis-системы* сводится к формированию разработчиками предложений по редактированию того или иного раздела *базы знаний* и последующему рассмотрению этих предложений администраторами *базы знаний*.

Кроме того, предполагается, что в случае необходимости для верификации поступающих предложений по редактированию базы знаний могут привлекаться эксперты, а управление процессом разработки осуществляется менеджерами соответствующих проектов по разработке базы знаний. При этом формирование проектных заданий и их спецификация осуществляются также при помощи механизма предложений по редактированию соответствующего раздела базы знаний.

Таким образом, вся информация, связанная с текущими процессами разработки базы знаний, историей и планами ее развития, хранится в той же базе знаний, что и ее предметная часть, то есть часть базы знаний, доступная конечному пользователю системы. Такой подход обеспечивает широкие возможности автоматизации процесса создания баз знаний, а также последующего анализа и совершенствования базы знаний.

Каждое предложение по редактированию базы знаний представляет собой структуру, содержащую sc-текст, который предлагается включить в состав согласованной части базы знаний. В состав таких предложений могут входить знаки действий по редактированию базы знаний, которые автоматически инициируются и выполняются соответствующими агентами после утверждения предложения.

§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы

Важным этапом в разработке любой системы является контроль ее качества, так как именно на этом этапе определяется степень жизнеспособности и эффективности системы.

Верификация является видом анализа качества и частью процесса разработки системы. Она заключается в проверке информации на правильность и полноту. Ее целью является выявление ошибок, различных дефектов и недоработок для своевременного их устранения.

Существующие на данный момент методы верификации хорошо развиты и разработано большое количество различных моделей верификации, использующих расширенные таблицы решения, сети Петри, различные логики, например, логики с векторной семантикой (см. *Аршинский Л.В..КомплВПБЗсИ-2020ст*) и другие модели. Более того формируются специализированные онтологии для описания многообразия средств и моделей верификации баз знаний (см. *Rybina G..Metho aAfVoKB-2007art*). Однако механизма взаимодействия средств, использующих данные методы, нет.

Поэтому средства верификации баз знаний, существующие на данный момент, обладают рядом таких проблем как (см. *Zhang D.KnowlBVIaA-2023art*):
- зависимость от формата представления информации, из-за чего приходится тратить дополнительно время на конвертирование информации;
- проблема невозможности быть переиспользованными, так как средства обычно делаются с учетом особенностей конкретной системы;
- проблема отсутствия механизма взаимодействия средств верификации и анализа знаний;
- высокая роль человека в процессе верификации, так как самым распространенным методом верификации баз знаний является ручная проверка базы экспертом, человек выступает как администратор, принимающий единогласное решение, навязывая свое мнение системе;
- современные средства не учитывают и не рассматривают процесс верификации в рамках взаимодействия систем друг с другом.

Эти проблемы могли бы быть решены, если:
- использовать унифицированный и удобный формат представления знаний;
- системы создавались бы по общей методологии и были бы совместимы друг с другом;
- продумать и реализовать механизм позволяющий системе стремиться самой принимать решение относительно своего состояния и наличия в нем проблемных моментов и ошибок, система может допускать ошибки и не всегда принимать верные решения, но это должны быть ее ошибки, а не экспертов и разработчиков.

Преимуществами *Технологии OSTIS* в рамках задачи верификации являются:
- наличие общей методологии проектирования интеллектуальных систем, позволяющая решить проблему совместимости систем при их коллективном взаимодействии;
- все знания представлены в унифицированном виде, что позволяет эффективно их обрабатывать, сводя затраты на конвертирование к минимуму;
- средства, с помощью которых производится выявление, анализ и устранение противоречий, описаны в самой базе знаний, а также их спецификация представлены в самой базе знаний системы, тем самым обеспечивая легкость их расширения и позволяя системе знать, каким инструментарием она обладает;
- отсутствие семантических эквивалентных фрагментов, что обеспечивает локальность вносимых исправлений и исключает необходимость вносить исправления многократно в разных местах;
- многоагентный подход, который позволяет рассматривать средства анализа и верификации баз знаний как коллектив агентов, способных взаимодействовать друг с другом и в дальнейшем принимать общее решение касательно состояния базы знаний.

Предлагаемый подход сводится к разработке:
- специализированной *предметной области и онтологии*, которая бы содержала в себе все необходимые знания о возможных видах проблемных фрагментов базы знаний и методах их исправления;
- алгоритма, позволяющего системе выявить в себе проблемные фрагменты и устранить их, при этом обеспечив согласованность работы средств самой системы;
- *специализированного решателя задач*, содержащего необходимые агенты для выявления и устранения проблемных фрагментов.

Качество базы знаний во многом определяется уровнем наличия/отсутствия *не-факторов* (см. *Нариньяни А.С.нФактоКВ-2004ст*) в *базе знаний*.

не-фактор

:= [группа семантических свойств, определяющих качество информации, хранимой в памяти кибернетической системы]

= {• *корректность/некорректность информации, хранимой в памяти кибернетической системы*
• *однозначность/неоднозначность информации, хранимой в памяти кибернетической системы*
• *целостность/нецелостность информации, хранимой в памяти кибернетической системы*
• *чистота/загрязненность информации, хранимой в памяти кибернетической системы*
• *достоверность/недостоверность информации, хранимой в памяти кибернетической системы*
• *точность/неточность информации, хранимой в памяти кибернетической системы*
• *четкость/нечеткость информации, хранимой в памяти кибернетической системы*
• *определенность/недоопределенность информации, хранимой в памяти кибернетической системы*
}

проблемная структура
:= [структура, описывающая проблемный фрагмент базы знаний]
:= [структура, описывающая некачественный фрагмент базы знаний]
⇐ *объединение**:
{• *некорректная структура*
:= [структура, содержащая фрагменты, противоречащие каким-либо правилам или закономерностям описанным в базе знаний]
• *структура, описывающая неполноту в базе знаний*
:= [структура, в которой имеется неполнота (то есть имеется некоторое количество информационных дыр)]
⇒ *примечание**:
[Под структурой, описывающей неполноту в базе знаний, понимается структура, содержащая фрагмент базы знаний, в котором отсутствует какая-либо информация, которая необходима (или, по крайней мере, желательна) для однозначного и полного понимания смысла данного фрагмента.]
• *информационный мусор*
:= [структура, удаление которой существенно не усложнит деятельность системы]
:= [структура, содержащая фрагмент базы знаний, который по каким-либо причинам стал ненужным и требует удаления]
}

противоречие*
:= [пара противоречащих друг другу фрагментов информации, хранимой в памяти кибернетической системы*]
⇒ *примечание**:
[Чаще всего противоречащими друг другу информационными фрагментами являются:

- явно представленная в памяти некоторая закономерность (некоторое правило);
- информационный фрагмент, не соответствующий (противоречащий) указанной закономерности.

В этом случае некорректность может присутствовать:

- либо в информационном фрагменте, который противоречит указанной закономерности;
- либо в самой этой закономерности;
- либо и там и там.

]

некорректная структура
⇐ *включение**:
{• *дублирование системных идентификаторов*
• *несоответствие элементов связки доменам отношения*
• *цикл по отношению порядка*
• *структура, противоречащая свойству единственности*
}

структура, описывающая неполноту в базе знаний
⇐ *включение**:
{• *не указан максимальный класс объектов исследования предметной области*
• *для сущности указан системный, но не указаны основные идентификаторы для всех внешних языков*
• *не указаны домены отношения*
• *понятие не соотнесено ни с одной предметной областью*
}

структура, требующее внимание разработчика
:= [проблемная структура, для исправления которой требуется участие разработчика]

множество элементов, которые должны быть удалены для исправления структуры*
:= [множество элементов, удаление которых из структуры позволяет устранить в ней противоречия]

множество элементов, которые должны быть добавлены для исправления структуры*
:= [множество элементов, добавление которых в структуру позволяет устранить в ней противоречия]

структура, которую система не способна исправить сама
:= [структура, в которой система не способна автоматически устранить противоречия]

следует отличать*
∋ {• *структура, которую система не может решить сама*
⇒ *примечание**:
[Здесь структура, которую система не может решить сама, не может быть исправлена при взаимодействии с разработчиком и требует полного исправления от самого разработчика]
• *требующее внимание разработчика*
⇒ *примечание**:
[Здесь структура, требующая внимания разработчика, может быть решена в процессе верификации, но потребуется участие разработчика]
}

Решатель задач средств выявления и устранения противоречий
⇒ *декомпозиция абстрактного sc-агента**:
{• *Неатомарный абстрактный sc-агент выявления противоречий*
:= [Множество агентов, обеспечивающих поиск и фиксирование противоречий в структуре]
• *Неатомарный абстрактный sc-агент устранения противоречий*
:= [Множество агентов, создающих предложения по исправлению противоречий]
⇒ *примечание**:
[Результатом работы таких агентов будут множества предлагаемых к удалению из структуры или добавлению в структуру элементов]
• *Абстрактный sc-агент объединения структур*
:= [Агент, создающий структуру содержащую все элементы сливаемых структур]
• *Абстрактный sc-агент применения предложений по устранению противоречий*
• *Абстрактный sc-агент внесения исправлений в базу знаний*
⇒ *примечание**:
[Внесение изменений подразумевает не только исправление в базе знаний изначальной проблемной структуры, но и фиксацию самого факта изменения состояния базы знаний]
• *Неатомарный абстрактный sc-агент верификации структуры*
⇒ *примечание**:
[Агент, обеспечивающий полный цикл верификации структуры и координирующий другие агенты]
}

§ 5.2.5. Логико-семантическая модель ostis-системы автоматизации управления взаимодействием разработчиков различных категорий в процессе проектирования базы знаний ostis-системы

В первую очередь все пользователи любой *ostis-системы* делятся на *зарегистрированных пользователей* и *незарегистрированных пользователей*.

пользователь базы знаний ostis-системы*
:= [бинарное отношение, связывающее sc-модель базы знаний ostis-системы и sc-элемент, обозначающий персону, участвующую в разработке или эксплуатации этой базы знаний]
∈ *бинарное отношение*
∈ *ориентированное отношение*
⇒ *разбиение**:
Типология отношений между базами знаний ostis-систем и их пользователями по наличию факта прохождения регистрации в этих ostis-системах^

= {• *зарегистрированный пользователь**
• *незарегистрированный пользователь**
}

Данное отношение отражает связь *пользователя* и *базы знаний* в целом, при этом тот же сам пользователь может быть связан другими более частными отношениями с какими-либо фрагментами этой же *базы знаний*.

зарегистрированный пользователь имеет доступ на чтение всей базы знаний и внесение предложений ко всей базе знаний, может выполнять роль конечного пользователя ostis-системы, то есть работать в режиме эксплуатации, а также роль ее разработчика.

При этом независимо от роли, которую выполняет тот или иной *пользователь*, он может делать предложения по редактированию любой из частей базы знаний, которые в зависимости от его уровня будут либо приняты автоматически, либо будут отдельно рассматриваться.

пользователь, обладающий правом просмотра sc-структуры базы знаний ostis-систем*
:= [бинарное отношение, связывающее sc-элемент, обозначающий sc-структуру (например, фрагмент sc-модели базы знаний), и sc-элемент, обозначающий пользователя этой ostis-системы, который обладает правом просмотра этой sc-структуры.]
∈ *бинарное отношение*
∈ *ориентированное отношение*

пользователь, обладающий правом просмотра sc-структуры базы знаний ostis-системы* может быть зарегистрирован или не зарегистрирован в *sc-модели базы знаний*.

пользователь, обладающий правом редактирования sc-структуры базы знаний ostis-систем*
:= [бинарное отношение, связывающее sc-элемент, обозначающий sc-структуру (например, фрагмент sc-модели базы знаний), и sc-элемент, обозначающий зарегистрированного пользователя ostis-системы, который обладает правом редактирования этой sc-структуры.]
∈ *бинарное отношение*
∈ *ориентированное отношение*
⊃ *пользователь, обладающий правом просмотра sc-структуры**
⇒ *покрытие**:
{• *пользователь, обладающий правом редактирования sc-структуры посредством формирования предложений по внесению изменений в согласованную часть базы знаний этой ostis-системы**
• *пользователь, обладающий правом редактирования sc-структуры с автоматическим формированием и принятием предложений по внесению изменений в согласованную часть базы знаний этой ostis-системы**
}

Связки отношения *пользователя, обладающего правом редактирования sc-структуры ostis-системы** связывают sc-структуру (не обязательно всю sc-модель базы знаний) и пользователя, зарегистрированного в этой sc-модели базы знаний.

разработчик*
⊂ *пользователь, обладающий правом редактирования sc-структуры**
:= [бинарное отношение, связывающее sc-элемент, обозначающий некоторый раздел базы знаний (в пределе — всю базы знаний), и sc-элемент, обозначающий пользователя ostis-системы, который может быть разработчиком данного раздела базы знаний, то есть выполнять проектные задачи в рамках данного раздела]
⇒ *разбиение**:
Типология разработчиков баз знаний ostis-систем^
= {• *администратор**
• *менеджер**
• *эксперт**
}

администратор*
:= [бинарное отношение, связывающее sc-элемент, обозначающий некоторый раздел базы знаний (в пределе — всю базы знаний), и sc-элемент, обозначающий пользователя ostis-системы, который является администратором данного раздела базы знаний]
⇒ *функции**:
{• [контроль целостности и непротиворечивости всей базы знаний]
• [определение уровней доступа других пользователей]

- [принятие решения относительно принятия или отклонения предложений в различные части базы знаний, в том числе при необходимости отправка их на экспертизу]
- [самостоятельное внесение изменений в различные части базы знаний путем использования соответствующих команд редактирования (при этом изменения автоматически оформляются как предложения и заносятся в раздел истории развития ostis-системы)]

}

менеджер*

:= [бинарное отношение, связывающее sc-элемент, обозначающий некоторый раздел базы знаний (в пределе — всю базы знаний), и sc-элемент, обозначающий персону, которая является менеджером данного раздела базы знаний]

⇒ *функции**:

{• [планирование объемов работ по разработке базы знаний]
- [детализация проектных задач на подзадачи, непосредственно формулирование проектных задач, назначение исполнителей проектных задач]
- [установка приоритетов и сроков выполнения задач]
- [контроль сроков выполнения проектных задач]

}

эксперт*

:= [бинарное отношение, связывающее sc-элемент, обозначающий какой-либо проект по разработке раздела базы знаний ostis-системы (в общем случае — всей базы знаний), и sc-элемент, обозначающий персону, которая является экспертом данного раздела базы знаний]

⇒ *функции**:

{• [верификация результатов выполнения проектных задач]
- [при необходимости эксперт может оставлять комментарии к любому фрагменту базы знаний относительно его корректности. Все комментарии попадают в раздел, описывающий план развития компьютерной системы]

}

При необходимости разработки объемной *базы знаний* может вводиться иерархия разработчиков, соответствующая иерархии разделов разрабатываемой *базы знаний*.

В этом случае утверждение какого-либо предложения администратором раздела нижнего уровня не приводит к интеграции предложения в соответствующий раздел, а требует рассмотрения администраторами более высокого уровня. Окончательное решение принимается администратором всей базы знаний.

Кроме того, любой участник процесса разработки имеет возможность оставить естественно-языковой комментарий к любому фрагменту или элементу базы знаний, таким образом, может осуществляться обсуждение каких-либо вопросов, связанных с указанным фрагментом или элементом базы знаний.

Такого рода комментарии попадают в раздел базы знаний текущие процессы развития компьютерной системы.

§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем

⇒ *ключевое понятие**:

- *компонентное проектирование баз знаний интеллектуальных систем*

⇒ *библиографическая ссылка**:

- *Ивашенко В.П.СеманТКПБЗ-2011ст*
- *Голенков В.В..ОткрыПНнСТ-2013ст*
- *Давыденко И.Т.ТехноКПБЗнО-2013ст*

Для широкого применения интеллектуальных систем, способных повысить качество решения прикладных задач, разработано большое число *баз знаний* по самым различным предметным областям. Однако в большинстве случаев каждая база знаний разрабатывается отдельно и независимо от других, в отсутствие единой унифицированной формальной основы для представления знаний, а также единых принципов формирования систем понятий для описываемой предметной области. В связи с этим разработанные базы оказываются, как правило, несовместимы между собой и не пригодны для повторного использования. Для быстрой разработки достаточного количества баз знаний, кроме наличия средств разработки интеллектуальных систем, обеспечивающих разработку и проектирование различных компонентов интеллектуальной системы, включая базу знаний, требуется наличие соответствующей отлаженной технологии проектирования баз знаний (*см. Ивашенко В.П.СеманТКПБЗ-2011ст*).

Компонентный подход к разработке интеллектуальных компьютерных систем, реализуемый в виде ***библиотеки многократно используемых компонентов ostis-систем***, позволяет решить описанные проблемы (см. *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*). *Библиотека многократно используемых компонентов баз знаний ostis-систем в составе Метасистемы OSTIS* является важнейшим фрагментом *Метасистемы OSTIS*, который обеспечивает надежность и совместимость проектируемых фрагментов баз знаний, а также повышение скорости разработки *баз знаний интеллектуальных компьютерных систем*.

Описываемая библиотека включает в себя множество компонентов баз знаний и их спецификаций.

Рассмотрим классификацию многократно используемых компонентов баз знаний ostis-систем
(*см. Давыденко И.Т.ТехноКПБЗнО-2013ст*).

многократно используемый компонент баз знаний ostis-систем
- ⊃ *предметная область и онтология*
  - ⊂ *раздел базы знаний*
- ⊃ *семантическая окрестность*
  - ⊃ *семантическая окрестность по инцидентным коннекторам*
  - ⊃ *полная семантическая окрестность*
  - ⊃ *базовая семантическая окрестность*
  - ⊃ *специализированная семантическая окрестность*
- ⊃ *базовые фрагменты предметных областей и онтологий*
  - ⇒ *примечание**:
    [Базовый фрагмент предметной области и онтологии включает в себя теоретико-множественную, логическую онтологии, а также терминологические фрагменты.**]**
  - ⇒ *примечание**:
    [Данный вид многократно используемых компонентов позволяет использовать только те знания, которые непосредственно необходимы для функционирования интеллектуальных систем, исключив то, что никак не влияет на работу конечной системы (пояснения, примеры, дидактический материал и так далее).**]**
  - ∋ *Базовый фрагмент теории логических формул, высказываний и логических sc-языков*
  - ∋ *Базовый фрагмент теории множеств*
  - ∋ *Базовый фрагмент теории связок и отношений*
- ⊃ *база знаний*
  - ⇒ *примечание**:
    [Целые базы знаний могут быть многократно используемыми компонентами в случае разработки интеллектуальных систем, назначение которых совпадает.**]**

Важнейшими компонентами, которые входят в состав библиотеки, являются *онтологии предметных областей*. *онтологии предметных областей*, описывающие виды знаний, которые являются основой для построения *базы знаний* любой *интеллектуальной системы*, входят в *Ядро базы знаний*, поскольку являются *онтологиями верхнего уровня* (см. *§ 2.5.6. Онтологии верхнего уровня*). Следовательно, *Ядро базы знаний* представляет собой компонент, входящий в состав каждой базы знаний, разрабатываемой по *Технологии OSTIS*, и устанавливающийся в первую очередь.

многократно используемый компонент баз знаний ostis-систем
- ∋ *Ядро базы знаний*
  - ∋ *Предметная область и онтология множеств*
  - ∋ *Предметная область и онтология связок и отношений*
  - ∋ *Предметная область и онтология структур*
  - ∋ *Предметная область и онтология семантических окрестностей*
  - ∋ *Предметная область и онтология предметных областей*
  - ∋ *Предметная область и онтология онтологий*

онтологии предметных областей, которые используются в большинстве интеллектуальных систем, являются частью *Расширенного ядра базы знаний*.

многократно используемый компонент баз знаний ostis-систем
- ∋ *Расширенное ядро базы знаний*
  - ⇒ *включение**:
    Ядро базы знаний
  - ⇒ *пояснение**:
    [В отличие от *Ядра базы знаний Расширенное ядро базы знаний* содержит в себе не только обязательные для установки *онтологии предметных областей*, но и такие *онтологии предметных областей*, кото-

рые используются в большинстве *интеллектуальных компьютерных систем*. Следовательно, являются компонентами, которые наиболее часто устанавливаются пользователями *Библиотеки многократно используемых компонентов баз знаний ostis-систем в составе Метасистемы OSTIS*.]

- ∋ *Предметная область и онтология параметров, величин и шкал*
- ∋ *Предметная область и онтология чисел и числовых структур*
- ∋ *Предметная область и онтология темпоральных сущностей*
- ∋ *Предметная область и онтология пространственных сущностей различных форм*
- ∋ *Предметная область и онтология материальных сущностей*

Подробное описание представленных *многократно используемых компонентов баз знаний* можно найти в *Главе 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*, а также в *Главе 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*.

Представленный список *многократно используемых компонентов баз знаний* не является окончательным. В случае, когда разработчик *базы знаний* интеллектуальной системы считает, что разработанный им компонент сможет стать неотъемлемой частью библиотеки, то компонент будет добавлен в библиотеку, как многократно используемый, если:

- компонент специфицирован;
- компонент прошел верификацию и соответствует требованиям разработчиков библиотеки (см. *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*).

Чтобы *многократно используемый компонент баз знаний* мог быть принят в библиотеку, он должен быть корректно специфицирован. Для этого используются отношения класса *необходимое для установки отношение, специфицирующее многократно используемый компонент ostis-систем*, а также *необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем*. В *§ 5.1.3. Понятие многократно используемого компонента ostis-систем* описана спецификация, общая для любых типов компонентов. Однако в зависимости от типа компонента, спецификация может расширяться. Рассмотрим *необязательное для установки отношение, специфицирующее многократно используемый компонент баз знаний ostis-систем*, его поиска и установки в *дочернюю ostis-систему*, если таковым компонентом является *предметная область и онтология*.

необязательное для установки отношение, специфицирующее многократно используемый компонент баз знаний ostis-систем

- ⊂ *необязательное для установки отношение, специфицирующее многократно используемый компонент ostis-систем*
- ∋ *максимальный класс объектов исследования'*
  - ≔ [класс объектов исследования, для которого в заданной предметной области отсутствует другой класс объектов исследования, который был бы его надмножеством']
  - ⇒ *первый домен**:
    предметная область и онтология
  - ⇒ *второй домен**:
    понятие
- ∋ *немаксимальный класс объектов исследования'*
  - ⇒ *первый домен**:
    предметная область и онтология
  - ⇒ *второй домен**:
    понятие
- ∋ *исследуемое отношение'*
  - ⇒ *первый домен**:
    предметная область и онтология
  - ⇒ *второй домен**:
    понятие

Для компонентов, которые являются частью *Библиотеки многократно используемых компонентов баз знаний ostis-систем в составе Метасистемы OSTIS*, также существуют средства поиска, обновления (см. *§ 5.1.4. Менеджер многократно используемых компонентов ostis-систем*).

Корректно спроектированные спецификации компонентов позволят построить полную иерархию зависимостей компонентов, а также их структуру, что в свою очередь позволит беспрепятственное использование компонентов и их фрагментов в рамках компонентного проектирования баз знаний.

Заключение к Главе 5.2.

Проектирование и анализ качества *баз знаний* являются важнейшими этапами разработки *интеллектуальных компьютерных систем*, так как они во многом определяют качество всей интеллектуальной системы.

Предложенная методология коллективной разработки базы знаний на основе *Технологии OSTIS*, которая включает в себя модель верификации и контроля качества *базы знаний*, а также *компонентный подход* к проектированию *баз знаний*, позволяет повысить эффективность проектирования *интеллектуальных компьютерных систем* и средств автоматизации разработки таких систем.

Глава 5.3.

Методика и средства компонентного проектирования решателей задач ostis-систем

⇒ *автор**:
- *Шункевич Д. В.*
- *Марковец В. С.*

⇒ *аннотация**:
[В главе рассматривается методика построения и модификации *решателей задач ostis-систем*, модель системы автоматизации проектирования *решателей задач*, а также принципы организации *библиотеки многократно используемых компонентов решателей задач ostis-систем*.]

⇒ *подраздел**:
- *§ 5.3.1. Действия и методики проектирования решателей задач ostis-систем*
- *§ 5.3.2. Логико-семантическая модель комплекса ostis-систем автоматизации проектирования решателей задач ostis-систем*
- *§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем*

⇒ *ключевой знак**:
- *Методика построения и модификации машины обработки знаний ostis-систем*
- *Система автоматизации проектирования решателей задач ostis-систем*

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *точка останова*
- *некорректность в scp-программе*
- *ошибка в scp-программе*
- *многократно используемый компонент решателей задач*

⇒ *библиографическая ссылка**:
- *Шункевич Д.В.Модели иСКПМ-2013ст*
- *Шункевич Д.В..МетодКПСУЗ-2013ст*
- *Голенков В.В..СеманТКПС-2015ст*
- *Шункевич Д.В..СредстваПКПС-2015ст*
- *Shunkevich.D.V.AgentMMaToC-2018art*
- *Борисов А.Н.ПострИСОнЗсП-2014ст*
- *Грибова В.В..БазовТРИС-2015ст*
- *Заливако С.С..СеманТКПИРЗ-2012ст*

Введение в Главу 5.3.

В области разработки *решателей задач* существует большое количество конкретных реализаций, однако вопросы совместимости различных решателей задачи и их компонентов практически не рассматриваются. Гипотетически возможно существование универсального решателя задач, объединяющего в себе все известные способы и методы решения задач. Однако использование такого решателя в прикладных целях не является целесообразным. Таким образом, наиболее приемлемым вариантом становится создание библиотеки совместимых между собой компонентов (см. *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*), из которых впоследствии может быть собран решатель, удовлетворяющий необходимым требованиям.

Как показано в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*, *решатель задач ostis-системы* представляет собой иерархическую систему *навыков*, которыми владеет ostis-система на текущий момент. В свою очередь, *навык* представляет собой некоторый *метод* решения задач заданного класса и соответствующую ему систему *sc-агентов* (*машину обработки знаний* частного вида), обеспечивающих интерпретацию данного метода. В связи с этим методика проектирования решателей задач ostis-систем фактически сводится к методике проектирования баз знаний ostis-систем (см. *Главу 5.2. Методика и средства проектирования и анали-*

за качества баз знаний ostis-систем) и методике проектирования машин обработки знаний ostis-систем, которая детально рассматривается в данной главе.

Вопросам разработки *решателей задач ostis-систем* посвящен ряд работ, таких как *Шункевич Д.В.Модели иСКПМ-2013ст* , *Шункевич Д.В..МетодКПСУЗ-2013ст* , *Шункевич Д.В..СредстваПКПС-2015ст* , *Голенков В.В..СеманТКПС-2015ст* , *Shunkevich.D.V.AgentMMaToC-2018art* . Среди других известных работ, посвященных вопросам компонентного проектирования решателей задач в целом, стоит отметить работы *Борисов А.Н.ПострИСОнЗсП-2014ст*, *Грибова В.В..БазовТРИС-2015ст*, в которых, однако, не уделяется внимание разработке комплексной методики проектирования решателей задач, которой посвящена данная глава.

§ 5.3.1. Действия и методики проектирования решателей задач ostis-систем

⇒ *ключевой знак**:
- *Методика построения и модификации машины обработки знаний ostis-систем*

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *sc-агент*
- *scp-программа*

Основу любого решателя задач составляет система sc-агентов, то есть *машина обработки знаний ostis-системы*, в связи с этим целесообразно рассмотреть более детально процесс проектирования *машин обработки знаний ostis-систем*. Приведенная методика разработана с учетом того, что каждая *машина обработки знаний ostis-систем* представляет собой иерархический коллектив sc-агентов.

Построение же общей методики проектирования решателей задач ostis-систем требует также уточнения принципов разработки *методов* решения задач, которые, в свою очередь, тесно пересекаются с общими принципами разработки баз знаний ostis-систем (см. *Главу 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*). Рассмотрим более подробно *Методику построения и модификации машин обработки знаний ostis-систем*.

Методика построения и модификации машин обработки знаний ostis-систем включает несколько этапов:
- формирование требований к машине обработки знаний и ее формальная спецификация;
- формирование коллектива sc-агентов, входящих в состав разрабатываемой машины;
- разработка алгоритмов атомарных sc-агентов;
- реализация программ sc-агентов;
- верификация разработанных компонентов;
- отладка разработанных компонентов, исправление ошибок.

Предлагаемая методика может быть применена как при разработке *машины обработки знаний* объединенных *решателей задач*, так и при разработке *машины обработки знаний решателей задач* частного вида, поскольку с формальной точки зрения все такие машины трактуются как *неатомарные абстрактные sc-агенты*.

Этап 1. Формирование требований к машине обработки знаний и ее формальная спецификация

На данном этапе необходимо:
- четко выделить задачи, решение которых должен обеспечивать *решатель задач*;
- продумать предполагаемые способы их решения и на основе данного анализа определить место будущей машины обработки знаний *решателя задач* в общей иерархии *решателей задач*.

Важность данного этапа заключается в том, что при правильной классификации существует вероятность того, что в составе *библиотеки многократно используемых компонентов решателей задач ostis-систем* уже есть реализованный вариант требуемой машины обработки знаний. В противном случае, у разработчика появляется возможность включить разработанную машину в *библиотеку многократно используемых компонентов решателей задач ostis-систем* для последующего использования. Данные факты обусловлены тем, что структура *библиотеки многократно используемых компонентов решателей задач ostis-систем* основана на семантической классификации *решателей задач* и, соответственно, их компонентов.

При недостаточно четкой спецификации и классификации разрабатываемого *решателя задач* повышается вероятность того, что подходящая машина обработки знаний не будет найдена в библиотеке компонентов даже в случае, если она там есть, а вновь разработанный *решатель задач* не сможет быть включен в библиотеку. Таким образом, принцип многократного использования уже разработанных компонентов будет нарушен, что существенно повысит затраты на разработку.

Этап 2. Формирование коллектива sc-агентов, входящих в состав разрабатываемой машины обработки знаний

В случае, когда найти в библиотеке готовую машину обработки знаний, удовлетворяющую всем предъявляемым требованиям, не удалось, необходимо выделить (то есть собственно определить множество требуемых компонентов) и специфицировать все ее компоненты.

Результатом данного этапа является перечень полностью специфицированных *абстрактных sc-агентов*, которые войдут в состав разрабатываемой машины обработки знаний, с их иерархией вплоть до *атомарных абстрактных sc-агентов*. В рамках данного этапа очень важно проектировать коллектив агентов таким образом, чтобы максимально задействовать уже имеющиеся в библиотеке многократно используемые компоненты ostis-систем, а при отсутствии нужного компонента — иметь возможность включить его в библиотеку после реализации.

При разработке перечня sc-агентов (в том числе их спецификаций) необходимо соблюдать ряд принципов:

- каждый разрабатываемый sc-агент должен быть, по возможности, предметно независим, то есть во множество ключевых узлов данного sc-агента не должны входить понятия, имеющие отношение непосредственно к рассматриваемой предметной области. Исключение составляют понятия из общих предметных областей, которые носят междисциплинарный характер (например, отношение *включение** или понятие *действие*, см. *Главу 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*). Данное правило также может быть нарушено в случае, если sc-агент является вспомогательным и ориентирован на обработку какого-либо конкретного класса объектов (например, sc-агенты, выполняющие арифметические вычисления, могут напрямую работать с конкретными отношениями *сложение** и *умножение** и тому подобное). Всю необходимую для решения задачи информацию sc-агент должен извлекать из семантической окрестности соответствующего инициированного действия. Разработанный с учетом указанных требований sc-агент может быть использован при проектировании большего числа ostis-систем, чем в случае, если бы он был реализован с ориентацией на конкретную предметную область. После завершения разработки и отладки такой sc-агент должен быть включен в *библиотеку многократно используемых абстрактных sc-агентов*, которая входит в состав *библиотеки многократно используемых компонентов решателей задач ostis-систем*;
- не стоит путать понятия sc-агент и *агентная программа* (в том числе *агентная scp-программа*). Взаимодействие sc-агентов осуществляется исключительно посредством спецификации информационных процессов в общей памяти, каждый sc-агент реагирует на некоторый класс событий в sc-памяти. Таким образом, каждому sc-агенту соответствует некоторое условие инициирования и одна агентная программа, которая запускается автоматически при возникновении в sc-памяти соответствующего условия инициирования. При этом в рамках данной программы могут сколько угодно раз вызываться различные подпрограммы. Однако не стоит путать инициирование sc-агента, которое осуществляется при появлении в sc-памяти соответствующей конструкции, и вызов подпрограммы другой программой, который предполагает явное указание вызываемой подпрограммы и перечня ее параметров;
- каждый sc-агент должен самостоятельно проверять полноту соответствия своего условия инициирования текущему состоянию sc-памяти. В процессе решения какой-либо задачи может возникнуть ситуация, когда на появление одной и той же структуры среагировали несколько sc-агентов. В таком случае выполнение продолжают только те из них, условие инициирования которых полностью соответствует сложившейся ситуации. Остальные sc-агенты в данном случае прекращают выполнение и возвращаются в режим ожидания. Выполнение данного принципа достигается за счет тщательного уточнения спецификаций разрабатываемых sc-агентов. В общем случае условия инициирования у нескольких sc-агентов могут совпадать, например, в случае, когда одна и та же задача может быть решена разными способами и заранее неизвестно, какой из них приведет к желаемому результату;
- необходимо помнить, что неатомарный sc-агент с точки зрения других sc-агентов, не входящих в его состав, должен функционировать как целостный sc-агент (выполнять логически атомарные действия), что накладывает определенные требования на спецификации атомарных sc-агентов, входящих в его состав: как минимум, необходимо, чтобы в составе неатомарного sc-агента присутствовал хотя бы один атомарный sc-агент, условие инициирования которого полностью совпадает с условием инициирования данного неатомарного sc-агента;
- при необходимости реализации нового sc-агента следует руководствоваться следующими принципами выделения атомарных абстрактных *sc-агентов*:
  - проектируемый sc-агент должен быть максимально независим от предметной области, что позволит в дальнейшем использовать его при разработке *решателей задач* максимально возможного числа ostis-систем. При этом универсальность предполагает не только минимизацию числа ключевых узлов sc-агента, но и выделение класса действий, выполняемых данным sc-агентом таким образом, чтобы имело смысл включить данный sc-агент в *библиотеку многократно используемых абстрактных sc-агентов* и использовать его при разработке *решателей задач* других ostis-систем. Не следует искусственно увязывать ряд действий в один sc-агент и, наоборот, расчленять одно самодостаточное действие на поддействия: это вызовет сложности восприятия принципов работы sc-агента разработчиками и не позволит использовать sc-агент в ряде систем (например, в обучающих системах, которые должны объяснять ход решения пользователю, см. *Главу 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*);
  - выполняемое данным sc-агентом действие должно быть логически целостным и завершенным. Следует помнить, что все sc-агенты взаимодействуют исключительно через общую sc-память, и избегать ситуаций, в которых инициирование одного sc-агента осуществляется путем явной генерации известного условия

инициирования другим *sc-агентом* (то есть, по сути, явным непосредственным вызовом одного sc-агента другим);
    - имеет смысл выделять в отдельные sc-агенты относительно крупные фрагменты реализации некоторого общего алгоритма, которые могут выполняться независимо друг от друга.
- при объединении sc-агентов в коллективы рекомендуется проектировать их таким образом, чтобы они могли быть использованы не только как часть рассматриваемого *неатомарного абстрактного sc-агента*. В случае, если это не представляется возможным и некоторые sc-агенты, будучи отделенными от коллектива, теряют *смысл*, необходимо указать данный факт при документировании рассматриваемых sc-агентов;
- фактическим инициатором запуска sc-агента посредством общей памяти (автором соответствующей конструкции) может быть как непосредственно пользователь системы, так и другой sc-агент, что никак не должно отражаться в работе самого sc-агента.

Этап 3. Разработка алгоритмов атомарных sc-агентов

В рамках данного этапа необходимо продумать алгоритм работы каждого разрабатываемого *атомарного sc-агента*. Разработка алгоритма подразумевает выделение в нем логически целостных фрагментов, которые могут быть реализованы как отдельные *scp-программы*, в том числе выполняемые параллельно. Таким образом, появляется необходимость говорить не только о *библиотеке многократно используемых абстрактных sc-агентов*, но и о *библиотеке многократно используемых программ обработки sc-текстов* на различных языках программирования, в том числе *библиотеке многократно используемых scp-программ*. Благодаря этому часть scp-программ, реализующих алгоритм работы некоторого sc-агента, может быть заимствована из соответствующей библиотеки.

Важно помнить, что если в процессе работы *sc-агент* генерирует в памяти какие-либо временные структуры, то при завершении работы он обязан удалять всю информацию, использование которой в системе более нецелесообразно (убрать за собой информационный мусор). Исключение составляют ситуации, когда подобная информация необходима нескольким *sc-агентам* для решения одной задачи, однако после решения задачи информация становится бесполезной или избыточной и требует удаления. В данном случае может возникнуть ситуация, когда ни один из *sc-агентов* не в состоянии удалить информационный мусор. В таком случае возникает необходимость говорить о включении в состав *решателя задач* специализированных *sc-агентов*, задачей которых является выявление и уничтожение информационного мусора.

Этап 4. Реализация программ агентов

Конечным этапом непосредственно разработки является реализация специфицированных ранее *scp-программ* или при необходимости программ, реализуемых на уровне *ostis-платформы*.

Этап 5. Верификация разработанных компонентов

Верификация разработанных компонентов может осуществляться как вручную, так и с использованием специфицированных средств, входящих в состав системы автоматизации проектирования *решателей задач ostis-систем*.

Этап 6. Отладка разработанных компонентов. Исправление ошибок

Этап отладки разработанных компонентов, в свою очередь, можно также условно разделить на более частные этапы:
- отладка отдельных scp-программ или программ, реализуемых на уровне платформы;
- отладка отдельных атомарных sc-агентов;
- отладка неатомарных sc-агентов, входящих в состав машины обработки знаний;
- отладка всей машины обработки знаний.

Этап 5 и **Этап 6** могут выполняться параллельно и повторяются до тех пор, пока разработанные компоненты не будут удовлетворять необходимым требованиям.

Методика построения и модификации машин обработки знаний ostis-систем основана на онтологии деятельности разработчиков машин обработки знаний.

Каждый *решатель задач ostis-системы* представляет собой совокупность навыков, а машина обработки знаний — совокупность интерпретаторов навыков, составляющих некоторый *решатель задач ostis-системы*, то есть его операционную семантику. Таким образом *машина обработки знаний* представляет собой *абстрактный sc-агент*, в связи с чем разработка машины сводится к разработке такого агента.

Рассмотрим фрагмент онтологии деятельности, направленной на построение и модификацию машин обработки знаний, записанный в *SCn-коде*:

действие. разработать машину обработки знаний ostis-системы
⊂ *действие. разработать абстрактный sc-агент*
⇒ *разбиение**:
    {• *действие. разработать атомарный абстрактный sc-агент*
        ⊃ *действие. разработать платформенно-независимый атомарный абстрактный sc-агент*

- *действие. разработать неатомарный абстрактный sc-агент*

}

⇒ *обобщенное поддействие**:
- *действие. специфицировать абстрактный sc-агент*
- *действие. найти в библиотеке абстрактный sc-агент, удовлетворяющий заданной спецификации*
- *действие. верифицировать sc-агент*
- *действие. отладить sc-агент*

действие. разработать платформенно-независимый атомарный абстрактный sc-агент

⇒ *обобщенное поддействие**:
- *действие. декомпозировать платформенно-независимый атомарный абстрактный sc-агент на scp-программы*
- *действие. разработать scp-программу*

действие. разработать неатомарный абстрактный sc-агент

⇒ *обобщенное поддействие**:
- *действие. декомпозировать неатомарный абстрактный sc-агент*
- *действие. разработать абстрактный sc-агент*

действие. разработать scp-программу

⇒ *обобщенное поддействие**:
- *действие. специфицировать scp-программу*
- *действие. найти в библиотеке scp-программу, удовлетворяющую заданной спецификации*
- *действие. реализовать специфицированную scp-программу*
- *действие. верифицировать scp-программу*
- *действие. отладить scp-программу*

действие. верифицировать sc-агент

⇒ *разбиение**:

{
- *действие. верифицировать атомарный sc-агент*
- *действие. верифицировать неатомарный sc-агент*

}

действие. отладить sc-агент

⇒ *разбиение**:

{
- *действие. отладить атомарный sc-агент*
- *действие. отладить неатомарный sc-агент*

}

Наличие такой онтологии позволяет:
- частично автоматизировать процесс построения и модификации машины обработки знаний при помощи соответствующей *Системы автоматизации проектирования решателей задач ostis-систем*;
- повысить эффективность информационной поддержки разработчиков, поскольку данная онтология включена в базу знаний *Метасистемы OSTIS*.

§ 5.3.2. Логико-семантическая модель комплекса ostis-систем автоматизации проектирования решателей задач ostis-систем

⇒ *ключевой знак**:
- *Система автоматизации проектирования решателей задач ostis-систем*
- *Решатель задач системы автоматизации проектирования агентов обработки знаний*
- *Решатель задач системы автоматизации проектирования scp-программ*

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *sc-агент*
- *база знаний*
- *scp-программа*
- *точка останова**
- *точка останова*

- *некорректность в scp-программе**
- *некорректность в scp-программе*
- *ошибка в scp-программе*
- *библиотека многократно используемых компонентов ostis-систем*
- *команда пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем*
- *команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний*
- *команда пользовательского интерфейса системы автоматизации проектирования scp-программ*

⇒ *подраздел**:
- *Пункт 5.3.2.1. Семантическая модель базы знаний системы автоматизации проектирования решателей задач ostis-систем*
- *Пункт 5.3.2.2. Семантическая модель решателя задач системы автоматизации проектирования решателей задач ostis-систем*
- *Пункт 5.3.2.3. Семантическая модель пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем*

К числу задач системы автоматизации проектирования *решателей задач ostis-систем* относится техническая поддержка разработчиков решателей, в том числе — обеспечение корректного и эффективного выполнения этапов, предусмотренных *Методикой проектирования машины обработки знаний ostis-систем*.

При разработке любых компонентов ostis-систем используются схожие принципы. Одним из основных принципов является принцип использования готовых компонентов различного рода, уже имеющихся в библиотеке многократно используемых компонентов ostis-систем, входящей в состав *Метасистемы OSTIS*.

Система автоматизации проектирования решателей задач ostis-систем сама по себе также является ostis-системой и имеет соответствующую структуру. Таким образом, модель данной системы включает *sc-модель базы знаний*, *sc-модель объединенного решателя задач* и *sc-модель пользовательского интерфейса*.

В рамках данной системы условно выделяются две подсистемы – подсистема автоматизации проектирования агентов обработки знаний (sc-агентов) и подсистема автоматизации проектирования scp-программ.

Система может использоваться тремя способами:
- Как подсистема в рамках *Метасистемы OSTIS*. Данный вариант использования предполагает отладку необходимых компонентов в рамках *Метасистемы OSTIS* с последующим переносом их в *дочернюю ostis-систему* (см. *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*).
- Как самостоятельная ostis-система, предназначенная исключительно для разработки и отладки компонентов решателей задач. В этом случае проектируемые компоненты отлаживаются в рамках такой системы, а затем должны быть перенесены в дочернюю ostis-систему.
- Как подсистема в рамках дочерней ostis-системы. В таком варианте отладка компонентов осуществляется непосредственно в той же системе, в которой предполагается их использование, и дополнительного переноса не требуется.

Независимо от выбранного способа использования системы, разработанные компоненты впоследствии могут быть включены в состав *библиотеки многократно используемых компонентов ostis-систем*.

Выделяются два принципиально разных уровня отладки *решателя задач*:
- отладка на уровне sc-агентов;
- отладка на уровне scp-программ.

В случае отладки на уровне sc-агентов акт выполнения каждого агента считается неделимым и не может быть прерван. При этом может выполняться отладка как атомарных sc-агентов, так и неатомарных. Инициирование того или иного агента, в том числе входящего в состав неатомарного, осуществляется путем создания соответствующих конструкций в sc-памяти, таким образом, отладка может осуществляться на разных уровнях детализации агентов, вплоть до атомарных.

Система поддержки проектирования агентов может служить основой для моделирования систем агентов, использующих другие принципы коммуникации, например, непосредственный обмен сообщениями между агентами.

Отладка на уровне *scp-программ* осуществляется аналогично существующим современным подходам к отладке процедурных программ и предполагает возможность установки точек останова, пошагового выполнения программы и так далее.

Система автоматизации проектирования *решателей задач* и, соответственно, ее sc-модель, разделяется на две более частные:

Система автоматизации проектирования решателей задач ostis-систем
⇒ *базовая декомпозиция**:
{• *Система автоматизации проектирования агентов обработки знаний*
⇒ *базовая декомпозиция**:
{• *База знаний системы автоматизации проектирования агентов обработки знаний*
• *Решатель задач системы автоматизации проектирования агентов обработки знаний*
• *Пользовательский интерфейс системы автоматизации проектирования агентов обработки знаний*
}
• *Система автоматизации проектирования scp-программ*
⇒ *базовая декомпозиция**:
{• *База знаний системы автоматизации проектирования scp-программ*
• *Решатель задач системы автоматизации проектирования scp-программ*
• *Пользовательский интерфейс системы автоматизации проектирования scp-программ*
}
}

Пункт 5.3.2.1. Семантическая модель базы знаний системы автоматизации проектирования решателей задач ostis-систем

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *scp-программа*
- *точка останова**
- *точка останова*
- *некорректность в scp-программе**
- *некорректность в scp-программе*
- *ошибка в scp-программе*

База знаний системы автоматизации проектирования решателей задач ostis-систем включает в себя кроме Ядра баз знаний ostis-систем и Предметной области Базового языка программирования ostis-систем (см. *§ 3.3.5. Базовый язык программирования ostis-систем*) также описание ключевых понятий, связанных с верификацией и отладкой *scp-программ*.

Рассмотрим основные понятия, специфичные для базы знаний системы автоматизации проектирования scp-программ.

Связки отношения ***точки останова**** связывают *scp-программу* с некоторым множеством sc-переменных, соответствующих *scp-операторам* в рамках этой программы. При генерации каждого *scp-процесса*, соответствующего этой *scp-программе*, все *scp-операторы*, соответствующие таким переменным, будут добавлены во множество *точка останова*, то есть выполнение данного scp-процесса будет прерываться при достижении каждого из этих *scp-операторов*. Использование данного отношения приводит к указанию точек останова для всех *scp-процессов*, формируемых на основе заданной *scp-программы*. Для указания точки останова в рамках отдельно взятого *scp-процесса* нужный scp-оператор явно включается во множество *точка останова*.

точка останова*
∈ *квазибинарное отношение*

Во множество ***точка останова*** входят все *scp-операторы*, являющиеся точками останова в рамках какого-либо *scp-процесса*. Это означает, что в момент, когда в соответствии с переходами между *scp-операторами* по связкам отношения *последовательность действий** указанный *scp-оператор* должен стать *активным действием*, он становится *отложенным действием*, и, соответственно, выполнение всего *scp-процесса* по данной ветке приостанавливается. Чтобы продолжить выполнение, необходимо удалить указанный *scp-оператор* из множества *отложенных действий* и добавить его во множество *активных действий*.

точка останова
⊂ *scp-оператор*

Под *некорректностью в scp-программе* понимается *некорректная структура*, описывающая некорректность (не обязательно делающую невозможным выполнение соответствующих данной *scp-программе scp-процессов*), выявленную в рамках какой-либо конкретной *scp-программы*.

некорректность в scp-программе
- ⊂ *некорректная структура*
- ⊃ *ошибка в scp-программе*
- ⊃ *недостижимый scp-оператор*
- ⊃ *потенциально бесконечный цикл*

Под *ошибкой в scp-программе* понимается такая *некорректность в scp-программе*, которая делает невозможным успешное выполнение любого *scp-процесса*, соответствующего данной *scp-программе*, или даже создание такого *scp-процесса*.

ошибка в scp-программе
- ⇒ *разбиение**:
  - {• *синтаксическая ошибка в scp-программе*
    - ⇒ *пояснение**:
      - [Под *синтаксической ошибкой в scp-программе* понимается *ошибка в scp-программе*, в состав которой входит некоторая конструкция, не соответствующая синтаксису *scp-программы* или какой-либо ее части, например, конкретного *scp-оператора*.]
  - • *семантическая ошибка в scp-программе*
    - ⇒ *пояснение**:
      - [Под *семантической ошибкой в scp-программе* понимается *ошибка в scp-программе*, в состав которой входит некоторая конструкция, корректная с точки зрения синтаксиса, но некорректная с семантической точки зрения, например, нарушающая логическую целостность *scp-программы*.]
  - }
- ⇒ *разбиение**:
  - {• *ошибка в scp-программе на уровне программы*
  - • *ошибка в scp-программе на уровне множества параметров*
  - • *ошибка в scp-программе на уровне множества операторов*
  - • *ошибка в scp-программе на уровне оператора*
  - • *ошибка в scp-программе на уровне операнда*
  - }

Каждая *ошибка в scp-программе на уровне программы* описывает некорректный фрагмент, выявление которого требует анализа всей *scp-программы* как единого целого, и не может быть выполнено путем анализа ее отдельных частей, например, конкретных *scp-операторов*.

ошибка в scp-программе на уровне программы
- ⊃ *отсутствует scp-процесс, соответствующий данной scp-программе*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *не указана декомпозиция scp-процесса, соответствующего данной scp-программе*
  - ∈ *синтаксическая ошибка в scp-программе*

Каждая *ошибка в scp-программе на уровне множества параметров* описывает некорректный фрагмент, для выявления которого достаточно анализа параметров некоторой *scp-программы*, то есть явным образом выделенных аргументов *действия (scp-процессе)*, соответствующего данной scp-программе. К такого рода ошибкам относятся, например, неверное указание ролей этих аргументов в рамках данного действия.

ошибка в scp-программе на уровне множества параметров
- ⊃ *не указан тип параметра scp-программы*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *не указан порядковый номер параметра scp-программы*
  - ∈ *синтаксическая ошибка в scp-программе*

Каждая *ошибка в scp-программе на уровне множества операторов* описывает некорректный фрагмент, для выявления которого достаточно анализа множества операторов некоторой *scp-программы*, то есть элементов декомпозиции *действия (scp-процесса)*, соответствующего данной *scp-программе*. К таким ошибкам относится, например, факт отсутствия *начального оператора' scp-программы* или факт отсутствия в программе *scp-оператора завершения выполнения программы*.

ошибка в scp-программе на уровне множества операторов
- ⊃ *декомпозиция scp-процесса не содержит ни одного элемента*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *отсутствует scp-оператор завершения выполнения программы*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *scp-оператор, к которому осуществляется переход, не является частью текущего scp-процесса*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *не указана последовательность действий после выполнения текущего scp-оператора*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *отсутствует начальный оператор scp-программы*
  - ∈ *синтаксическая ошибка в scp-программе*

Каждая *ошибка в scp-программе на уровне оператора* описывает некорректный фрагмент, для выявления которого достаточно анализа одного конкретного *scp-оператора*, при этом не важно, в состав какой *scp-программы* он входит. К такого рода ошибкам относится, например, факт указания количества операндов *scp-оператора*, не соответствующего спецификации соответствующего класса *scp-операторов*.

ошибка в scp-программе на уровне оператора
- ⊃ *scp-оператор не принадлежит ни одному из атомарных классов scp-операторов*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *ни один операнд scp-оператора удаления не помечен как удаляемый sc-элемент*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *в scp-операторе поиска пятиэлементной конструкции совпадает второй и четвертый операнд*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *scp-оператор поиска не содержит ни одного операнда с заданным значением*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *scp-оператор поиска с формированием множеств не содержит ни одного операнда с атрибутом формируемое множество*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *атрибутом формируемое множество отмечен операнд, которому соответствует операнд с заданным значением*
  - ∈ *синтаксическая ошибка в scp-программе*
- ⊃ *количество операндов scp-оператора не совпадает со спецификацией*
  - ∈ *синтаксическая ошибка в scp-программе*

Каждая *ошибка в scp-программе на уровне операнда* описывает некорректный фрагмент, для выявления которого достаточно анализа одного конкретного операнда в рамках scp-программы, точнее sc-дуги принадлежности, связывающей указанный операнд и соответствующий *scp-оператор*, при этом не важно, какой именно *scp-оператор*. К такого рода ошибкам относится, например, факт отсутствия ролевого отношения, указывающего на номер операнда в рамках *scp-оператора*.

ошибка в scp-программе на уровне операнда
- ⊃ *не указан номер операнда в рамках scp-оператора*
  - ∈ *синтаксическая ошибка в scp-программе*

некорректность в scp-программе*
- ∈ *бинарное отношение*
- ⇒ *первый домен**:
  некорректность в scp-программе
- ⇒ *второй домен***:
  scp-программа

Отношение *scp-программа поиска некорректности в scp-программе** связывает *класс некорректностей в scp-программе* и *scp-программу*, которая может использоваться для выявления соответствующей некорректности в какой-либо другой *scp-программе*.

Указанная *scp-программа* должна иметь единственный параметр, который является *in-параметром'* и, в зависимости от соответствующего класса некорректностей в *scp-программе*, обозначает:
- саму *scp-программу* в случае выявления *некорректности в scp-программе* вообще или *ошибки в scp-программе на уровне программы*;
- *scp-процесс*, являющийся *ключевым sc-элементом* данной *scp-программы* в случае выявления ошибки в *scp-программе на уровне множества параметров*;

- *множество операторов* данной *scp-программы* в случае выявления *ошибки в scp-программе на уровне множества операторов*;
- *знак конкретного scp-оператора* в случае выявления ошибки в *scp-программе на уровне оператора*;
- *sc-дугу принадлежности* в случае выявления *ошибки в scp-программе на уровне операнда*.

Если в результате верификации *scp-программы* выявлена некорректность, то формируется соответствующая *структура* и генерируется связка отношения *некорректность в scp-программе**.

Пункт 5.3.2.2. Семантическая модель решателя задач системы автоматизации проектирования решателей задач ostis-систем

⇒ *ключевое понятие**:
- *Решатель задач системы автоматизации проектирования агентов обработки знаний*
- *Решатель задач системы автоматизации проектирования scp-программ*

Решатель задач системы автоматизации проектирования агентов обработки знаний
⇒ *декомпозиция**:
{• *Множество методов решателя задач системы автоматизации проектирования агентов обработки знаний*
- *Машина обработки знаний системы автоматизации проектирования агентов обработки знаний*
  ⇒ *декомпозиция абстрактного sc-агента**:
  {• *Абстрактный sc-агент верификации sc-агентов*
    ⇒ *декомпозиция абстрактного sc-агента**:
    {• *Абстрактный sc-агент верификации спецификации sc-агента*
    • *Абстрактный sc-агент проверки неатомарного sc-агента на непротиворечивость его спецификации спецификациям более частных sc-агентов в его составе*
    }
  • *Абстрактный sc-агент отладки коллективов sc-агентов*
    ⇒ *декомпозиция абстрактного sc-агента**:
    {• *Абстрактный sc-агент поиска всех выполняющихся процессов, соответствующих заданному sc-агенту*
    • *Абстрактный sc-агент инициирования заданного sc-агента на заданных аргументах*
    • *Абстрактный sc-агент активации заданного sc-агента*
    • *Абстрактный sc-агент деактивации заданного sc-агента*
    • *Абстрактный sc-агент установки блокировки заданного типа для заданного процесса на заданный sc-элемент*
    • *Абстрактный sc-агент снятия всех блокировок заданного процесса*
    • *Абстрактный sc-агент снятия всех блокировок с заданного sc-элемента*
    }
  }
}

Единственным аргументом класса действий, соответствующего *Абстрактному sc-агенту поиска всех выполняющихся процессов, соответствующих заданному sc-агенту*, *Абстрактному sc-агенту активации заданного sc-агента*, *Абстрактному sc-агенту деактивации заданного sc-агента*, является знак этого *sc-агента*.

Класс действий, соответствующий *Абстрактному sc-агенту инициирования заданного sc-агента на заданных аргументах*, имеет два аргумента. Первый аргумент является знаком инициируемого sc-агента, второй — знаком связки, в которую под соответствующими атрибутами входят sc-элементы, которые станут аргументами соответствующего *действия в sc-памяти*.

Класс действий, соответствующий *Абстрактному sc-агенту установки блокировки заданного типа на заданный sc-элемент*, имеет три аргумента. Первый аргумент является знаком класса блокировок, второй — знаком процесса в sc-памяти, третий — sc-элементом, на который должна быть установлена блокировка.

Единственным аргументом класса действий, соответствующего *Абстрактному sc-агенту снятия всех блокировок заданного процесса*, является знак этого *процесса в sc-памяти*.

Единственным аргументом класса действий, соответствующего *Абстрактному sc-агенту снятия всех блокировок с заданного sc-элемента*, является знак этого *sc-элемента*.

Решатель задач системы автоматизации проектирования scp-программ

⇒ *декомпозиция**:
{• *Множество методов решателя задач системы автоматизации проектирования агентов обработки знаний*
• *Машина обработки знаний системы автоматизации проектирования агентов обработки знаний*
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент верификации scp-программ*
• *Абстрактный sc-агент отладки scp-программ*
⇒ *декомпозиция абстрактного sc-агента**:
{• *Абстрактный sc-агент запуска заданной scp-программы для заданного множества входных данных*
• *Абстрактный sc-агент запуска заданной scp-программы для заданного множества входных данных в режиме пошагового выполнения*
• *Абстрактный sc-агент поиска всех scp-операторов в рамках scp-программы*
• *Абстрактный sc-агент поиска всех точек останова в рамках scp-процесса*
• *Абстрактный sc-агент добавления точки останова в scp-программу*
• *Абстрактный sc-агент удаления точки останова из scp-программы*
• *Абстрактный sc-агент добавления точки останова в scp-процесс*
• *Абстрактный sc-агент удаления точки останова из scp-процесса*
• *Абстрактный sc-агент продолжения выполнения scp-процесса на один шаг*
• *Абстрактный sc-агент продолжения выполнения scp-процесса до точки останова или завершения*
• *Абстрактный sc-агент просмотра информации об scp-процессе*
• *Абстрактный sc-агент просмотра информации об scp-операторе*
}
}
}

Алгоритм работы *абстрактного sc-агента верификации scp-программ* сводится к поиску некорректностей в рамках *scp-программы* на основе определений, соответствующих различным классам таких некорректностей, а также посредством запуска соответствующих данным классам некорректностей *scp-программ поиска некорректности в scp-программе**.

Результатом работы *абстрактного sc-агента верификации scp-программ* является формирование в *sc-памяти структур*, описывающих некорректности в исследуемой *scp-программе*, если таковые имеются.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту верификации scp-программ*, является знак верифицируемой *scp-программы*.

Классы действий, соответствующие *абстрактному sc-агенту запуска заданной scp-программы для заданного множества входных данных* и *абстрактному sc-агенту запуска заданной scp-программы для заданного множества входных данных в режиме пошагового выполнения*, имеют два аргумента. Первый аргумент является знаком запускаемой scp-программы, второй — знаком связки, в которую под соответствующими атрибутами входят sc-элементы, которые станут аргументами соответствующего scp-процесса.

В режиме пошагового выполнения предполагается, что на каждом шаге инициируется выполнение всех scp-операторов в рамках заданного scp-процесса, для которых предыдущий scp-оператор стал прошлой сущностью (выполнился). В свою очередь, шаг заканчивается, когда все инициированные таким образом операторы закончат выполнение. Таким образом, в случае, если в рамках scp-программы есть параллельные ветви, то на одном шаге могут одновременно инициироваться два и более scp-оператора.

Классы действий, соответствующие *абстрактному sc-агенту добавления точки останова в scp-программу, абстрактному sc-агенту удаления точки останова из scp-программы, абстрактному sc-агенту добавления точки останова в scp-процесс* и *абстрактному sc-агенту удаления точки останова из scp-процесса*, имеют два аргумента. Первый аргумент является знаком scp-программы или scp-процесса соответственно, второй — знаком scp-оператора, входящего в состав этой scp-программы или scp-процесса.

Единственным аргументом классов действий, соответствующих *абстрактному sc-агенту поиска всех точек останова в рамках scp-процесса, абстрактному sc-агенту продолжения выполнения scp-процесса на один шаг, абстрактному sc-агенту продолжения выполнения scp-процесса до точки останова или завершения* и *абстрактному sc-агенту просмотра информации об scp-процессе*, является знак scp-процесса, с которым будет выполнено соответствующее действие.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту поиска всех scp-операторов в рамках scp-программы*, является знак этой scp-программы.

Единственным аргументом класса действий, соответствующего *абстрактному sc-агенту просмотра информации об scp-операторе*, является знак scp-оператора, входящего в состав некоторого scp-процесса. Результатом работы данного агента является структура, описывающая значения операндов данного scp-оператора, его атомарный тип и другую служебную информацию, полезную для разработчика.

Пункт 5.3.2.3. Семантическая модель пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *команда пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем*
- *команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний*
- *команда пользовательского интерфейса системы автоматизации проектирования scp-программ*

пользовательский интерфейс системы автоматизации проектирования решателей задач ostis-систем представлен набором интерфейсных команд, позволяющих пользователю инициировать деятельность нужного агента, входящего в состав этой системы.

команда пользовательского интерфейса системы автоматизации проектирования решателей задач ostis-систем
⇒ *разбиение**:
{
- *команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний*
- *команда пользовательского интерфейса системы автоматизации проектирования программ языка SCP*

}

команда пользовательского интерфейса системы автоматизации проектирования агентов обработки знаний
⇒ *разбиение**:
{
- *команда верификации sc-агентов*
  ⇒ *разбиение**:
  {
  - *команда верификации спецификации sc-агента*
  - *команда верификации неатомарного sc-агента на непротиворечивость его спецификации спецификациям более частных sc-агентов в его составе*

  }
- *команда отладки коллективов sc-агентов*
  ⇒ *разбиение**:
  {
  - *команда поиска всех выполняющихся процессов, соответствующих заданному sc-агенту*
  - *команда инициирования заданного sc-агента на заданных аргументах*
  - *команда активации заданного sc-агента*
  - *команда деактивации заданного sc-агента*
  - *команда установки блокировки заданного типа для заданного процесса на заданный sc-элемент*
  - *команда снятия всех блокировок заданного процесса*
  - *команда снятия всех блокировок с заданного sc-элемента*

  }

}

команда пользовательского интерфейса системы автоматизации проектирования scp-программ
⇒ *разбиение**:
{
- *команда верификации scp-программ*
- *команда отладки scp-программ*
  ⇒ *разбиение**:
  {
  - *команда запуска заданной scp-программы для заданного множества входных данных*
  - *команда запуска заданной scp-программы для заданного множества входных данных в режиме пошагового выполнения*

- *команда поиска всех scp-операторов в рамках scp-программы*
- *команда поиска всех точек останова в рамках scp-процесса*
- *команда добавления точки останова в scp-программу*
- *команда удаления точки останова из scp-программы*
- *команда добавления точки останова в scp-процесс*
- *команда удаления точки останова из scp-процесса*
- *команда продолжения выполнения scp-процесса на один шаг*
- *команда продолжения выполнения scp-процесса до точки останова или завершения*
- *команда просмотра информации об scp-процессе*
- *команда просмотра информации об scp-операторе*

}

}

§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем

⇒ *ключевое понятие**:
- *решатель задач ostis-системы*
- *библиотека многократно используемых компонентов решателей задач*
- *многократно используемый компонент решателей задач*
- *метод*
- *Средства автоматизации библиотеки многократно используемых компонентов решателей задач*
- *отношение, специфицирующее многократно используемый компонент решателей задач ostis-систем*
- *Решатель задач библиотеки многократно используемых компонентов решателей задач*

Библиотека многократно используемых компонентов решателей задач в составе Метасистемы OSTIS является важнейшим фрагментом Метасистемы OSTIS, обеспечивающим надежность проектируемых решателей задач и повышение скорости их разработки.

Библиотека включает:
- множество компонентов решателей задач;
- средства спецификации компонентов решателей задач;
- средства поиска компонентов решателей задач на основе их спецификации, уточняющие общие средства поиска компонентов в рамках библиотеки, рассмотренные в *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*.

библиотека многократно используемых компонентов решателей задач ostis-систем
⇒ *обобщенная декомпозиция**:
{• *база знаний библиотеки многократно используемых компонентов решателей задач ostis-систем*
⇒ *примечание**:
[База знаний представляет собой иерархию многократно используемых компонентов решателей задач ostis-систем и их спецификацию.]
• *решатель задач библиотеки многократно используемых компонентов решателей задач ostis-систем*
⇒ *примечание**:
[Решатель задач позволяет осуществлять поиск компонентов решателей задач, находить зависимости таких компонентов, конфликты между компонентами.]
• *интерфейс библиотеки многократно используемых компонентов решателей задач ostis-систем*
⇒ *примечание**:
[Интерфейс библиотеки позволяет подключиться к библиотеке и получить доступ к компонентам хранящимся в ней и к ее функционалу.]

}

Если *многократно используемый компонент решателей задач* является *платформенно-зависимым многократно используемым компонентом ostis-системы*, то его интеграция производится в соответствии с инструкцией, предоставляемой разработчиком в зависимости от платформы, как и для любого компонента такого рода. В противном случае процесс интеграции можно конкретизировать в зависимости от подклассов данного типа компонентов.

Рассмотрим классификацию многократно используемых компонентов решателей задач ostis-систем.

многократно используемый компонент решателей задач ostis-систем
⊃ *метод*

⊃ *пакет программ*
⊃ *абстрактный sc-агент*
⊃ *решатель задач ostis-системы*
⇒ *примечание**:
[Целые решатели задач могут быть многократно используемыми компонентами в случае разработки интеллектуальных систем, назначение которых совпадает.]

абстрактный sc-агент
⇒ *разбиение**:
{• *абстрактный sc-агент информационного поиска*
• *абстрактный sc-агент погружения интегрируемого знания в базу знаний*
• *абстрактный sc-агент выравнивания онтологии интегрируемого знания с основной онтологией текущего состояния базы знаний*
• *абстрактный sc-агент планирования решения явно сформулированных задач*
• *абстрактный sc-агент логического вывода*
• *абстрактный sc-агент верификации базы знаний*
• *абстрактный sc-агент редактирования базы знаний*
• *абстрактный sc-агент автоматизации деятельности разработчиков базы знаний*
}

Классификация решателей задач ostis-систем подробно рассмотрена в *§ 3.3.6. Решатели задач ostis-систем*.

Под *многократно используемым абстрактным sc-агентом* подразумевается компонент, соответствующий некоторому *абстрактному sc-агенту*, который может быть использован в решателях задач других ostis-систем, возможно, в составе более сложных *неатомарных абстрактных sc-агентов*. Спецификация многократно используемого *абстрактного sc-агента* должна содержать всю информацию, необходимую для функционирования соответствующего *sc-агента* в дочерней ostis-системе.

Таким образом, указанная спецификация формируется следующим образом:
- В нее включается *sc-узел*, обозначающий соответствующий *абстрактный sc-агент*, и вся его спецификация, то есть, как минимум, указание *ключевых sc-элементов sc-агента**, *условия инициирования и результат**, *первичного условия инициирования**, *sc-описание поведения sc-агента* и класса решаемых им задач;
- В случае, если специфицируется многократно используемый *абстрактный sc-агент*, который рассматривается как *неатомарный абстрактный sc-агент*, то его спецификация будет содержать *sc-узлы*, обозначающие все более частные *абстрактные sc-агенты*, а также все их спецификации;
- Для каждого *атомарного абстрактного sc-агента*, знак которого вошел в такую спецификацию, необходимо выбрать вариант его реализации (то есть элемент класса *платформенно-независимый абстрактный sc-агент* или *платформенно-зависимый абстрактный sc-агент*, связанный с исходным *атомарным абстрактным sc-агентом* связкой отношения *включение**) и включить в указанную спецификацию sc-узел, обозначающий указанную реализацию, а также знаки всех программ, входящие во множество, связанное с указанной реализацией отношением *программа sc-агента**;
- В спецификацию компонента включаются также все связки отношений, связывающие уже включенные в его состав sc-элементы, а также сами знаки этих отношений (например, *включение**, *программа sc-агента** и так далее).

После того как многократно используемый *абстрактный sc-агент* был скопирован в дочернюю ostis-систему, необходимо сгенерировать sc-узел, обозначающий конкретный *sc-агент*, работающий в данной системе и принадлежащий выбранной реализации *абстрактного sc-агента*, и добавить его во множество *активных sc-агентов* при необходимости.

Под *многократно используемой программой* подразумевается компонент, соответствующий программе, записанной на произвольном языке программирования, которая ориентирована на обработку *структур*, хранящихся в памяти *ostis-системы*. Приоритетным в данном случае является использование *scp-программ* по причине их платформенной независимости, за исключением случаев проектирования некоторых компонентов интерфейса, когда полная платформенная независимость невозможна (например, при проектировании *эффекторных sc-агентов и рецепторных sc-агентов*).

Также каждую *scp-программу*, попавшую в *дочернюю ostis-систему* при копировании *многократно используемого компонента решателя задач ostis-систем*, необходимо добавить во множество *корректных scp-программ* (корректность верифицируется при попадании в библиотеку компонентов).

Для удобства работы с библиотекой многократно используемых компонентов необходимы также средства автоматизации поиска компонентов на основе заданной спецификации, представляющие собой *решатель задач ostis-системы* частного вида.

Ниже представлена структура такого *решателя задач*:

Средства автоматизации библиотеки многократно используемых компонентов решателей задач ostis-систем

⇒ *декомпозиция**:

{• *Множество методов, входящих в состав средств автоматизации библиотеки многократно используемых компонентов решателей задач ostis-систем*

• *Машина обработки знаний библиотеки многократно используемых компонентов решателей задач ostis-систем*

⇒ *декомпозиция абстрактного sc-агента**:

{• *Абстрактный sc-агент формирования неатомарного компонента решателей задач ostis-систем из атомарных*

• *Абстрактный sc-агент поиска всех неатомарных компонентов, частью которых является заданный атомарный компонент*

• *Абстрактный sc-агент поиска всех сопутствующих компонентов*

• *Абстрактный sc-агент поиска sc-агента по условию инициирования*

• *Абстрактный sc-агент поиска sc-агента по результату работы*

• *Абстрактный sc-агент поиска scp-программы по входным/выходным параметрам*

• *Абстрактный sc-агент поиска sc-агентов, для которых элементы заданного множества являются ключевыми sc-элементами*

}

}

Под *неатомарным компонентом решателей задач ostis-систем* понимается такой компонент, в составе которого можно выделить другие компоненты, которые могут использоваться самостоятельно, отдельно от исходного компонента. Чаще всего в роли таких неатомарных компонентов выступают неатомарные sc-агенты, в составе которых могут быть выделены самодостаточные sc-агенты, которые могут быть использованы отдельно от исходного неатомарного, или scp-программы, которые являются общими для нескольких агентов и могут быть использованы не только в составе неатомарного sc-агента. Таким образом, задачей *Абстрактного sc-агента формирования неатомарного компонента решателей задач ostis-систем из атомарных* является формирование структуры, содержащей в себе полный sc-текст неатомарного компонента, включая спецификации всех sc-агентов в его составе, а также тексты всех необходимых scp-программ. Формирование такой структуры необходимо для того, чтобы упростить процесс копирования указанного компонента в другие ostis-системы.

Под сопутствующим компонентом понимается компонент, который часто используется в ostis-системе одновременно с некоторым другим компонентом (см. *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*)

Абстрактный sc-агент поиска sc-агентов, для которых элементы заданного множества являются ключевыми sc-элементами играет важную роль при внесении изменений в базу знаний, в частности, при переопределении каких-либо понятий. Указанный sc-агент позволяет выявить те sc-агенты, для которых могут потребоваться изменения в алгоритме работы в связи с изменением семантической трактовки каких-либо понятий.

Рассмотрим отношения необходимые для спецификации многократно используемого компонента *решателя задач*, его поиска и установки в дочернюю ostis-систему.

отношение, специфицирующее многократно используемый компонент решателей задач ostis-систем

⊂ *отношение, специфицирующее многократно используемый компонент ostis-систем*

∋ *первичное условие инициирования**

⇒ *пояснение**:

[Связки отношения первичное условие инициирования* связывают между собой sc-узел, обозначающий абстрактный sc-агент и бинарную ориентированную пару, описывающую первичное условие инициирования данного абстрактного sc-агента, то есть такой ситуации в sc-памяти, которая побуждает sc-агента перейти в активное состояние и начать проверку наличия своего полного условия инициирования.

Первым компонентом данной ориентированной пары является знак некоторого подмножества понятия событие, например событие добавления выходящей sc-дуги, то есть по сути конкретный тип события в sc-памяти.

Вторым компонентом данной ориентированной пары является произвольный в общем случае sc-элемент, с которым непосредственно связан указанный тип события в sc-памяти, то есть, например, sc-элемент, из которого выходит либо в который входит генерируемая либо удаляемая sc-дуга, либо файл ostis-системы, содержимое которого было изменено.

После того, как в sc-памяти происходит некоторое событие, активизируются все активные sc-агенты, первичное условие инициирования* которых соответствует произошедшему событию.]

⇒ *первый домен**:

абстрактный sc-агент
⇒ *второй домен**:
бинарная ориентированная пара
∋ *условие инициирования и результат**
⇒ *пояснение**:
[Связки отношения условие инициирования и результат* связывают между собой sc-узел, обозначающий абстрактный sc-агент и бинарную ориентированную пару, связывающую условие инициирования данного абстрактного sc-агента и результаты выполнения экземпляров данного sc-агента в какой-либо конкретной системе.

Указанную ориентированную пару можно рассматривать как логическую связку импликации, при этом на sc-переменные, присутствующие в обеих частях связки, неявно накладывается квантор всеобщности, на sc-переменные, присутствующие либо только в посылке, либо только в заключении неявно накладывается квантор существования.

Первым компонентом указанной ориентированной пары является логическая формула, описывающая условие инициирования описываемого абстрактного sc-агента, то есть конструкции, наличие которой в sc-памяти побуждает sc-агент начать работу по изменению состояния sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.

Вторым компонентом указанной ориентированной пары является логическая формула, описывающая возможные результаты выполнения описываемого абстрактного sc-агента, то есть описание произведенных им изменений состояния sc-памяти. Данная логическая формула может быть как атомарной, так и неатомарной, в которой допускается использование любых связок логического языка.]
⇒ *первый домен**:
абстрактный sc-агент
⇒ *второй домен**:
бинарная ориентированная пара

Заключение к Главе 5.3.

В данной главе предложена *Методика построения и модификации машины обработки знаний ostis-систем*, которая включает несколько этапов:
- формирование требований и спецификация *машины обработки знаний*;
- формирование коллектива sc-агентов, входящих в состав разрабатываемой машины;
- разработка алгоритмов атомарных sc-агентов;
- реализация scp-программ;
- верификация разработанных компонентов;
- отладка разработанных компонентов, исправление ошибок.

Методика построения и модификации машины обработки знаний ostis-систем основана на онтологии деятельности разработчиков *машины обработки знаний*. Наличие такой онтологии позволяет:
- частично автоматизировать процесс построения и модификации *машины обработки знаний*;
- повысить эффективность информационной поддержки разработчиков, поскольку данная онтология может быть включена в *Базу знаний Метасистемы OSTIS*.

Также предложена модель *Системы автоматизации проектирования решателей задач ostis-систем*. Система может использоваться тремя способами:
- Как подсистема в рамках *Метасистемы OSTIS*. Данный вариант использования предполагает отладку необходимых компонентов в рамках метасистемы с последующим переносом их в *дочернюю ostis-систему*.
- Как *самостоятельная ostis-система*, предназначенная исключительно для разработки и отладки компонентов решателей задач. В этом случае проектируемые компоненты отлаживаются в рамках такой системы, а затем должны быть перенесены в дочернюю ostis-систему.
- Как подсистема в рамках дочерней ostis-системы. В таком варианте отладка компонентов осуществляется непосредственно в той же системе, в которой предполагается их использование, и дополнительного переноса не требуется.

Рассмотрены многократно используемые компоненты *решателей задач ostis-систем* и соответствующая им библиотека.

В дальнейшем планируется уточнение принципов и разработка соответствующей методики проектирования *методов* решения задач.

Глава 5.4.

Методика и средства компонентного проектирования интерфейсов ostis-систем

⇒ *автор**:
- *Садовский М. Е.*
- *Жмырко А. В.*

⇒ *аннотация**:
[Проектирование *интерфейса компьютерных систем* — это один из наиболее важных этапов разработки любой системы. Пользователь при использовании *интерфейса* должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Эффективность работы пользователя и его интерес обеспечивает правильно сформулированная методика разработки и проектирования пользовательского интерфейса.
В рамках главы рассмотрены этапы проектирования *пользовательских интерфейсов* и этапы проектирования *адаптивных интеллектуальных мультимодальных пользовательских интерфейсов*.]

⇒ *подраздел**:
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*
- *§ 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем*

⇒ *ключевое понятие**:
- *библиотека многократно используемых компонентов пользовательских интерфейсов ostis-систем*
- *метод оценки пользовательских интерфейсов*
- *качественный метод оценки пользовательских интерфейсов*
- *количественный метод оценки пользовательских интерфейсов*
- *тестирование удобства использования пользовательских интерфейсов*
- *отслеживание движения глаз*
- *экспертная оценка пользовательских интерфейсов*
- *А/Б тестирование пользовательских интерфейсов*
- *древовидное тестирование пользовательских интерфейсов*

⇒ *библиография**:
- *Ehlert P.IntelUIIaS-2003bk*
- *Kong J..Desig oHCAM-2011art*
- *Sadouski M..Metho aTfCID-2022art*
- *Ivory M..tState otAiAUEoUI-2001art*
- *Jeffries R..UserIEitRW-1991art*
- *ISOEoHSIPGoVUIE-2016el*
- *Зенг В.А..ОценкКППИНП-2019ст*
- *ISOTREoHSI-2002el*
- *Корончик Д.Н.СеманТКПП-2011ст*
- *Корончик Д.Н.ПользИИМП-2014ст*

Введение в Главу 5.4.

В настоящее время организация взаимодействия пользователя с компьютерной системой основана на парадигме грамотного пользователя, который знает, как управлять системой и несет полную ответственность за качество взаимодействия с ней. Многообразие форм и видов *интерфейсов* приводит к необходимости пользователя адаптироваться к каждой конкретной системе, обучаться принципам взаимодействия с ней для решения необходимых ему задач.

Проектирование *пользовательских интерфейсов* включает в себя ряд последовательных этапов.

Методика проектирования *пользовательских интерфейсов* является важной частью *Технологии OSTIS*, так как она описывает этапы проектирования *пользовательских интерфейсов ostis-систем*, что позволяет ускорить процесс разработки, обеспечивает создание удобных *пользовательских интерфейсов*, улучшает опыт использования

интеллектуальной системы и повышает эффективность работы пользователей, учитывая их потребности и предпочтения.

§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов

Среди существующих методик проектирования *адаптивных интеллектуальных мультимодальных пользовательских интерфейсов* можно выделить методики, предложенные в *Ehlert P.IntelUIIaS-2003bk* и *Kong J..Desig oHCAM-2011art*.

В рамках работы *Ehlert P.IntelUIIaS-2003bk* выделяется 4 основных этапа проектирования:
- анализ;
- разработка *интерфейса*;
- оценка *интерфейса*;
- доработка и усовершенствование.

Этап анализа является, вероятно, самой важной фазой в любом процессе проектирования, в том числе в проектировании *интерфейсов ostis-систем*. В процессе проектирования традиционного *интерфейса* необходимо проанализировать, кто является обычным пользователем, какие задачи *интерфейс* должен поддерживать.

В *пользовательском интерфейсе* часто нет среднего пользователя. В идеале, *пользовательский интерфейс* должен быть способен адаптироваться к любому пользователю в любой среде. Поэтому используемая техника адаптации должна быть разработана таким образом, чтобы она могла поддерживать все типы пользователей.

Этап *анализа* включает выполнение четырех взаимосвязанных видов анализа:
- *функциональный анализ*;
- *анализ данных*;
- *анализ пользователей*;
- *анализ среды*.

В рамках *функционального анализа* необходимо дать ответ на вопрос: “каковы основные функции системы”. В рамках *анализа данных* необходимо определить значение и структуру данных, используемых в приложении. В рамках *анализа пользователей* необходимо выделить типы пользователей и их возможности в интеллектуальном и когнитивном плане. В рамках *анализа среды* необходимо определить требования, предъявляемые к среде, в которой будет работать система.

Результатом данного этапа является спецификация целей и информационных потребностей пользователя, а также спецификация функций и информации, которые требуются системе. *Разработка интерфейса* включает следующие шаги:
- *создание модели интерфейса* в соответствии с этапом анализа;
- реализация модели *интерфейса*.

Результатом данного этапа является *пользовательский интерфейс*, который, по мнению разработчика, удовлетворяет требованиям пользователей и соответствует требованиям, сформулированным на этапе анализа.

Оценка интерфейса предполагает, что:
- требования, которые были сформулированы на этапе *анализа*, должны быть удовлетворены;
- эффективность модели *интерфейса* должна быть исследована.

На этапе *оценки интерфейса* необходимо вернуться к требованиям *этапа анализа*. Требования, которые были сформулированы на *этапе анализа*, должны быть выполнены, а также должна быть исследована эффективность модели *интерфейса*. Чтобы определить эту эффективность, необходимо определить критерии эффективности.

Очень важным, но субъективным критерием является удовлетворенность пользователя. Поскольку пользователь должен работать с *интерфейсом*, он имеет право голоса в вопросе о том, удобно ли работать с *интерфейсом*, насколько привлекательным является интерфейс.

Критериями эффективности могут выступать различные показатели, такие как:
- количество ошибок;
- время выполнения задачи;
- отношение пользователя к *интерфейсу*;
- и так далее.

Доработка и усовершенствование осуществляется на основе проблем, выявленных на этапе оценки. В рамках данного этапа вносится ряд улучшений в модель *интерфейса*. Затем начинается новый цикл проектирования. Этот итеративный процесс будет продолжаться до тех пор, пока результат оценки не будет удовлетворять обозначенным требованиям.

Методика, предложенная в *Kong J..Desig oHCAM-2011art* включает 6 этапов:
- моделирование *пользовательского интерфейса* (описание абстрактного *пользовательского интерфейса*);
- проектирование *пользовательского интерфейса* по умолчанию (стандартная версия, конкретный *пользовательский интерфейс*);
- разработка *пользовательского интерфейса* (расширение или замена *пользовательского интерфейса* по умолчанию) — этот шаг опускается, когда система генерирует *пользовательский интерфейс* по умолчанию автоматически;
- создание контекста использования (идентификация и создание контекста использования — модели пользователя, модель устройства и модель среды/платформы);
- адаптация *пользовательского интерфейса* — автоматически (адаптация пользовательского интерфейса во время выполнения для соответствия конкретного контекста использования);
- кастомизация *пользовательского интерфейса* — настройка *пользовательского интерфейса* самим пользователем (адаптируемость).

На основе рассмотренных методик проектирования *интерфейсов* можно выделить следующие общие этапы:
- *анализ контекста использования и задач пользователей*;
- *проектирование и разработка интерфейса*;
- *оценка качества спроектированного интерфейса.*

Среди недостатков предложенных подходов можно выделить:
- знания по каждому этапу проектирования находятся у разных специалистов в неформализованном неунифицированном виде;
- отсутствие *этапа формализованного документирования* этапов проектирования приводит в дальнейшем к необходимости создания отдельных help-систем для пользователей, разработчиков и так далее.

Таким образом, на основе проведенного анализа предлагаемая методика проектирования *интерфейсов ostis-систем* (структура *пользовательского интерфейса ostis-системы* была рассмотрена в *Главе 4.1. Общие принципы организации интерфейсов ostis-систем*) будет включать (см. *Sadouski M..Metho aTfCID-2022art*):
- анализ пользователя, его задач и целей, а также контекста использования;
- анализ требований к *пользовательскому интерфейсу* и спецификация проектируемого *пользовательского интерфейса*;
- задачно-ориентированная декомпозиция *пользовательского интерфейса*;
- проектирование *пользовательского интерфейса* по умолчанию;
- разработка *пользовательского интерфейса*;
- анализ *пользовательского интерфейса* и его адаптация.

Поскольку знания о конкретном этапе обычно находятся у разных экспертов, особенностью предлагаемого подхода является обязательное формализованное документирование знаний в унифицированном виде и применение на каждом из этапов компонентного подхода.

Для применения компонентного подхода предлагается использовать *библиотеку многократно используемых компонентов ostis-систем*.

Далее будет рассмотрен каждый этап по отдельности.

Анализ пользователя, его задач и целей, а также контекста использования

Результаты первого этапа, такие как: модель конкретного пользователя, его потребности и контекст использования системы (устройство, окружающая среда) должны быть формализованы в рамках соответствующих онтологий *базы знаний интерфейса*. При этом в процессе формализации по необходимости должны быть переиспользованы *компоненты базы знаний* из *библиотеку многократно используемых компонентов ostis-систем*, а новые компоненты могут пополнить эту же библиотеку.

Анализ требований к пользовательскому интерфейсу и спецификация проектируемого пользовательского интерфейса

Результатом второго этапа являются конечные требования к *интерфейсу*, которые должны быть сформулированы относительно модели пользователя и его цели, а также относительно контекста использования.

Спецификация включает в себя список задач решаемых интерфейсом, описание *внешних языков представления знаний*.

Результаты должны быть также формализованы, а в процессе выполнения могут быть использованы существующие компоненты из *библиотеки многократно используемых компонентов интерфейсов ostis-систем*.

Задачно-ориентированная декомпозиция пользовательского интерфейса

На этапе задачно-ориентированной декомпозиции *пользовательского интерфейса* специфицированный интерфейс разбивается на интерфейсные подсистемы, которые могут разрабатываться параллельно. Это позволяет сократить сроки проектирования *пользовательского интерфейса*. Целесообразно проводить разбиение таким

образом, чтобы максимальное количество подсистем уже имелось в *библиотеке многократно используемых компонентов пользовательских интерфейсов ostis-систем*.

Проектирование пользовательского интерфейса по умолчанию

В соответствии с требованиями к *пользовательскому интерфейсу*, строится модель *адаптивного интеллектуального мультимодального пользовательского интерфейса*, которая является результатом третьего этапа.

Такая модель будет включать в себя формализованную модель *базы знаний* и *решателя задач*. При проектировании могут быть использованы компоненты *интерфейса*, *базы знаний* и *решателя задач*. Компоненты будут записаны в унифицированном виде, что позволит обеспечить их автоматическую совместимость.

Разработка пользовательского интерфейса

Результатом четвертого этапа является реализация спроектированного *пользовательского интерфейса*.

После разработки *пользовательского интерфейса* выделяются типовые фрагменты интерфейса. Специфицируя фрагменты интерфейса необходимым образом следует включать их в *библиотеку многократно используемых компонентов пользовательских интерфейсов ostis-систем*.

При разработке пользовательского интерфейса также можно использовать готовые компоненты интерфейса из *библиотеки многократно используемых компонентов пользовательских интерфейсов ostis-систем*.

Анализ пользовательского интерфейса и его адаптация

На данном этапе используются готовые компоненты *решателя задач*: *sc-агенты анализа пользовательского интерфейса* и *sc-агенты изменения модели пользовательского интерфейса на основе логических правил адаптации*.

Таким образом будет сформирована *база знаний* проектируемого *интерфейса*, которая автоматически может быть использована в качестве help-системы для пользователей, разработчиков и так далее.

Анализ методов оценки пользовательских интерфейсов

Оценка *пользовательского интерфейса* необходима для улучшения коммуникации между *интеллектуальными системами* и их пользователями. Существует множество методов для оценки *пользовательских интерфейсов*, направленных, в основном, на выявление проблем с использованием системы и минимизацию риска ошибок (см. *Ivory M..tState otAiAUEoUI-2001art*, *Jeffries R..UserIEitRW-1991art*). Однако до сих пор отсутствует комплексный подход к оценке *пользовательских интерфейсов интеллектуальных систем*.

При оценке *пользовательских интерфейсов* большую роль играет человеческий фактор, а основными участниками являются пользователи и эксперты. Очень сложно оценить правильность решения по адаптации *пользовательского интерфейса интеллектуальной системы* и оценить его без участия человека. Конечно, есть небольшие обобщенные правила построения *пользовательских интерфейсов*, такие как:

- *интерфейс* должен быть интуитивно понятен для конечного пользователя;
- *интерфейс* должен быть доступным для пользователей с ограниченными возможностями и пользователей, впервые сталкивающимися с информационными технологиями;
- достижение цели пользователем должно осуществляться наименее возможным количеством шагов (см. *ISOEoHSIPGoVUIE-2016el*).

Оценка *пользовательских интерфейсов интеллектуальных систем* имеет свои особенности и требует специальных методов и инструментов.

Некоторые *методы оценки пользовательских интерфейсов* интеллектуальных систем включают:

- Оценку точности и полноты.

  Это метод, при котором оценивается точность и полнота ответов системы на запросы пользователей.

  Точность означает, насколько хорошо *пользовательский интерфейс* отражает действительный опыт пользователя и предлагает ему наиболее подходящий контент в соответствии с его потребностями и предпочтениями. Оценить точность возможно путем анализа, насколько легко использовать интерфейс, насколько он отражает действительные возможности интеллектуальной системы и насколько он помогает пользователям достигать своих целей более эффективно.

  Полнота, с другой стороны, означает, насколько хорошо *пользовательский интерфейс* предоставляет всю необходимую информацию и функциональность, которые пользователь может потребовать в процессе взаимодействия с системой. Оценить полноту возможно путем анализа того, насколько полное и четкое описание функций, возможностей и ограничений системы предоставляется через *пользовательский интерфейс*, в каком объеме пользователю доступна необходимая информация для принятия решений и насколько система может эффективно реагировать на запросы и потребности пользователя.
- Оценку персонализации.

  Это метод, при котором оценивается способность системы адаптироваться к потребностям и предпочтениям каждого пользователя. Оценка персонализации выполняется с помощью анализа пользовательских данных, а

также проведения опросов среди пользователей, чтобы определить, насколько хорошо система учитывает их потребности.

Оценка качества персонализации может включать в себя измерение следующих параметров:
- Работоспособность — действительно ли *пользовательский интерфейс* адаптивный и соответствует ли он потребностям пользователя?
- Уникальность — насколько уникальным и индивидуальным является персонализированный *пользовательский интерфейс*?
- Понятность — насколько просто и легко пользователь может настроить и использовать персонализацию?
- Эффективность — насколько хорошо персонализированный *пользовательский интерфейс* помогает пользователю в выполнении задач?
- Удовлетворенность — насколько положительным и удовлетворительным является *пользовательский интерфейс*?

Оценка персонализации *адаптивных пользовательских интерфейсов* может быть выполнена различными методами, включая тестирование с использованием фокус-групп, опросы пользователей, анализ данных и другие. Важно отметить, что оценка персонализации должна проходить на всех этапах разработки для повышения пользовательского опыта и улучшения качества *пользовательских интерфейсов интеллектуальных систем*.

методы оценки пользовательских интерфейсов можно разделить на две группы: качественные и количественные (см. *Зенг В.А..ОценкКППИНП-2019ст*).

Задача *качественных методов оценки пользовательских интерфейсов* — помочь понять мотивы поведения, потребности и логику пользователей.

качественные методы нацелены на сбор данных, описывающих предмет изучения. Они позволяют углубиться в предметную область для получения представления о мотивации, мышлении и взглядах пользователей.

К *качественным методам оценки пользовательских интерфейсов* относятся:
- *тестирование удобства использования*;
- *отслеживание движения глаз*;
- *экспертная оценка*.

количественные методы оценки пользовательских интерфейсов позволяют выявить сложности или возможности, с которыми сталкиваются пользователи, и отделить реальные проблемы от предполагаемых. Такие методы измеряют числовые показатели.

количественными методы формируют представление о том, чем занимаются пользователи, и включают:
- *А/Б тестирование*;
- *древовидное тестирование*.

тестирование удобства использования пользовательских интерфейсов является важной частью процесса проектирования и разработки. *тестирование удобства использования* гарантирует, что эти интерфейсы удобны и полезны для потенциальных пользователей.

тестирование удобства использования, как правило, включает несколько этапов:
- Определение целевых пользователей. Это может потребовать проведения исследований пользователей для понимания потребностей и предпочтений потенциальных пользователей.
- Разработка тестовых сценариев, которые отражают типичные случаи использования *интерфейса*. Эти сценарии должны быть разработаны для проверки адаптивных возможностей *интерфейса*, а также его удобства использования.
- Проведение тестирования пользователей. Тестирование пользователей включает в себя наблюдение за пользователями во время взаимодействия с *интерфейсом интеллектуальной системы*. Пользователи выполняют задачи, связанные с тестовыми сценариями. Тестирование пользователей может проводиться в контролируемой лабораторной среде или удаленно с использованием технологии совместного просмотра экрана.
- Анализ результатов тестирования. Результаты тестирования пользователей анализируются для выявления проблем с удобством использования и областей для улучшения.
- Итерация дизайна. На основе результатов тестирования пользователей и анализа результатов тестирования, *интерфейс* может быть изменен для устранения проблем с удобством использования и улучшения общего пользовательского опыта.

отслеживание движения глаз — это метод, который используется для анализа того, как пользователи взаимодействуют с *пользовательским интерфейсом*. Отслеживание движения глаз определяет точки фиксации взгляда пользователя при взаимодействии с системой, а также переходы между ними.

При использовании метода определяется *компонент пользовательского интерфейса*, на который пользователь смотрел дольше всего, сколько времени он уделяет каждому компоненту и как легко ему удается найти нужную информацию.

Метод выявляет *компоненты интерфейса*, которым уделяется больше внимания, позволяет обнаружить области, вызывающие у пользователей затруднения.

Метод позволяет получить реалистичный образ отношения пользователя и *интерфейса*, поскольку он фиксирует естественное движение глаз человека. Кроме того, *отслеживание движения глаз* позволяет быстро найти проблемные места в *пользовательском интерфейсе* и предложить улучшения, которые могут увеличить удобство использования *интерфейса*.

Метод *экспертной оценки пользовательских интерфейсов* заключается в исследовании *пользовательского интерфейса* на соответствие заранее определенным правилам (см. *ISOTREoHSI-2002el*).

Достаточно часто метод *экспертной оценки пользовательских интерфейсов* используется в тандеме с *тестированием удобства использования*: *экспертная оценка* используется для формирования гипотез о проблемах, а *тестирование удобства использования* — для их проверки.

Процесс *экспертной оценки пользовательских интерфейсов интеллектуальных систем* обычно включает следующие этапы:

- Выявление экспертов.

  Первый этап — поиск экспертов в предметной области, имеющих опыт работы как с *интеллектуальными системами*, так и с моделью *пользовательского интерфейса*. В число этих экспертов могут входить специалисты по взаимодействию человека с компьютером, специалисты по машинному обучению или эксперты в области, для которой разрабатывается интерфейс.

  Экспертам предоставляется доступ к *пользовательскому интерфейсу* и предлагается взаимодействовать с ним различными способами. Им может быть предложено выполнить задачи или сценарии, которые отражают типичные варианты использования интерфейса.
- Проведение оценки.

  Эксперты оценивают *пользовательский интерфейс* на основе набора установленных принципов удобства использования. Также может быть произведена оценка адаптивности *интерфейса*.
- Анализ результатов.

  Результаты оценки анализируются для выявления проблем с удобством использования и областей для улучшения. Эксперты могут давать рекомендации по улучшению интерфейса на основе своей оценки.

А/Б тестирование пользовательских интерфейсов — метод сравнения двух версий *пользовательского интерфейса*. Результатом проведения метода является выявление версии *интерфейса* наиболее подходящей для выполнения конкретной задачи.

Пользователи случайным образом разбиваются на два сегмента, каждый из которых видит только одну версию интерфейса.

Процесс *А/Б тестирования пользовательских интерфейсов* включает следующие этапы:

- Определение гипотезы.

  В первую очередь необходимо определить цель для проверки. В данном случае это могут быть как отдельные *компоненты пользовательского интерфейса*, так и *пользовательский интерфейс* в целом.

  На основе анализа данных или личных предположений формулируется гипотеза, например, “если мы изменяем цвет и размер кнопки, пользователи будут чаще нажимать на нее” или “если мы добавим кнопку, которая будет видна на каждой странице, пользователи лучше будут понимать, как оставить отзыв”.
- Определение размеров выборок.

  Чтобы получить репрезентативные результаты, нужно определить размер контрольной и экспериментальной групп. Например, 50% пользователей попадут в контрольную группу, которая будет видеть старый интерфейс, а другие 50% в экспериментальную, которая будет видеть измененный интерфейс.
- Итерация интерфейса.

  На этом этапе вносятся изменения в интерфейс, которые соответствуют определенной гипотезе.
- Наблюдение и сбор данных.

  Во время тестирования фиксируются действия пользователей, например, клики и время, проведенное на странице. Эти данные необходимы для определения изменений интерфейса, наиболее повлиявших на поведение пользователей.
- Анализ результатов.

  После того, как тестирование завершено, проводится анализ данных для понимания, насколько значимы различия между контрольной и экспериментальной группами. Например, если пользователи, которые видели измененный интерфейс, кликали на кнопку в 2 раза чаще, чем пользователи, которые видели старый интерфейс,

это говорит о том, что изменения были успешными и их необходимо внедрять в конечный *пользовательский интерфейс*.

древовидное тестирование пользовательских интерфейсов — это метод оценки качества *пользовательских интерфейсов*, который заключается в тестировании древовидной структуры навигации по системе.

Метод помогает определить, насколько эффективно пользователи могут находить нужную информацию и выполнять различные задачи.

В *древовидном тестировании* объектом оценки является древовидная структура навигации по системе. Эта структура представляет собой дерево, в котором корневой элемент является главной страницей, а дочерние элементы — подстраницы или разделы.

Цель метода — проверить, насколько легко пользователи могут навигировать по этой структуре и находить нужную информацию.

Процесс *древовидного тестирования пользовательских интерфейсов* включает следующие этапы:
- На основе структуры навигации создается тестовая среда, которая представляет собой виртуальный *интерфейс*.
- Респондентам предлагается выполнить задание, связанное с поиском нужной информации. Это может быть, например, поиск конкретной страницы или поиск информации по определенному тематическому разделу.
- Респондентам дается доступ к тестовой среде, и они проходят по древовидной структуре навигации, пытаясь найти нужную им информацию.
- В процессе прохождения теста респонденты записывают свои действия и комментарии о том, как они ориентируются в системе, какой путь выбирают для поиска нужной информации, какие ошибки и препятствия им приходится преодолевать и так далее.
- По результатам тестирования делается анализ путей, выбранных респондентами, выявляются наиболее эффективные и неэффективные способы навигации, а также выделяются проблемные зоны *интерфейса* системы.
- На основе анализа результатов тестирования и выявленных проблем делается рекомендация по оптимизации древовидной структуры навигации или изменению *пользовательского интерфейса* для улучшения пользовательского опыта.

Представленные *методы оценки пользовательских интерфейсов* предлагается применять для оценки *пользовательских интерфейсов ostis-систем* в рамках рассмотренного ранее этапа “Анализ пользовательского интерфейса и его адаптация”. При этом особенностью проектирования *пользовательских интерфейсов ostis-систем* является постоянная информационная поддержка пользователя на всех этапах проектирования *интерфейса* за счет наличия в *базе знаний* каждой *ostis-системы Предметной области методик проектирования пользовательских интерфейсов ostis-систем*, содержащей методики, рассмотренные в рамках данного параграфа.

§ 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем

Большое разнообразие *интерфейсов* влечет за собой разработку большого числа компонентов. В качестве *многократно используемых компонентов интерфейсов ostis-систем* могут выступать как уже спроектированные *интерфейсы*, так и специфицированные *компоненты интерфейсов*. Большое число *многократно используемых компонентов интерфейсов ostis-систем* создает проблему их хранения и поиска. Чтобы решить эту проблему, в технологию включена *библиотека многократно используемых компонентов пользовательских интерфейсов ostis-систем* и *менеджер многократно используемых компонентов ostis-систем* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*).

В рамках библиотеки, все компоненты специфицируются и классифицируются. Их классификация производится по различным признакам. К примеру, в зависимости от решаемых задач компоненты делятся на следующие классы:
- просмотрщик содержимого *файлов ostis-системы*;
- редактор содержимого *файлов ostis-системы*;
- транслятор содержимого *файла ostis-системы* в *SC-код*;
- транслятор из *SC-кода* в содержимое *файла ostis-системы*;
- транслятор *базы знаний* во внешнее представление (см. *Корончик Д.Н.СеманТКПП-2011ст* , *Корончик Д.Н.ПользИИМП-2014ст*).

Технология OSTIS позволяет интегрировать в качестве компонентов редакторы и просмотрщики, разработанные с использованием других технологий (далее их будем называть *платформенно-зависимыми многократно используемыми компонентами интерфейса ostis-систем*). В основном они используются для просмотра и редактирования содержимого *файлов ostis-системы*. Это значительно позволяет сэкономить время при их разработке.

пользовательский интерфейс библиотеки многократно используемых компонентов интерфейсов ostis-систем строится на основе sc.g-интерфейса (комплекс информационно-программных средств обеспечивающих общение

интеллектуальных систем с пользователями на основе *SCg-кода*, как способа внешнего представления информации). Однако, это не исключает возможность использования других способов диалога пользователя с *библиотекой многократно используемых компонентов интерфейсов ostis-систем*.

В рамках *библиотеки многократно используемых компонентов интерфейсов ostis-систем* могут содержаться различные версии и модификации какого-либо компонента. К примеру, компонент просмотра *sc.g-конструкций* может иметь модификации, для отображения в которых может использоваться двумерная, трехмерная или же многослойная визуализация, при этом каждая из модификаций компонента может иметь различные версии.

Использование *библиотеки многократно используемых компонентов интерфейсов ostis-систем* при проектировании *интерфейса* прикладной системы позволяет значительно сократить сроки проектирования, а также снизить требования, предъявляемые к начальной квалификации разработчика. Это достигается за счет проектирования *интерфейса* из уже заранее подготовленных моделей интерфейса, что также позволяет повысить качество проектируемого *интерфейса*.

Заключение к Главе 5.4.

В рамках главы был проведен анализ существующих методик проектирования *интерфейсов компьютерных систем*, на основе которого была предложена методика проектирования *интерфейсов ostis-систем*. Для предложенной методики были рассмотрены все необходимые этапы проектирования, включающие:

- анализ пользователя, его задач и целей, а также контекста использования;
- анализ требований к *пользовательскому интерфейсу* и спецификация проектируемого *пользовательского интерфейса*;
- задачно-ориентированная декомпозиция *пользовательского интерфейса*;
- проектирование *пользовательского интерфейса* по умолчанию;
- разработка *пользовательского интерфейса*;
- анализ *пользовательского интерфейса* и его адаптация.

Одной из ключевых особенностей предлагаемой методики является использование *многократно используемых компонентов интерфейсов*, что позволяет существенно сократить время проектирования, а также уменьшить требования к профессиональной квалификации разработчика.

Часть 6.

Платформы реализации интеллектуальных компьютерных систем нового поколения

:= [Часть 6. Аппаратные и программные платформы реализации интеллектуальных компьютерных систем нового поколения]

:= [Часть 6. Платформы для массовой разработки семантически совместимых интероперабельных интеллектуальных компьютерных систем нового поколения]

:= [Часть 6. Платформы для ostis-систем]

⇒ *аннотация**:
[Уточнение понятия платформенной независимости интеллектуальных компьютерных систем нового поколения. Требования, предъявляемые к платформам интерпретации sc-моделей ostis-систем. Принципы и способы реализации аппаратной платформы для ostis-систем (ассоциативного семантического компьютера). Спецификация и существующие аналоги программного варианта реализации платформы для ostis-систем.]

⇒ *подраздел**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Глава 6.3. Программная платформа ostis-систем*

Глава 6.1.

Универсальная модель интерпретации логико-семантических моделей ostis-систем

⇒ *автор**:
- *Шункевич Д.В.*

⇒ *аннотация**:
[В главе рассматривается подход к решению проблемы платформенной независимости компьютерных систем, предполагающий унификацию принципов реализации таких систем и обеспечения их семантической совместимости на основе Технологии OSTIS. Приводится формализованная система понятий, определяющая принципы реализации данного подхода.]

⇒ *подраздел**:
- *§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению*
- *§ 6.1.2. Методы и средства реализации ostis-систем*
- *§ 6.1.3. Уточнение понятия ostis-платформы*

⇒ *ключевое понятие**:
- *sc-машина*
- *ostis-платформа*
- *базовая ostis-платформа*
- *расширенная ostis-платформа*
- *специализированная ostis-платформа*
- *минимальная конфигурация ostis-системы*
- *программный вариант ostis-платформы*
- *ассоциативный семантический компьютер*

⇒ *библиографическая ссылка**:
- *Комарцова Л.Г..Нейро-2004кн*
- *USBA-2022el*
- *Колесников А.В.ГибриИСТиТ-2001кн*
- *Hopcroft ..Intro tATLaC-2000art*
- *Neumann J.FirstDoaRotEDVAC-1993art*
- *Godfrey M.D..tCompu avNPI-1993art*

§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению

В общем случае разработка любой искусственной системы, в частности, *интеллектуальной компьютерной системы*, предполагает выполнение двух этапов:
- этапа проектирования, то есть построения формальной модели системы, достаточной для понимания принципов ее устройства и выполнения последующего этапа ее реализации;
- этапа реализации, то есть непосредственно воплощения разработанной модели с использованием конкретных средств (инструментов, материалов, комплектующих и так далее). В случае компьютерных систем выполнение данного этапа обычно предполагает выбор конкретных языков программирования, библиотек, сторонних средств, таких как с.у.б.д. и различные сервисы, а также собственно программирование и отладку системы с использованием выбранных средств.

Для каждого из указанных этапов могут существовать свои методики, а также средства автоматизации соответствующих процессов.

Если этап проектирования компьютерной системы как правило требует участия высококвалифицированных специалистов и экспертов в предметных областях, в которых осуществляется автоматизация, то этап реализации, с одной стороны, как правило является более простым (при условии качественного выполнения этапа проектирования), а с другой стороны требует значительных ресурсов. Одной из причин этого является необходимость

работы компьютерной системы на различных платформах (устройствах), каждое из которых в общем случае может иметь свои особенности и ограничения, которые необходимо учитывать на этапе реализации. Решением данной проблемы является обеспечение платформенной независимости (или кроссплатформенности) разрабатываемых компьютерных систем.

Сама по себе идея обеспечения платформенной независимости давно и широко используется в современных компьютерных системах. Данная проблема, как правило, рассматривается на двух уровнях:

- проблема обеспечения возможности работы программной системы в разных операционных системах;
- проблема обеспечения совместимости операционной системы с различными аппаратными архитектурами. Для решения этой проблемы могут существовать разные сборки ядра операционной системы для разных аппаратных архитектур, как это делается для операционных систем семейства Linux. При этом следует отметить, что речь в подавляющем большинстве случаев идет не о принципиально разных архитектурах, а о вариантах реализации базовой архитектуры фон Неймана.

В случае, когда разрабатываемая компьютерная система проектируется на более низком уровне, чем операционная система как таковая (например, при программировании контроллеров управления различными устройствами), проблема обеспечения платформенной независимости значительно усугубляется и чаще всего может быть решена только для набора аппаратных средств определенного класса, для которого стандартизируется интерфейс доступа, то есть, по сути, набор низкоуровневых команд обработки информации.

Таким образом, можно сказать, что большее внимание при проектировании современных компьютерных систем на данный момент уделяется первому из перечисленных уровней платформенной независимости, то есть обеспечению работы программной системы на разных операционных системах. Это может достигаться разными путями:

- Использование кроссплатформенных языков программирования, которые, в свою очередь, можно разделить на "полностью" интерпретируемые языки (Python, JavaScript и языки на его основе, PHP и другие) и языки, использующие компиляцию в сохраняющий независимость от платформы низкоуровневый байт-код, с его возможной последующей компиляцией в машинный код непосредственно в процессе исполнения (Just-in-time компиляция или JIT-компиляция). К языкам второго класса относятся, например, Java и C#. Реализация такого подхода требует установки на целевой компьютер с операционной системой интерпретатора соответствующего языка программирования или байт-кода.

  Несмотря на популярность такой вариант имеет ряд ограничений:
  - в среднем производительность интерпретируемых программ ниже, чем компилируемых. Одним из подходов к решению данной проблемы и является JIT-компиляция;
  - строго говоря, кроссплатформенность при таком варианте обеспечивается не для всех операционных систем, а для класса операционных систем и соответствующего класса устройств, например, операционных систем, предназначенных для персональных компьютеров. Так, например, приложение, написанное на языке Java для персонального компьютера не может быть напрямую перенесено на мобильное устройство, поскольку при разработке мобильных приложений учитываются другие принципы работы пользователя с интерфейсом системы, отсутствие многооконности и многое другое.

- Реализация системы в виде web-приложения, работа с которым осуществляется через web-браузер и интерфейс которого, таким образом, реализуется на базе общепринятых стандартов Всемирной паутины (HTML, CSS, JavaScript и языки и библиотеки на его основе). Такой вариант обеспечивает возможность работы с приложением с любого устройства, имеющего web-браузер, в том числе, мобильного. К недостаткам такого варианта относятся:
  - как правило, высокая требовательность к производительности конечного устройства. Современный web-браузер является одним из самых ресурсоемких приложений почти на любом устройстве;
  - остается за кадром проблема обеспечения платформенной независимости серверной части web-приложения, которая должна решаться каким-то другим способом;
  - несмотря на стандартизацию, разработчикам часто приходится учитывать особенности конкретных web-браузеров и тестировать работоспособность приложений для каждого из них;
  - потенциально одним и тем же web-приложением можно пользоваться на любом устройстве, однако для обеспечения удобства и наглядности как правило приходится разрабатывать отдельные версии web-приложения, адаптированные под разные устройства, имеющие, например разные размеры экрана.

- Виртуализация (контейнеризация, эмуляция). Перечисленные термины не являются полностью синонимичными, но в целом обозначают подход, при котором в рамках операционной системы создается некоторое изолированное локальное окружение (виртуальная машина, контейнер, среда эмуляции), содержащее все необходимые для работы приложения настройки и гарантирующее его работу на любых операционных системах и устройствах, где может интерпретироваться соответствующая виртуальная машина или контейнер. Соответственно, запуск таких окружений требует установки на конечное устройство соответствующего интерпретатора или эмулятора.

Данный подход бурно развивается и набирает популярность в настоящее время, поскольку позволяет решить не только проблему кроссплатформенности, но и избавить потребителя от установки большого числа зависимостей и выполнения настройки приложения на конечном устройстве.

Среди популярных средств реализующих данный подход можно указать средства виртуализации (VirtualBox, DosBox, VMWare Workstation), контейнеризации (Docker), эмуляции приложений Android для настольных операционных систем (Genymotion, Bluestacks, Anbox) и многие другие.

К недостаткам такого подхода можно отнести его ресурсоемкость и снижение производительности, а также ограниченность применения (как правило, соответствующие интерпретаторы разрабатываются только для наиболее популярных и востребованных операционных систем). Кроме того, возникает проблема следующего уровня, связанная уже с зависимостью от выбранного средства виртуализации (контейнеризации).

Важно также отметить, что даже для интерпретируемых языков программирования существует проблема зависимости приложения от используемого набора библиотек и фреймворков. Так, при разработке интерфейса web-приложения могут использоваться популярные фреймворки AngularJS и ReactJS, при этом после выбора одного из них быстрый перевод приложения на другой фреймворк невозможен.

Таким образом, можно сделать вывод о том, что проблеме обеспечения платформенной независимости в современных компьютерных системах уделяется достаточно много внимания, однако в полной мере она не решена. В то же время, существует большое количество успешных частных решений, которые, однако, обладают серьезными ограничениями, связанными, в первую очередь, с отсутствием унификации современных подходов к разработке компьютерных систем.

Еще более актуальной проблема обеспечения платформенной независимости становится в контексте разработки *интеллектуальных компьютерных систем*. Это обусловлено следующими особенностями таких систем:

- значительно более сложная по сравнению с традиционными компьютерными системами структура представляемой информации и, соответственно, многообразие форм ее представления, хранение и обработка которых на разных платформах могут быть организованы совершенно по-разному;
- высокие требования к производительности для некоторых классов систем, в частности, систем, использующих машинное обучение, что приводит к созданию специализированных аппаратных архитектур, таких как, например, нейрокомпьютеры (*Комарцова Л.Г..Нейро-2004кн*, *USBA-2022el*);
- многообразие моделей решения задач, которые в общем случае реализуются по-разному в разных системах;
- актуальность разработки гибридных интеллектуальных систем *Колесников А.В.ГибриИСТиТ-2001кн*, в рамках которых интегрируются различные виды знаний и различные модели решения задач. В виду отсутствия на настоящий момент общепринятой унифицированной основы для их интеграции такие системы создаются в основном с ориентацией на какую-то определенную платформу и трудно переносимы на другие платформы.

Таким образом, можно сказать, что проблема обеспечения платформенной независимости для интеллектуальных систем обусловлена во многом отсутствием семантической совместимости компонентов таких систем между собой, что, в свою очередь, создает препятствия даже для реализации подходов к обеспечению платформенной независимости, реализуемых в процессе разработки традиционных компьютерных систем. То есть, для решения проблемы обеспечения платформенной независимости интеллектуальных систем, требуется вначале обеспечить семантическую совместимость компонентов таких систем между собой, что, в свою очередь, предполагает:

- унификацию представления различного рода информации, хранимой в базах знаний таких систем;
- унификацию базовых моделей обработки информации, хранимой в базах знаний таких систем, то есть выделение универсального низкоуровневого языка программирования, позволяющего осуществлять обработку информации, хранимой в унифицированном виде;
- унификацию принципов реализации различных моделей решения задач и, как следствие, возможность их интеграции в рамках гибридных интеллектуальных систем;
- унификацию принципов разработки интерфейсов компьютерных систем, которая бы позволила реализовать в рамках одной интеллектуальной системы возможность взаимодействия с другими системами и пользователями таких систем на разных внешних языках, включая естественные языки.

Указанные принципы реализуются в рамках *Технологии OSTIS*), которая, таким образом, может стать основой для решения проблемы обеспечения семантической совместимости компонентов *интеллектуальных компьютерных систем* в целом и обеспечения платформенной независимости таких систем. С одной стороны, принципы, лежащие в основе *Технологии OSTIS* (см. *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*), обеспечивают принципиальную возможность реализации платформенной независимости компьютерных систем, разрабатываемых на ее основе (*ostis-систем*). С другой стороны, благодаря своей универсальности *Технология OSTIS* позволяет преобразовать любую современную компьютерную систему в *ostis-систему*, которая будет функционально эквивалентна исходной компьютерной системе, но при этом будет обладать всеми перечисленными выше свойствами, создающими предпосылки для решения проблемы платформенной независимости.

Для реализации данного подхода в рамках *Технологии OSTIS* требуется разработать семейство *онтологий*, обеспечивающих уточнение таких понятий, как *ostis-система*, *ostis-платформа*, их структуры, типологии и предъявляемых к ним требований. Рассмотрению указанных онтологий и посвящена данная глава.

Что касается обозначенной выше проблемы зависимости компьютерных систем от конкретных фрейморков, то аналогичная проблема может возникнуть и при дальнейшем развитии *Технологии OSTIS*, в ситуации, когда соответствующие библиотеки будут содержать достаточно большое количество функционально эквивалентных компонентов. Однако, благодаря принципам, лежащим в основе *Технологии OSTIS*, в частности, смысловому представлению информации и семантической совместимости компонентов, данная проблема будет значительно менее острой, поскольку:

- число функционально эквивалентных компонентов будет значительно ниже, чем в традиционных информационных технологиях, нет необходимости создавать синтаксически разные компоненты, отличия будут только на семантическом уровне;
- сами по себе компоненты будут являться более универсальными, то есть смогут быть использованы в значительно большем количестве систем;
- есть возможность автоматически выявить близкие компоненты, их сходства, различия, потенциальные конфликты и зависимости компонентов;
- есть возможность построения достаточно простых (по сравнению с традиционными технологиями) процедур перехода от одного фреймворка к другому, поскольку все компоненты и фреймворки имеют общую формальную смысловую основу, более высокоуровневую, чем в традиционных технологиях.

§ 6.1.2. Методы и средства реализации ostis-систем

⇒ *ключевое понятие**:
- *sc-модель кибернетической системы*
- *ostis-система*
- *ostis-платформа*
- *sc-память*
- *sc-модель базы знаний*
- *sc-модель решателя задач*
- *sc-модель интерфейса кибернетической системы*
- *sc-машина*
- *scp-машина*

Рассмотрим предлагаемый подход к организации реализации *ostis-систем*. Одним из ключевых принципов *Технологии OSTIS* является обеспечение платформенной независимости *ostis-систем*, то есть строгое разделение логико-семантической модели кибернетической системы (*sc-модели кибернетической системы*) и платформы интерпретации sc-моделей кибернетических систем (*ostis-платформы*). Преимущества такого строгого разделения достаточно очевидны:

- Перенос *ostis-системы* с одной платформы на другую (например более новую и эффективную или ориентированную на определенный класс устройств) выполняется с минимальными накладными расходами (в идеальном случае — вообще сводится просто к загрузке *sc-модели кибернетической системы* на платформу);
- Компоненты *ostis-систем* становятся универсальными, то есть могут использоваться в любых ostis-системах, где их использование является целесообразным;
- Развитие платформы и развитие sc-моделей систем может осуществляться параллельно и независимо друг от друга, в общем случае отдельными независимыми коллективами разработчиков по своим собственным правилам и методикам.

Рассмотрим более детально понятие *логико-семантической модели кибернетической системы*.

логико-семантическая модель кибернетической системы

:= [формальная модель (формальное описание) функционирования кибернетической системы, состоящая из (1) формальной модели информации, хранимой в памяти кибернетической системы и (2) формальной модели коллектива агентов, осуществляющих обработку указанной информации.]

⊃ *sc-модель кибернетической системы*

:= [логико-семантическая модель кибернетической системы, представленная в SC-коде]

:= [логико-семантическая модель ostis-системы, которая, в частности, может быть функционально эквивалентной моделью какой-либо кибернетической системы, не являющейся ostis-системой]

кибернетическая система

⊃ *компьютерная система*

:= [искусственная кибернетическая система]
⊃ *ostis-система*
:= [компьютерная система, построенная по Технологии OSTIS на основе интерпретации спроектированной логико-семантической sc-модели этой системы]

ostis-система
⊂ *субъект*
⇒ *обобщенная декомпозиция**:
{• *sc-модель кибернетической системы*
• *ostis-платформа*
}

sc-модель кибернетической системы
⇒ *обобщенная декомпозиция**:
{• *sc-память*
• *sc-модель базы знаний*
• *sc-модель решателя задач*
• *sc-модель интерфейса кибернетической системы*
}

sc-память
:= [абстрактная sc-память]
:= [sc-хранилище]
:= [семантическая память, хранящая конструкции SC-кода]
:= [хранилище конструкций SC-кода]

sc-память представляет собой с одной стороны общую среду для хранения *базы знаний*, а с другой стороны — среду для взаимодействия *sc-агентов*. При этом каждый *sc-агент* опирается при работе на некоторые известные ему *sc-элементы*, хранящиеся в *sc-памяти* (*ключевые sc-элементы* данного *sc-агента*).

В общем случае *sc-память* реализует следующие функции:
- хранение конструкций *SC-кода*;
- хранение внешних по отношению к *SC-коду* информационных конструкций (файлов). В общем случае хранение файлов может быть реализовано отличным от хранения *sc-конструкций* образом;
- доступ (чтение, создание, удаление) к конструкциям *SC-кода*, реализуемый через соответствующий программный или аппаратный интерфейс. Такой интерфейс по сути представляет собой язык микропрограммирования, позволяющий реализовывать на его основе более сложные процедуры обработки хранимых конструкций, в том числе — операторы *Языка SCP*, набор которых по сути определяет перечень команд такого языка микропрограммирования. Сама *sc-память* в этом плане является пассивной и просто выполняет команды, инициируемые извне какими-либо субъектами.

Отметим, что разделение функции хранения и доступа является достаточно условным, поскольку реализовать функцию хранения конструкций без возможности доступа к ним хотя бы на самом низком уровне представляется нецелесообразным, ведь пользоваться таким хранилищем будет невозможно.

Термины "*sc-память*" и "абстрактная sc-память" являются синонимами в том смысле, что говоря об *sc-памяти* мы подразумеваем некоторую абстракцию, для которой не уточняется ее максимальный объем (максимальное количество *sc-элементов*, которые могут одновременно храниться в такой памяти), конкретный способ хранения *sc-элементов*, средства обеспечения надежности хранения и так далее. Все указанные особенности уточняются на уровне *реализации sc-памяти* в аппаратном варианте или варианте программной модели на базе какой-либо другой архитектуры.

Явное выделение *sc-модели базы знаний*, *sc-модели решателя задач* и *sc-модели интерфейса кибернетической системы* в рамках *sc-модели кибернетической системы* является в известной мере условным, поскольку для обеспечения платформенной независимости *sc-модели кибернетической системы* и *решатель задач*, и *интерфейс системы* описываются средствами *SC-кода* и, таким образом, тоже являются частью *базы знаний*. Такое явное выделение указанных компонентов обусловлено удобством проектирования и сопровождения системы.

Таким образом, при условии строгого разделения *sc-модели кибернетической системы* и *ostis-платформы*, а также обеспечении универсальности *ostis-платформы*, то есть возможности интерпретировать любую *sc-модель кибернетической системы* на любом варианте *ostis-платформы*, этап реализации *ostis-системы* фактически сводится к загрузке *sc-модели кибернетической системы* на выбранный вариант *ostis-платформы*.

Важно отметить, что универсальность конкретного варианта реализации *ostis-платформы* очевидно ограничивается физической (аппаратной) частью этой реализации. Например, если аппаратная часть выбранного варианта

платформы представляет собой обычный персональный компьютер, то без добавления дополнительных аппаратных компонентов система не сможет решать задачи, связанные с физическим перемещением себя и других объектов в пространстве, даже если программная часть системы способна выполнить необходимые расчеты. Говоря другими словами, любая *ostis-платформа* всегда будет ограничена в решении *поведенческих задач* каких-либо классов, какими бы мощными физическими ресурсами она не обладала. Таким образом, корректнее говорить об универсальности *ostis-платформы* в контексте решения *информационных задач*, то есть возможности интерпретировать любые *sc-модели кибернетических систем* независимо от того, какого рода *информационные задачи* рещают эти системы.

Исходя из этого можно сформулировать ключевое требование, предъявляемое к *sc-модели кибернетической системы* — ни на одном из этапов решения любой *информационной задачи* в данной системе не должны учитываться особенности той платформы, на которой в дальнейшем будет интерпретироваться указанная *sc-модель*. Аналогично ключевым требованием к *ostis-платформе* является обеспечение интерфейса доступа (поиска и преобразования) к хранимой в *sc-памяти* информации некоторым универсальным способом, не зависящим от особенностей реализации конкретной платформы. Таким образом, важнейшей задачей для обеспечения платформенной независимости *ostis-систем* является четкая спецификация требований, предъявляемых к каждой реализации *ostis-платформы*, то есть стандартизация *ostis-платформ*. Важно отметить, что такая стандартизация не должна зависеть от того, в каком виде реализуется *ostis-платформа*, и, соответственно, подходить и для аппаратного варианта реализации.

Для уточнения требований, предъявляемых к *ostis-платформе*, введем понятие *sc-машины*, которое является аналогом таких моделей как машина Поста и машина Тьюринга (*Hopcroft ..Intro tATLaC-2000art*), машина фон Неймана (*Neumann J.FirstDoaRotEDVAC-1993art*, *Godfrey M.D..tCompu avNPI-1993art*).

sc-машина
- := [абстрактная sc-машина]
- ∈ *абстрактная машина обработки информации*
- := [обобщение всевозможных реализаций ostis-платформ, для которого задаются общие функциональные требования]
- := [обобщенная модель, описывающая функционирование любой ostis-платформы независимо от способа ее реализации]
- := [обобщенная модель, определяющая общие закономерности любой ostis-платформы независимо от способа ее реализации]
- := [обобщенный информационный образ ostis-платформы]
- ⇐ *обобщенная модель**:
  ostis-платформа
- ⇒ *обобщенная декомпозиция**:
  - {• *sc-память*
    - ⇐ *обобщенная модель**:
      реализация sc-памяти
  - • *абстрактная машина обработки знаний*
    - ⊂ *абстрактный sc-агент*
  - }
- ⊃ *scp-машина*
  - ⇐ *обобщенная модель**:
    scp-интерпретатор
  - := [sc-машина, обеспечивающая интерпретацию базового языка программирования ostis-систем]
  - := [обобщенная модель интерпретатора базового языка программирования ostis-систем]
  - := [обобщенная модель, определяющая общие принципы интерпретации базового языка программирования ostis-систем]
  - := [обобщенная модель операционной семантики базового языка программирования ostis-систем]

Потенциально можно говорить о нескольких возможных функционально эквивалентных вариантах *scp-машины*, которые будут соответствовать разным вариантам базового языка программирования. В рамках текущей версии *Технологии OSTIS* фиксируется как денотационная семантика *Языка SCP*, так и его операционная семантика, реализуемая в виде *Абстрактной scp-машины*. Более подробно об этом говорится в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*.

Важно подчеркнуть, что несмотря на преимущества платформенно-независимой реализации *ostis-систем* иногда оказывается целесообразным реализовывать некоторые компоненты *ostis-систем* (например, конкретные *sc-агенты* или компоненты пользовательского интерфейса) на уровне *ostis-платформы*. В случае подобной реализации программ *sc-агентов* можно провести аналогию с реализацией каких-либо подпрограмм на уровне языков микропрограммирования для современных компьютеров. Чаще всего целесообразность такого решения обусловлена повышением производительности таких компонентов и системы в целом, поскольку реализация компонента

с учетом особенностей платформы в общем случае является более производительной. В то же время заметим, что последнее утверждение не всегда верно, поскольку при реализации компонента на уровне логико-семантической модели могут быть реализованы, например, модели параллельной обработки информации, не всегда легко и понятно реализуемые на уровне платформы.

Таким образом, при проектировании каждой конкретной *ostis-системы* разработчику необходимо принимать решение о реализации тех или иных компонентов на платформенно-независимом уровне или уровне платформы. При этом очевидно, что с точки зрения развития технологии и накопления проектного опыта более приоритетной является реализация компонентов *ostis-систем* на платформенно-независимом уровне.

Исходя из сказанного, можно предположить существование *ostis-систем*, в которых все *sc-агенты* реализованы на уровне платформы, которая в таком случае по сути "заточена" под конкретную *ostis-систему* и может рассматриваться как аналог специализированного компьютера, ориентированного на решение задач только определенного ограниченного класса. Назовем такой вариант реализации *ostis-систем минимальной конфигурацией ostis-системы*. Для того, чтобы *минимальная конфигурация ostis-системы* вообще могла считаться *ostis-системой*, то есть системой, построенной в соответствии с принципами *Технологии OSTIS*, она должна удовлетворять следующему минимальному набору требований:

- использование *SC-кода* как базового языка кодирования информации в базе знаний, и, соответственно, наличие памяти, хранящей конструкции *SC-кода*;
- наличие *базы знаний*, определяющей денотационную семантику понятий, используемых системой;
- наличие хотя бы одного *внутреннего sc-агента*, осуществляющего обработку знаний в памяти *ostis-системы*. Этот *sc-агент* может быть реализован на уровне платформы, соответственно *база знаний* такой системы может не содержать процедурных знаний (методов);

Такой вариант *минимальной конфигурации ostis-системы* обладает только *внутренним sc-агентом* и, соответственно, не имеет возможности общаться с внешним миром (можно сказать, что такая *ostis-система* не обладает "органами чувств"). Для того, чтобы система имела возможность общаться с внешним миром, необходимо добавить к *минимальной конфигурации ostis-системы* хотя бы один *рецепторный sc-агент* и хотя бы один *эффекторный sc-агент*.

Важно отметить, что, как видно из представленного описания *минимальной конфигурации ostis-системы*, в общем случае *ostis-система* не обязана по умолчанию быть *интеллектуальной системой*. Применение *Технологии OSTIS* для разработки компьютерных систем не делает их автоматически интеллектуальными, оно позволяет обеспечить возможность последующей неограниченной интеллектуализации таких систем с минимальными накладными расходами при условии соблюдения при их разработке всех принципов *Технологии OSTIS*.

§ 6.1.3. Уточнение понятия ostis-платформы

⇒ *ключевое понятие**:
- *ostis-платформа*
- *базовая ostis-платформа*
- *расширенная ostis-платформа*
- *специализированная ostis-платформа*
- *реализация sc-памяти*
- *реализация файловой памяти sc-машины*
- *scp-интерпретатор*
- *базовая подсистема взаимодействия ostis-системы с внешней средой*
- *подсистема обеспечения жизнедеятельности ostis-системы*
- *специализированная платформенно-зависимая машина обработки знаний*
- *минимальная конфигурация ostis-системы*
- *однопользовательская ostis-платформа*
- *многопользовательская ostis-платформа*
- *программный вариант ostis-платформы*
- *ассоциативный семантический компьютер*

ostis-платформа
:= [платформа интерпретации sc-моделей компьютерных систем]
:= [интерпретатор sc-моделей кибернетических систем]
:= [интерпретатор унифицированных логико-семантических моделей компьютерных систем]
:= [Семейство платформ интерпретации sc-моделей компьютерных систем]
:= [платформа реализации sc-моделей компьютерных систем]
:= ["пустая" ostis-система]

:= [реализация sc-машины]
⊂ *платформенно-зависимый многократно используемый компонент ostis-систем*

Реализация *ostis-платформы* (интерпретатора sc-моделей кибернетических систем) может иметь большое число вариантов — как программно, так и аппаратно реализованных. При необходимости, в ***ostis-платформу*** могут быть заранее на платформенно-зависимом уровне включены какие-либо компоненты решателей задач или баз знаний, например, с целью упрощения создания первой версии *прикладной ostis-системы*. Реализация *ostis-платформы* может осуществляться на основе произвольного набора существующих технологий, включая аппаратную реализацию каких-либо ее частей. С точки зрения компонентного подхода любая ***ostis-платформа*** является ***платформенно-зависимым многократно используемым компонентом ostis-систем***.

ostis-платформа
⇒ *разбиение**:
{• *базовая ostis-платформа*
:= [базовый интерпретатор логико-семантических моделей ostis-систем]
:= [минимальная универсальная ostis-платформа, обеспечивающая интерпретацию sc-модели любой *ostis-системы* и включающая интерпретатор базового языка программирования *ostis-систем* (Языка SCP)]
:= [универсальный интерпретатор sc-моделей ostis-систем]
:= [универсальная базовая ostis-система, обеспечивающая имитацию любой *ostis-системы* путем интерпретации sc-модели имитируемой ostis-системы]
• *расширенная ostis-платформа*
:= [ostis-платформа, содержащая дополнительные компоненты, реализованные на уровне платформы]
:= [базовая ostis-платформа и множество компонентов, реализованных на уровне платформы]
• *специализированная ostis-платформа*
:= [ostis-платформа, не содержащая реализацию интерпретатора Языка SCP]
:= [неуниверсальная ostis-платформа]
}

Понятие *базовой ostis-платформы* является ключевым с точки зрения обеспечения платформенной независимости *ostis-систем*. Универсальность *базовой ostis-платформы* подразумевает возможность интерпретации на ее основе любой *sc-модели кибернетической системы*. Это достигается за счет наличия в рамках *Технологии OSTIS* средств, позволяющих описывать на уровне sc-модели *базу знаний*, *решатель задач* и *интерфейс кибернетической системы*, а также наличия Базового универсального языка программирования для *ostis-систем* (*Языка SCP*). *Язык SCP* в таком случае выступает в роли базового низкоуровневого стандарта (ассемблера) обработки конструкций *SC-кода*, гарантирующего полноту с точки зрения обработки, то есть, обеспечивающего возможность осуществить любое преобразование любого фрагмента *SC-кода* при условии сохранения синтаксической корректности этого фрагмента. Следует отметить, что в общем случае таких функционально эквивалентных ассемблеров может быть несколько (и, как следствие, соответствующих им *scp-машин*), но для обеспечения совместимости в рамках *Технологии OSTIS* один из таких вариантов выбирается в качестве стандарта и описывается в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*.

Таким образом, основным и единственным требованием, предъявляемым ко всем *базовым ostis-платформам* для обеспечения их универсальности, является необходимость обеспечения интерпретации *Языка SCP*, стандартизированного в рамках *Технологии OSTIS*. При этом важно отметить, что все *базовые ostis-платформы* обязаны быть функционально эквивалентными, поскольку интерпретируют один и тот же стандарт *Языка SCP*.

Каждая *базовая ostis-платформа* включает в себя:
- реализацию средств хранения конструкций *SC-кода* (sc-памяти), включая реализацию файловой памяти;
- реализацию средств обработки конструкций *SC-кода — scp-интерпретатора*;
- реализацию базового набора *рецепторных sc-агентов* и *эффекторных sc-агентов*, обеспечивающих минимально необходимый обмен информацией между *ostis-системой* и внешней средой. Конкретный перечень таких агентов требует уточнения, однако можно сказать, что в общем случае они могут быть реализованы как в составе *scp-интерпретатора* (в этом случае им будут соответствовать определенные классы *scp-операторов*), так и отдельно от него в составе платформы.
- реализацию набора sc-агентов, обеспечивающих базовые функции *ostis-системы*, связанные с обеспечением ее жизнедеятельности, которые принципиально не могут быть реализованы на платформенно-независимом уровне. К таким функциям относятся, например, запуск системы, загрузка базы знаний в память системы, запуск *scp-интерпретатора* и так далее.

Более формально модель *базовой ostis-платформы* можно записать следующим образом:

базовая ostis-платформа

⇒ *обобщенная декомпозиция**:
{• *реализация sc-памяти*
⇒ *обобщенная часть**:
реализация файловой памяти sc-машины
• *scp-интерпретатор*
• *базовая подсистема взаимодействия ostis-системы с внешней средой*
• *подсистема обеспечения жизнедеятельности ostis-системы*
}

расширенная ostis-платформа представляет собой *базовую ostis-платформу*, дополненную каким-либо множеством компонентов (хотя бы одним), реализованных на уровне платформы, при условии сохранения при этом всех возможностей *базовой ostis-платформы*. Таким образом, *расширенная ostis-платформа* по сути представляет собой *базовую ostis-платформу*, адаптированную для более эффективного решения задач определенных классов в рамках конкретного класса *ostis-систем*. Компонент, реализуемый на уровне платформы, становится частью этой платформы и, таким образом, преобразует *базовую ostis-платформу* в *расширенную ostis-платформу*.

Введение понятия *расширенной ostis-платформы* позволяет сформулировать ряд дополнительных принципов реализации *ostis-систем*:

- Может существовать произвольное количество ostis-систем, каждая из которых будет иметь свою уникальную *расширенную ostis-платформу*, но при этом все они будут основаны на одном и том же варианте *базовой ostis-платформы*.
- Для каждого варианта *базовой ostis-платформы* может существовать своя *библиотека многократно используемых компонентов ostis-платформ* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*), совместимых с данным вариантом *базовой ostis-платформы*, и позволяющая компоновать различные варианты *расширенной ostis-платформы* на основе *базовой ostis-платформы*.

специализированная ostis-платформа представляет собой ограниченный вариант реализации *ostis-платформы*, не содержащий *scp-интерпретатора*. Таким образом, все *sc-агенты*, в рамках *ostis-системы*, основанной на *специализированной ostis-платформе* должны быть реализованы на платформенно-зависимом уровне. Такая *специализированная ostis-платформа* является аналогом специализированного компьютера, реализованного для конкретной компьютерной системы. Таким образом, в общем случае каждая *ostis-система*, реализуемая на *специализированной ostis-платформе* будет иметь свою уникальную *специализированную ostis-платформу*.

специализированная ostis-платформа может быть получена из *базовой ostis-платформы* путем исключения из нее реализации *scp-интерпретатора* и реализации всех необходимых *sc-агентов* на уровне платформы (или заимствования всех или части агентов из соответствующей данному варианту *базовой ostis-платформы библиотеки многократно используемых компонентов ostis-платформ*).

специализированная ostis-платформа
⇒ *обобщенная декомпозиция**:
{• *реализация sc-памяти*
⇒ *обобщенная часть**:
реализация файловой памяти sc-машины
• *базовая подсистема взаимодействия ostis-системы с внешней средой*
• *подсистема обеспечения жизнедеятельности ostis-системы*
• *специализированная платформенно-зависимая машина обработки знаний*
:= [sc-агент, как правило неатомарный, обеспечивающий выполнение всех функций некоторой специализированной ostis-платформы, связанных с обработкой знаний]
⊂ *платформенно-зависимый sc-агент*
}

Введенное ранее понятие *минимальной конфигурации ostis-системы* может быть уточнено с учетом понятия *специализированной ostis-платформы*.

минимальная конфигурация ostis-системы
⇒ *обобщенная декомпозиция**:
{• *sc-модель базы знаний*
• *специализированная ostis-платформа*
}

Применение *специализированных ostis-платформ* может быть целесообразным на стартовом этапе развития *Технологии OSTIS*, а также с целью повышения производительности конкретных наиболее высоконагруженных *ostis-*

систем, однако активное развитие таких *специализированных ostis-платформ* и их компонентов с точки зрения *Технологии OSTIS* является нецелесообразным, поскольку:

- если какой-либо компонент разработан с ориентацией на конкретную платформу, то нет гарантий возможности его повторного использования в других вариантах реализации *ostis-платформы* (как минимум, компоненты, разработанные для *программного варианта реализации ostis-платформы* не смогут быть использованы в рамках *ассоциативного семантического компьютера*);
- наличие большого числа платформенно-зависимых компонентов требует развития и сопровождения отдельной инфраструктуры библиотек для хранения и повторного использования таких компонентов. Чем больше будет вариантов *ostis-платформ* и чем больше будет число платформенно-зависимых компонентов, тем более сложной и громоздкой будет такая инфраструктура. Как минимум, необходимо будет отслеживать совместимость компонентов с разными версиями разных вариантов реализации *ostis-платформ*;
- изменения в *специализированной ostis-платформе*, например, связанные с переходом на более новую и эффективную версию *базовой ostis-платформы*, на основе которой построена данная *специализированная ostis-платформа* в общем случае могут привести к необходимости внесения изменений в компоненты, зависящие от данного варианта реализации *ostis-платформы*. Чем больше таких платформенно-зависимых компонентов, тем больше потенциальных изменений может потребоваться и, соответственно, тем сложнее будет осуществляться эволюция платформы при условии сохранения работоспособности *ostis-систем*, в которых она используется.

Перечисленные тезисы справедливы и для *расширенных ostis-платформ*, однако в случае *расширенной ostis-платформы* проблемы, связанные с переходом на более новую версию платформы и изменениями в соответствующих компонентах всегда могут быть решены путем временной замены платформенно-зависимых компонентов на их платформенно-независимые версии с соответствующим снижением производительности, но зато сохранением функциональной целостности системы.

ostis-платформа
⇒ *разбиение**:
{• *однопользовательская ostis-платформа*
:= [вариант реализации платформы интерпретации sc-моделей компьютерных систем, рассчитанный на то, что с конкретной *ostis-системой* взаимодействует только один пользователь (владелец)]
• *многопользовательская ostis-платформа*
:= [вариант реализации платформы интерпретации sc-моделей компьютерных систем, рассчитанный на то, что с конкретной *ostis-системой* одновременно или в разное время могут взаимодействовать разные пользователи, в общем случае обладающие разными правами, сферами ответственности, уровнем опыта, и имеющие свою конфиденциальную часть хранимой в базе знаний информации]
}

При однопользовательском варианте реализации платформы оказывается невозможным реализовать некоторые важные принципы *Технологии OSTIS*, например, коллективную согласованную разработку базы знаний системы в процессе ее эксплуатации. При этом могут использоваться различные сторонние средства, например для разработки базы знаний на уровне исходных текстов.

ostis-платформа
⇒ *разбиение**:
{• *программный вариант ostis-платформы*
:= [платформа интерпретации sc-моделей ostis-систем, реализованная в виде программной системы на базе традиционной компьютерной архитектуры]
:= [программная платформа интерпретации sc-моделей ostis-систем]
:= [программный интерпретатор sc-моделей ostis-систем]
• *ассоциативный семантический компьютер*
:= [аппаратная платформа интерпретации sc-моделей ostis-систем]
:= [аппаратно реализованный базовый интерпретатор sc-моделей ostis-систем]
}

Важно отметить, что в любом варианте реализации *ostis-платформы* всегда присутствует как программная, так и аппаратная часть. Так, любой *программный вариант ostis-платформы* предполагает его последующую интерпретацию на какой-либо аппаратной основе, например, на персональном компьютере с традиционной архитектурой. В то же время, разработка *ostis-платформы* в виде *ассоциативного семантического компьютера* предполагает разработку набора микропрограмм, реализующих базовые операции поиска и преобразования sc-конструкций, хранящихся в *sc-памяти*.

Таким образом, разделение множества возможных реализаций *ostis-платформы* на программный и аппаратный варианты скорее отражает вариант аппаратной архитектуры, на которую в конечном итоге ориентирован тот

или иной вариант реализации платформы — либо на традиционную фон-неймановскую архитектуру, либо на специализированную архитектуру *ассоциативного семантического компьютера* со структурно-перестраиваемой (графодинамической) памятью. *Программный вариант ostis-платформы* по сути является моделью (виртуальной машиной) *ассоциативного семантического компьютера*, построенной на базе традиционной фон-неймановской архитектуры, а *Язык SCP* выступает в роли ассемблера для *ассоциативного семантического компьютера* и также может интерпретироваться как в рамках аппаратной реализации такого компьютера, так и в рамках его программной модели.

Целесообразность разработки *программных вариантов ostis-платформы* на настоящий момент обусловлена очевидной распространенностью фон-неймановской архитектуры и, соответственно, необходимостью реализации *ostis-систем* на современных компьютерах различного вида. В то же время очевидно, что разработка специализированных *ассоциативных семантических компьютеров* позволит существенно повысить эффективность работы *ostis-систем*, а четкое разделение *sc-модели кибернетической системы* и платформы ее интерпретации позволит осуществить перевод уже работающих *ostis-систем* с традиционных архитектур на *ассоциативные семантические компьютеры* с минимальными накладными расходами.

Каждой конкретной *ostis-системе* однозначно соответствует конкретная *ostis-платформа*, которая может относиться к разному набору классов *ostis-платформ*. В то же время очевидно, что на этапе разработки платформы проектируется и реализуется некоторый вариант *ostis-платформы*, который затем тиражируется в разные *ostis-системы*. Впоследствии в каждой *ostis-системе* в этот вариант *ostis-платформы* могут быть внесены изменения, но в общем случае в большом количестве *ostis-систем* могут использоваться полностью эквивалентные *ostis-платформы*. Таким образом, целесообразно говорить о *типовых ostis-платформах*, которые:

- являются объектом разработки для разработчиков *ostis-платформ*;
- *являются многократно используемым компонентом ostis-систем* и специфицируются в рамках соответствующих библиотек;
- являются образцом для тиражирования (копирования) при создании новых *ostis-систем*.

Заключение к Главе 6.1.

В данной главе рассмотрены современные проблемы в области обеспечения платформенной независимости *компьютерных систем* в целом, а также особенности обеспечения платформенной независимости *интеллектуальных компьютерных систем*. Предложен подход к решению указанных проблем путем преобразования современных компьютерных систем в *ostis-системы*, для которых решение данной проблемы значительно упрощается благодаря принципам, лежащим в основе *Технологии OSTIS*.

Детально рассмотрены принципы обеспечения платформенной независимости ostis-систем, уточнено понятие ostis-платформы, их классификация и архитектура.

Глава 6.2.
Ассоциативные семантические компьютеры для ostis-систем

⇒ *автор**:
- *Голенков В. В.*
- *Шункевич Д. В.*
- *Гулякина Н. А.*
- *Ивашенко В. П.*
- *Захарьев В. А.*

⇒ *аннотация**:
[В главе рассмотрены принципы реализации аппаратной платформы для реализации систем, построенных на основе Технологии OSTIS, — ассоциативного семантического компьютера.]

⇒ *подраздел**:
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*
- *§ 6.2.3. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для ostis-систем*
- *§ 6.2.4. Архитектура ассоциативных семантических компьютеров для ostis-систем*

⇒ *ключевое понятие**:
- *машина фон-Неймана*
- *архитектура вычислительной системы*
- *ассоциативный семантический компьютер*
- *scp-компьютер*
- *процессорный модуль*
- *накопительный модуль*
- *терминальный модуль*
- *процессорный элемент*
- *физический канал связи*
- *логический канал связи*
- *волновая микропрограмма*
- *волновой язык программирования*

⇒ *библиографическая ссылка**:
- *Neumann J.FirstDoaRotEDVAC-1993art*
- *Godfrey M.D..tCompu avNPI-1993art*
- *Глушков В.М.РекурМиВТ-1974кн*
- *Айлиф Дж.ПринцПБМ-1973кн*
- *Moldovan D..SNAPPPAtAI-1992art*
- *Chu Y.Evolu oCMS-1976art*
- *Калиниченко Л.А..МашинБДиЗ-1990кн*
- *Мартин Дж.ОрганБДвВС-1980кн*
- *Озкарахан Э.МашинБДиУБД-1989кн*
- *Кохонен Т.АссоцП-1980кн*
- *Игнатущенко В.В.кПостаЗПЭВСнОАМОИ-1981ст*
- *Беркович С.Я..оЭффекПАПвВП-1975ст*
- *Айзерман М.А..ДинамПкАСОГОГ-1977ст*
- *Марчук Г.И.МодулАРС-1978кн*
- *Прангишвили И.В..СовреСПСЭВМсНСиАУПД-1981ст*
- *Затуливер Ю.С..ВопроПиМРЯСППсУПД-1981art*
- *Ackerman W.B.DataFL-1979art*
- *Майерс Г.АрхитСЭВМ-1985кн*
- *Глушков В.М.ФундаИиТП-1980ст*
- *Глушков В.М..оРазвитСМЭВМсИЯВУ-1978ст*
- *Рабинович З.Л.оКонцеМИ-1995ст*

- *Задыхайло И.Б.ПроекАППнЦМДОнПРБД-1979кн*
- *Schuster S.A..RAP2aAPfDaIA-1979art*
- *Суворов Е.В..ПроцесБЗ-1985ст*
- *Brukle H.J.HighLLOHaPVNE-1978art*
- *Chu Y.Archi oaHDI-1977art*
- *Кохонен Т.АссоцЗУ-1982кн*
- *Фостер К.АссоцПП-1981кн*
- *Ершов А.П.АлгорМОиАМВС-1982кн*
- *Берштейн Л.С..ОднорПСдРКЛЗнГиГ-1975ст*
- *Васильев В.В..ЭлектМЗнГ-1987кн*
- *Сапатый П.С.АктивИПкМСРЗнГиС-1984ст*
- *Попов А.Ю.ПримеГВССнНК-2019ст*
- *Попов А.Ю.ПринцОГВССнН-2020ст*
- *Jialiang Z..Boost tPoFPGABGPUHMCaCfBFS-2017art*
- *Hu Y..GraphAGLAoHBMEFPGAs-2021art*
- *Song W..NovelGPAPSaR-2016art*
- *Afanasyev I..VGLaHPGPFftSXATVA-2021art*
- *Вайнцвайг М.Н.ОбучаСИИсАПП-1980кн*
- *Вайнцвайг М.Н..МеханМиМЕРвРВ-1987ст*
- *Somsubhra S.ReconSP-2006el*
- *Рабинович З.Л.НекотБПкСМЦМ-1979ст*
- *Рабинович З.Л.РазвиСУЭВМвСыПАНИ-1979ст*
- *Гладун В.П.ЭврисПвСС-1977кн*
- *Гладун В.П.ПланиР-1987кн*
- *Амосов Н.М..Автом иРП-1973кн*
- *Золотов Е.В..РасшиСАД-1982кн*
- *Галушкин А.СовреНРНТвР-1997ст*
- *Хехт-Нильсен Р.НейроИСП-1998ст*
- *Комарцова Л.Г..Нейро-2004кн*
- *USBA-2022el*
- *Moussa M..ArchiSaMfANNI-2013el*
- *НейроПА-2023эл*
- *Allen J..ParalMAfPRS-1989el*
- *CUDAT-2023el*
- *OpenCL-2023el*
- *Tran H-N.. aSurve oGPoGPU-2018art*
- *Shi X..GraphPoGPUs-2018art*
- *Lu Y..Graph-2021art*
- *Голенков В.В.СтрукОиПИвЭММУПСД-1996дс*
- *Голенков В.В.ПаралГКОнРЗИИиеП-1994кн*
- *Голенков В.В.ТермиМПГКИсПиПМСЛОМпМОЭВМ-1994кн*
- *Голенков В.В.ГрафоМиМПА-1996дс*
- *Гапонов П.А.Модел иМПАП-2000дс*
- *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*
- *Сердюков Р.Е.БазовАиИСО-2004дс*
- *Ivashenko V.P.Appli oaIPfOMBPSUaUSKR-2021art*
- *Ивашенко В.П.ПринцПНиПРО-2016ст*
- *Ивашенко В.П.ПредсССиАиОиСОнВССмП-2015ст*
- *Rasheed B..NetwoGDUDI-2019art*
- *Dubrovin E..GraphRMftDMI-2022art*
- *Wolfram S.NewKoS-2002bk*
- *фон Нейман Д.ТеориСА-1971кн*
- *Moon D.A.SymboA-1987art*
- *Smith S.tLMI LTS-1984art*
- *Steele G.L..ConneMLFP-1986art*
- *McJones P.ParalLCMS-2015art*
- *van der Leun V.Intro tJVML-2017bk*
- *Ivashenko V.P.StrinPMfKDS-2020art*
- *Hewitt C.MiddHoLPRPPatJFGP-2009el*
- *Ивашенко В.П.МоделРЗвИС-2020кн*
- *Ивашенко В.П.ОпераУМДвЛАП-2016ст*
- *Ивашенко В.П.РаспрПвНЛАП-2019ст*

- *Ivashenko V.P.GenerPSRLaSS-2022art*
- *Ивашенко В.П.ИсслеПРПРСТМО-2020ст*
- *Ivashenko V.P..SemanLoREiaFBTM-2021art*
- *LegUpHLS-2023el*
- *VHDPl-2023el*
- *SysteCCP-2023el*
- *MyHDL-2023el*
- *Сапатый П.С.ЯзыкВкОНС-1986ст*
- *Moldovan D.I..SNAP aVLSIAfAIP-1985art*
- *Летичевский А.А..ИнсерП-2003ст*
- *Летичевский А.А..ИнсерМ-2012ст*
- *Backus J.CanPBLftNS-1978art*
- *Котов В.Е..АсинхВПнОП-1966ст*

Введение в Главу 6.2.

Применение для разработки *ostis-систем* современных программно-аппаратных платформ, ориентированных на адресный доступ к хранящимся в памяти данным, не всегда оказывается эффективным, поскольку при разработке интеллектуальных систем фактически приходится моделировать нелинейную память на базе линейной. Повышение эффективности решения задач интеллектуальными системами требует разработки специализированных платформ, в том числе аппаратных, ориентированных на унифицированные семантические модели представления и обработки информации. Таким образом, основной целью создания *ассоциативных семантических компьютеров* является повышение производительности ostis-систем.

§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем

Подавляющее большинство современных программно-аппаратных платформ, применяемых при разработке современных компьютерных систем, и, в частности, интеллектуальных компьютерных систем, основаны на принципах абстрактной *машины фон-Неймана* или “архитектуры фон Неймана” (см. *Neumann J.FirstDoaRotEDVAC-1993art*, *Godfrey M.D..tCompi avNPI-1993art*). Рассмотрим основные принципы, лежащие в основе *машины фон-Неймана*.

машина фон-Неймана
- ≔ [абстрактная машина фон-Неймана]
- ∈ *абстрактная машина обработки информации*
- ⇒ *принципы, лежащие в основе**:
  - ⟨• [Информация в памяти представляется в виде последовательности строк символов в бинарном алфавите (“0” или “1”).]
  - • [Память машины представляет собой последовательность адресуемых ячеек памяти.]
  - • [В каждую ячейку может быть записана любая строка символов в бинарном алфавите. При этом длина строк для всех адресуемых ячеек одинакова (в текущем стандарте ячеек, называемых байтами, равна 8 бит).]
  - • [Каждой ячейке памяти взаимно однозначно соответствует битовая строка, обозначающая эту ячейку и являющаяся ее адресом.]
  - • [Каждому типу элементарных действий (операций), выполняемых в памяти машины фон-Неймана, взаимно однозначно ставится ее идентификатор, который в памяти представляется также в виде битовой строки.]
  - • [каждая конкретная операция (команда), выполняемая в памяти, представляется (специфицируется) в памяти в виде строки, состоящей
    - из кода соответствущего типа операции;
    - из последовательности адресов фрагментов памяти, в которых находятся операнды, над которыми выполняются операции — исходные аргументы и результаты. Любой такой фрагмент задается адресом первого байта и количеством байт. Количество операндов однозначно задается кодом типа операции;

    ]
  - • [Программа, выполняемая в памяти, хранится в памяти в виде последовательности спецификаций конкретных операций (команд).]

- [Таким образом, и обрабатываемые данные, и программы для обработки этих данных хранятся в одной и той же памяти (в отличие, например, от Гарвардской архитектуры) и кодируются одинаковым образом.]

⟩

Рассмотрим более детально особенности логической организации традиционной (фон-Неймановской) архитектуры компьютерных систем, существенно затрудняющие эффективную реализацию *ostis-систем* на ее основе:

- последовательная обработка, ограничивающая эффективность компьютеров физическими возможностями элементной базы;
- низкий уровень доступа к памяти, то есть сложность и громоздкость выполнения процедуры ассоциативного поиска нужного фрагмента знаний;
- линейная организация памяти и чрезвычайно простой вид конструктивных объектов, непосредственно хранимых в памяти. Это приводит к тому, что в интеллектуальных системах, построенных на базе современных компьютеров, манипулирование знаниями осуществляется с большим трудом. Во-первых, приходится оперировать не самими структурами, а их громоздкими линейными представлениями (списками, матрицами смежности, матрицами инцидентности); во-вторых, линеаризация сложных структур разрушает локальность их преобразований;
- представление информации в памяти современных компьютеров имеет уровень весьма далекий от смыслового, что делает переработку знаний довольно громоздкой, требующей учета большого количества деталей, касающейся не смысла перерабатываемой информации, а способа ее представления в памяти;
- в современных компьютерах имеет место весьма низкий уровень аппаратно реализуемых операций над нечисловыми данными и полностью отсутствует аппаратная поддержка логических операций над фрагментами знаний, имеющих сложную структуру, что делает манипулирование такими фрагментами весьма сложным.

Попытки преодоления ограничений традиционных фон-Неймановских ЭВМ привели к возникновению множества подходов, связанных с отдельными изменениями принципов логической организации ЭВМ, прежде всего в зависимости от классов задач и предметных областей, на которые ориентируется тот или иной класс ЭВМ. Все эти тенденции, рассмотренные в совокупности, позволяют очертить некоторые ключевые принципы логической организации ЭВМ, ориентированных на переработку знаний (машин переработки знаний — МПЗ). Перечислим основные из этих тенденций:

- Переход к нелинейной организации памяти (см. *Глушков В.М.РекурМиВТ-1974кн*, *Айлиф Дж.ПринцПБМ-1973кн*) и аппаратная интерпретация сложных структур данных (см. *Moldovan D..SNAPPPAtAI-1992art*, *Chu Y.Evolu oCMS-1976art*, *Калиниченко Л.А..МашинБДиЗ-1990кн*).
- Аппаратная реализация ассоциативного доступа к информации (см. *Мартин Дж.ОрганБДвВС-1980кн* , *Озкарахан Э.МашинБДиУБД-1989кн* , *Глушков В.М.РекурМиВТ-1974кн* , *Кохонен Т.АссоцП-1980кн* , *Игнатущенко В.В.кПостаЗПЭВСнОАМОИ-1981ст* , *Беркович С.Я..оЭффекПАПвВП-1975ст* , *Айзерман М.А..ДинамПкАСОГОГ-1977ст*).
- Реализация параллельных асинхронных процессов над памятью (см. *Глушков В.М.РекурМиВТ-1974кн*, *Марчук Г.И.МодулАРС-1978кн*) и, в частности, разработка вычислительных машин, управляемых потоком данных (см. *Прангишвили И.В..СовреСПСЭВМсНСиАУПД-1981ст* , *Затуливер Ю.С..ВопроПиМРЯСППсУПД-1981art* , *Ackerman W.B.DataFL-1979art*, *Майерс Г.АрхитСЭВМ-1985кн*).
- Аппаратная интерпретация языков высокого уровня (см. *Глушков В.М.ФундаИиТП-1980ст* , *Глушков В.М..оРазвитСМЭВМсИЯВУ-1978ст*, *Рабинович З.Л.оКонцеМИ-1995ст*).
- Разработка аппаратных средств ведения баз данных (процессоров баз данных) (см. *Задыхайло И.Б.ПроекАППнЦМДОнПРБД-1979кн*, *Schuster S.A..RAP2aAPfDaIA-1979art*, *Суворов Е.В..ПроцесБЗ-1985ст*).

На пересечении этих тенденций в разное время возникали различные классы вычислительных устройств. Перечислим некоторые из них:

- машины с аппаратной интерпретацией сложных структур данных (см. *Brukle H.J.HighLLOHaPVNE-1978art*, *Chu Y.Evolu oCMS-1976art*, *Chu Y.Archi oaHDI-1977art*);
- машины с развитой ассоциативной памятью (см. *Кохонен Т.АссоцП-1980кн* , *Кохонен Т.АссоцЗУ-1982кн* , *Фостер К.АссоцПП-1981кн*);
- ассоциативные параллельные матричные процессоры (см. *Ершов А.П.АлгорМОиАМВС-1982кн*);
- однородные параллельные структуры для решения комбинаторно-логических задач на графах и гиперграфах (см. *Берштейн Л.С..ОднорПСдРКЛЗнГиГ-1975ст*);
- всевозможные устройства переработки графов (см. *Васильев В.В..ЭлектМЗнГ-1987кн*, *Сапатый П.С.АктивИПкМСРЗнГиС-1984ст*,*Попов А.Ю.ПримеГВССсНК-2019ст*,*Попов А.Ю.ПринцОГВССсН-2020ст*), в частности, на основе FPGA (см. *Jialiang Z..Boost tPoFPGABGPUHMCaCfBFS-2017art* , *Hu Y..GraphAGLAoHBMEFPGAs-2021art*,*Song W..NovelGPAPSaR-2016art*) и векторных процессоров (см. *Afanasyev I..VGLaHPGPFftSXATVA-2021art*);
- системы, осуществляющие переработку информации непосредственно в памяти путем равномерного распределения функциональных средств по памяти и, в частности, предложенная М. Н. Вайнцвайгом процессоро-

память, ориентированная на решение задач Искусственного интеллекта (см. *Вайнцвайг М.Н.ОбучаСИИсАПП-1980кн*, *Вайнцвайг М.Н..МеханМиМЕРвРВ-1987ст*);

- машины, управляемые потоком данных (см. *Ершов А.П.АлгорМОиАМВС-1982кн*, *Прангишвили И.В..СовреСПСЭВМсНСиАУПД-1981ст*, *Майерс Г.АрхитСЭВМ-1985кн*), и, в частности, процессоры, реконфигурируемые с учетом семантики входного потока данных (см. *Somsubhra S.ReconSP-2006el*);
- рекурсивные вычислительные машины (см. *Глушков В.М.РекурМиВТ-1974кн*);
- процессоры реляционных баз данных (см. *Задыхайло И.Б.ПроекАППнЦМДОнПРБД-1979кн* , *Майерс Г.АрхитСЭВМ-1985кн*, *Озкарахан Э.МашинБДиУБД-1989кн*);
- вычислительные машины со структурно-перестраиваемой памятью (см. *Рабинович З.Л.НекотБПкСМЦМ-1979ст*,*Рабинович З.Л.РазвиСУЭВМвСыПАНИ-1979ст*,*Гладун В.П.ЭврисПвСС-1977кн*,*Гладун В.П.ПланиР-1987кн*);
- активные семантические сети (М-сети) (см. *Амосов Н.М..Автом иРП-1973кн*);
- ассоциативные однородные среды (см. *Золотов Е.В..РасшиСАД-1982кн*);
- нейроподобные структуры (см. *Галушкин А.СовреНРНТвР-1997ст*, *Хехт-Нильсен Р.НейроИСП-1998ст*). В последние годы активное развитие теории искусственных нейронных сетей привело к развитию различных подходов к построению высокопроизводительных компьютеров, предназначенных для обучения и интерпретации искусственных нейронных сетей (см. *Комарцова Л.Г..Нейро-2004кн*, *USBA-2022el*, *Moussa M..ArchiSaMfANNI-2013el*) и их внедрению в различные программно-аппаратные комплексы. В отдельное направление выделены так называемые нейроморфные процессоры (см. *НейроПА-2023эл*), отличающиеся высокой производительностью и низким уровнем энергопотребления.
- машины для интерпретации логических правил (см. *Allen J..ParalMAfPRS-1989el*).

Кроме того, развитие графических процессоров (graphics processing unit, GPU) привело к возможности организации параллельных вычислений непосредственно на GPU, для чего разрабатываются специализированные программно-аппаратные архитектуры, например CUDA (см. *CUDAT-2023el*) и OpenCL (см. *OpenCL-2023el*). Преимуществом GPU в данном случае выступает наличие в рамках одного GPU большого (по сравнению с центральным процессором) числа ядер, что позволяет эффективно решать на такой архитектуре задачи, обладающие естественным параллелизмом (например, операции с матрицами). Развиваются также работы, посвященные принципам обработки графовых структур на GPU (см. *Tran H-N.. aSurve oGPoGPU-2018art* , *Shi X..GraphPoGPUs-2018art* , *Lu Y..Graph-2021art*).

В целом можно сказать, что благодаря повышению производительности современных ЭВМ число разработок специализированных аппаратных решений в последние десятилетия снизилось, поскольку многие сложные вычислительные задачи в настоящее время за приемлемое время могут быть решены и на традиционных универсальных архитектурах. Как показано выше, исключение составляют в основном специализированные компьютеры для обработки искусственных нейронных сетей и других графовых моделей, что обусловлено большой востребованностью таких моделей и их сложностью.

В то же время, большинство перечисленных подходов (даже если они достаточно далеко отходят от предложенных фон Нейманом базовых принципов организации вычислительных машин) неявно сохраняют точку зрения на компьютер как на большой арифмометр и тем самым сохраняют ее ориентацию на задачи числового характера. Работы же направленные на разработку аппаратных архитектур, предназначенных для обработки информации, представленной в более сложных формах, чем в традиционных архитектурах не получили широкого распространения и применения, по причине, во-первых, частности предлагаемых решений, и во-вторых из-за отсутствия общего универсального и унифицированного языка кодирования любой информации, в роли которого в рамках *Технологии OSTIS* выступает *SC-код*, а также соответствующей апробированной технологии разработки программных систем для таких аппаратных архитектур. Таким образом, зачастую разработчики подобных архитектур сталкиваются необходимостью разработки специализированного программного обеспечения для этих архитектур, что в конечном итоге приводит к сильному ограничению сфер применения таких архитектур, поскольку их применение оказывается целесообразным только в случае, если трудоемкость разработки специализированного программного обеспечения оправдывает себя с учетом низкой эффективности решения соответствующих задач на более традиционных архитектурах.

SC-код, являющийся формальной основой *Технологии OSTIS* изначально разрабатывался как язык кодирования информации в памяти *ассоциативных семантических компьютеров*, таким образом в нем изначально заложены такие принципы, как универсальность (возможность представить знания любого рода) и унификация (единообразие) представления, а также минимизация *Алфавита SC-кода*, которая, в свою очередь, позволяет облегчить создание аппаратной платформы, позволяющей хранить и обрабатывать тексты *SC-кода*.

Основная методологическая особенность предлагаемого подхода к разработке средств аппаратной реализации поддержки интеллектуальных систем заключается в том, что такие средства должны разрабатываться не до, а после того, как основные положения соответствующей технологии проектирования и эксплуатации интеллектуальных систем будут апробированы на современных технических средствах. Более того, в рамках *Технологии OSTIS*

четко продумана методика перехода на новые аппаратные средства, которая затрагивает только самый нижний уровень технологии — уровень реализации базовой машины обработки семантических сетей (интерпретатора *Языка SCP*).

Проект *ассоциативного семантического компьютера* имеет давнюю историю, основными этапами которой являются следующие:

- 1984 год — в Московском институте электронной техники В.В. Голенковым защищена кандидатского диссертация на тему "Структурная организация и переработка информации в электронных математических машинах, управляемых потоком сложноструктурированных данных", в которой были сформулированы и рассмотрены основные принципы семантических ассоциативных компьютеров (см. *Голенков В.В.СтрукОиПИвЭММУПСД-1996дс*);
- 1993 год — комиссия Госкомпрома провела успешные испытания прототипа *ассоциативного семантического компьютера*, разработанного на базе транспьютеров в рамках научно-исследовательского проекта "Параллельная графовая вычислительная система, ориентированная на решение задач искусственного интеллекта" (см. *Голенков В.В.ПаралГКОнРЗИИиеП-1994кн*, *Голенков В.В.ТермиМПГКИсПиПМСЛОМпМОЭВМ-1994кн*);
- 1996 год — В.В. Голенковым защищена докторская диссертация на тему "Графодинамические модели и методы параллельной асинхронной переработки информации в интеллектуальных системах" (см. *Голенков В.В.ГрафоМиМПА-1996дс*).
- 2000 год — в Институте проблем управления РАН П.А. Гапоновым защищена кандидатская диссертация на тему "Модели и методы параллельной асинхронной переработки информации в графодинамической ассоциативной памяти" (см. *Гапонов П.А.Модел иМПАП-2000дс*);
- 2000 год — в Институте программных систем РАН В.М. Кузьмицким защищена кандидатская диссертация на тему "Принципы построения графодинамического параллельного компьютера, ориентированного на решение задач искусственного интеллекта" (см. *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*);
- 2004 год — в Белорусском государственном университете информатики и радиоэлектроники Р.Е. Сердюковым защищена кандидатская диссертация на тему "Базовые алгоритмы и инструментальные средства обработки информации в графодинамических ассоциативных машинах", в которой было рассмотрено базовое программное обеспечение семантических ассоциативных компьютеров (см. *Сердюков Р.Е.БазовАиИСО-2004дс*).

В тоже время, несмотря на наличие действующего прототипа *ассоциативного семантического компьютера* на базе транспьютеров, основное внимание в рамках соответствующего проекта и других перечисленных работ уделялось принципам организации распределенной параллельной обработки конструкций SC-кода, в частности, был разработан *SCD-код* (Semantic Code Distributed) для распределенного хранения конструкций SC-кода и *Язык SCPD* для распределенной параллельной их обработки. Однако, общие принципы хранения информации и общая архитектура каждого из процессорных элементов (транспьютеров) оставались фон-Неймановскими. В частности, для кодирования конструкций SC-кода в традиционной адресной памяти были разработаны соответствующие структуры данных, близкие к тем, что описаны в *6.3. Программная платформа ostis-систем*.

Таким образом, можно сказать, что обоснованность и необходимость разработки *ассоциативного семантического компьютера*, а также компетентность авторов в данной области подтверждается более чем 30-летним опытом работы и рядом успешных проектов в данном направлении, однако, в тоже время, в предшествующих работах в полной мере не устранены все недостатки фон-Неймановской архитектуры, рассмотренные выше, и разработка и реализация проекта *ассоциативного семантического компьютера*, устраняющего перечисленные недостатки, остается актуальной.

§ 6.2.2. Анализ существующих архитектур вычислительных систем

Как показано в предыдущем параграфе, для того, чтобы преодолеть недостатки существующих архитектур вычислительных систем, включая фон-Неймановскую, было предложено множество различных подходов (см. *Ivashenko V.P.Appli oaIPfOMBPSUaUSKR-2021art*, *Ивашенко В.П.ПринцПНиПРО-2016ст*, *Ивашенко В.П.ПредсССиАиОиСОнВССсМП-2015ст*, *Rasheed B..NetwoGDUDI-2019art*, *Dubrovin E..GraphRMftDMI-2022art*, *Wolfram S.NewKoS-2002bk*). При разработке новых архитектур и, в частности, архитектуры *ассоциативного семантического компьютера*, целесообразно в виде соответствующей онтологии выделить основные признаки классификации и соответствующие им классы (виды) архитектур вычислительных систем. Рассмотрим фрагмент такой онтологии, разработанных на основании анализа существующих решений и выявленных по его результатам подходов.

архитектура вычислительной системы

⇒ *разбиение**:

{• *архитектура вычислительной системы с глобальной оперативной памятью*

⇒ *примечание**:
[Архитектура, в которой все узлы (процессоры или машины) имеют доступ к глобальной оперативной памяти.]
⇒ *разбиение**:
{• *архитектура вычислительной системы с глобальной оперативной памятью данных*
• *архитектура вычислительной системы с глобальной оперативной памятью программ*
• *архитектура вычислительной системы с глобальной оперативной памятью программ и данных*
⇒ *примечание**:
[Примером такой архитектуры вычислительной системы является архитектура фон Неймана.]
}
• *архитектура вычислительной системы без глобальной оперативной памяти*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы с единственной глобальной внутренней памятью*
• *архитектура вычислительной системы со множественной глобальной внутренней памятью*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы со структурно перестраиваемыми межпроцессорными связями*
• *архитектура вычислительной системы без структурно перестраиваемых междпроцессорных связей*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы со структурно становящейся памятью*
• *архитектура вычислительной системы без структурно становящейся памяти*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы с ассоциативным доступом к глобальной (внутренней) памяти*
⇒ *примечание**:
[Ассоциативный характер доступа важен в системах, ориентированных на хранение данных со сложной структурой и ориентированных на масштабируемые (в том числе локальные) механизмы обработки информации.]
• *архитектура вычислительной системы без ассоциативного доступа к глобальной (внутренней) памяти*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы с адресным доступом к глобальной памяти с линейным адресным пространством*
⇒ *примечание**:
[Примерами таких архитектур является большинство используемых на настоящий момент, включая архитектуру фон Неймана. Задачи управления устройствами и данными для таких архитектур рассмотрены в работе *фон Нейман Д.ТеориСА-1971кн*.]
• *архитектура вычислительной системы без адресного доступа к глобальной памяти с линейным адресным пространством*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы с системой команд регистровой обработки данных*
⇒ *примечание**:
[Большинство используемых на настоящий момент архитектур являются примерами архитектур данного класса, включая архитектуру фон Неймана. Архитектуры с системой команд регистровой обработки данных удобны для задач управления данными как для систем обработки образов в задачах пользовательского интерфейса, так и для задач машинного обучения на основе аппарата линейной алгебры.]
• *архитектура вычислительной системы без системы команд с регистровой обработкой данных*
}
⇒ *разбиение**:
{• *архитектура вычислительной системы с системой команд стековой обработки данных*
⇒ *примечание**:

[Примерами применения такой архитектуры являются LISP-машины (см. *Moon D.A.SymboA-1987art* , *Smith S.tLMI LTS-1984art*, *Steele G.L..ConneMLFP-1986art*), другие примеры можно найти также в работах *McJones P.ParalLCMS-2015art*, *van der Leun V.Intro tJVML-2017bk*.]

- *архитектура вычислительной системы без системы команд стековой обработки данных*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с поддержкой системы команд обработки обобщенных строк*

⇒ *примечание**:

[Примером такой архитектуры является архитектура вычислительной системы с поддержкой системы команд обработки списков и обобщенных строк (см. *Ivashenko V.P.StrinPMfKDS-2020art*). Эта модель позволяет эффективно производить операции не только над строками и списками, но и работать с отношениями вида “ключ-значение” с целью их интеграции в системы, управляемые знаниями. Программная реализация этой модели использует B-деревья.]

- *архитектура вычислительной системы без системы поддержки команд обработки обобщенных строк*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с поддержкой системы команд обработки графовых структур*

⇒ *примечание**:

[Примером такой архитектуры является архитектура компьютера Leonhard (см. *Rasheed B..NetwoGDUDI-2019art*). Этот компьютер ориентирован на обработку графовых и гиперграфовых структур различных видов, включая иерархические графы (см. *Dubrovin E..GraphRMftDMI-2022art*). Поддерживается представление в виде строк и списка смежных вершин, упорядоченных локальных списков инцидентных ребер, глобального упорядоченного списка инцидентных ребер.]

- *архитектура вычислительной системы без системы поддержки команд обработки графовых структур*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с системой команд (аппаратной) обработки знаний*

⇒ *примечание**:

[Примером такой архитектуры является архитектура компьютера Leonhard (см. *Rasheed B..NetwoGDUDI-2019art*). Компьютер Leonhard поддерживает системы команд DISC (Discrete Instruction Set Computing) (см. *Dubrovin E..GraphRMftDMI-2022art*). Система команд DISC включает следующие команды: создание целочисленного отношения со схемой, являющегося множеством объектов формального контекста (первым доменом бинарного отношения), тогда как соответствующим множеством образов является множество неотрицательных целых чисел (второй домен бинарного отношения); добавление пары в формальный контекст, содержащей добавляемый объект (ключ), который добавляется как кортеж через добавление элементов этого кортежа, вместе с целочисленным образом (значением) для этого объекта; получение следующего или предыдущего объекта в линейно (лексикографически) упорядоченным списке объектов; получение следующего большего или предыдущего меньшего объекта в линейно (лексикографически) упорядоченном списке объектов; получение минимального или максимального объекта в линейно (лексикографически) упорядоченном списке объектов; получение количества (мощности множества) образов для заданного объекта (кортежа-ключа); поиск пар по ключу; удаление пар; удаление всех пар формального контекста, включая объекты (ключи) и образы (значения); срез (подмножество) формальных контекстов контекста; объединение,пересечение и дополнение формальных контекстов. Для представления обрабатываемых данных используются B+-деревья. Другие архитектуры рассматривают реализацию операций обработки знаний, используя логическую модель представления знаний (см. *Hewitt C.MiddHoLPRPPatJFGP-2009el*), LISP-структуры (см. *Moon D.A.SymboA-1987art* , *Smith S.tLMI LTS-1984art*), обобщенные формальные языки (см. *Ivashenko V.P.StrinPMfKDS-2020art*, *Ивашенко В.П.МоделРЗвИС-2020кн*). В последнем случае для развития системы команд обработки знаний рассматривается переход от обработки знаний к обработки метазнаний (на базе семантики становления актуального и неактуального), результатом которого является система метаопераций (см. *Ивашенко В.П.МоделРЗвИС-2020кн*). Рассмотрение подобных архитектур важно для создания систем, управляемых знаниями (см. *Ивашенко В.П.ОперaУМДвЛАП-2016ст*).]

- *архитектура вычислительной системы без системы команд (аппаратной) обработки знаний*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с адаптивным распределением данных*

⇒ *примечание**:

[Адаптивное распределение данных (включая как частный случай виртуальное адресное пространство) важно для целей управления данными (и знаниями) и задач виртуализации для многозадачных и многопользовательских систем, а также тесно связано с возможностями масштабируемости системы.]

- *архитектура вычислительной системы без адаптивного распределения данных*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с системой команд локальной обработки информации*

⇒ *примечание**:

[Примером такой архитектуры является клеточный автомат (см. *Wolfram S.NewKoS-2002bk*). Элементарные клеточные (двоичные) автоматы разделяются на: быстро переходящие в однородное состояние (состояние только из нулей или единиц); быстро переходящие в устойчивое или циклическое состояние; остающиеся в хаотическом (случайном) состоянии; образующие как области с устойчивым или циклическим состоянием, так и области, в которых проявляются сложные взаимодействия элементов состояний, вплоть до Тьюринг-полных.

Обработка информации с помощью клеточных автоматов позволяет строить вычислительные системы в том числе с перестраиваемой (в том числе фрактало-подобной) структурой на основе локальных параллельно (конкурентно) выполняемых несложных правил. Существуют разновидности клеточных автоматов, поддерживающих необратимые, обратимые, детерминированные, недетерминированные, специализированные, универсальные (в том числе Тьюринг-полные) вычисления. Работа клеточных автоматов напоминает волновые процессы распространяющиеся в среде элементов состояния клеточного автомата.]

- *архитектура вычислительной системы без системы команд локальной обработки информации*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы исключительно с двоичным представлением данных в оперативной памяти*

⇒ *примечание**:

[Большинство современных архитектур цифровых вычислительных систем, включая реализации архитектуры фон Неймана, использует именно двоичное представление.]

- *архитектура вычислительной системы не исключительно с двоичным представлением данных в оперативной памяти*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с исключительно дискретным представлением данных*

⇒ *примечание**:

[Примером такой архитектуры является архитектура фон Неймана.]

- *архитектура вычислительной системы без исключительно дискретного представления данных*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с дискретным представлением данных*

⊃ *архитектура вычислительной системы с исключительно дискретным представлением данных*

- *архитектура вычислительной системы без дискретного представления данных*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с управлением от потока данных*

⇒ *примечание**:

[Архитектуры вычислительных систем с управлением от потока данных видятся более естественными при решении многих задач Искусственного интеллекта. Варианты таких архитектур рассмотрены в работах. Архитектуру клеточных автоматов можно рассматривать как архитектуру вычислительной системы с управлением от потока данных.]

- *архитектура вычислительной системы без управления от потока данных*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с управлением от потока команд*

⇒ *примечание**:

[Примером такой архитектуры является архитектура фон Неймана.]

- *архитектура вычислительной системы без управления от потока команд*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с процессором с арифметико-логическим устройством*

⇒ *примечание**:
[Примером такой архитектуры является архитектура фон Неймана.]
- *архитектура вычислительной системы без процессора с арифметико-логическим устройством*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с управляющим блоком со счетчиком инструкций*
⇒ *примечание**:
[Примером такой архитектуры является архитектура фон Неймана.]
- *архитектура вычислительной системы без управляющего блока со счетчиком иструкций*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с управляющим блоком с регистром команд*
⇒ *примечание**:
[Примером такой архитектуры является архитектура фон Неймана.]
- *архитектура вычислительной системы без управляющего блока с регистром команд*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с устройством ввода-вывода*
⇒ *примечание**:
[Примером такой архитектуры является архитектура фон Неймана.]
- *архитектура вычислительной системы без устройства ввода-вывода*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с доступом к внешнему (оперативному) запоминающему устройству*
⇒ *примечание**:
[Примером такой архитектуры является архитектура фон Неймана.]
- *архитектура вычислительной системы без доступа к внешнему (оперативному) запоминающему устройству*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с масштабируемой (модульной) глобальной памятью*
⇒ *примечание**:
[Масштабируемость как свойство архитектуры важно для систем, ориентированных на обучение (самообучение) с целью решения широкого класса задач. Задачи управления масштабируемой памятью рассматривались в работах *Ивашенко В.П.РаспрПвНЛАП-2019ст*, *Ивашенко В.П.ОпераУМДвЛАП-2016ст*. Подобные архитектуры могут быть ориентированы на обработку структур знаний, интегрированных в единое смысловое пространство.]
- *архитектура вычислительной системы без масштабируемой глобальной памяти*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с поддержкой модели активной графовой памяти*
⇒ *примечание**:
[Модель активной графовой памяти в архитектурах вычисленных систем важна для эффективной и согласованной (конвергентной) реализации параллельных процессов обработки знаний, включая механизмы возбуждения и торможения процессов обработки знаний. Принципы архитектур с поддержкой модели активной графовой памяти рассмотрены в *Ivashenko V.P.Appli oaIPfOMBPSUaUSKR-2021art*. Модель активной графовой памяти ориентирована на реализацию представления знаний в смысловом пространстве (см. *Ivashenko V.P.GenerPSRLaSS-2022art*) и реализацию систем, управляемых знаниями (см. *Ивашенко В.П.ОпераУМДвЛАП-2016ст*.]
- *архитектура вычислительной системы без поддержки модели активной графовой памяти*

}

⇒ *разбиение**:
{• *архитектура вычислительной системы с поддержкой параллельной обработки информации*
⇒ *примечание**:
[Архитектуры вычислительных систем с поддержкой параллельной обработки знаний важны для эффективной реализации процессов обработки знаний (см. *Ивашенко В.П.ИсслеПРПРСТМО-2020ст*, *Ивашенко В.П.МоделРЗвИС-2020кн*), повышения производительности и масштабируемости систем обработки знаний, включая многоагентные системы в виде интеллектуальных компьютерных систем и коллективов интеллектуальных компьютерных систем. Различные модели архитектур вычислительных систем с поддержкой параллельной обработки знаний рассмотрены в работе *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*.]

- *архитектура вычислительной системы с последовательной обработкой информации*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы с поддержкой последовательной модели консистентности глобальной оперативной памяти*

⇒ *примечание**:

[Архитектуры с поддержкой моделей консистентности ориентированы на решение задач управления взаимодействующими процессами, включая их синхронизацию и синхронные и асинхронные механизмы исполнения алгоритмов обработки знаний. Целью поддержки последовательной модели консистентности (см. *Ивашенко В.П.МоделРЗвИС-2020кн*) является обеспечение существования глобальных состояний базы знаний (см. *Ivashenko V.P..SemanLoREiaFBTM-2021art*) как структур единого смыслового пространства в интеллектуальных компьютерных системах. Для обеспечение той или иной модели консистентности могут использоваться различные механизмы рассмотренные в работах *Гапонов П.А.Модел иМПАП-2000дс*, *Сердюков Р.Е.БазовАиИСО-2004дс*, а также *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*.]

- *архитектура вычислительной системы без поддержки последовательной модели консистентности глобальной оперативной памяти*

}

⇒ *разбиение**:

{• *архитектура с поддержкой причинной модели консистентности памяти*

⇒ *примечание**:

[Архитектуры с поддержкой моделей консистентности ориентированы на решение задач управления взаимодействующими процессами, включая их синхронизацию и синхронные и асинхронные механизмы исполнения алгоритмов обработки знаний. Целью поддержки причинной модели консистентности (см. *Ивашенко В.П.МоделРЗвИС-2020кн*) является обеспечение интероперабельности и конвергенции в едином смысловом пространстве структур знаний агентов коллективов интеллектуальных компьютерных системах. Для обеспечение той или иной модели консистентности могут использоваться различные механизмы, рассмотренные в работах *Гапонов П.А.Модел иМПАП-2000дс*, *Сердюков Р.Е.БазовАиИСО-2004дс*, а также *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*.]

- *архитектура без поддержки причинной модели консистентности памяти*

}

⇒ *разбиение**:

{• *архитектура вычислительной системы, обладающая асимметрией*

⇒ *примечание**:

[Архитектуры вычислительных систем, обладающие асимметрией важны для эволюции многоагентных систем, интеллектуальных компьютерных систем и их коллективов, в которых асимметрия рассматривается в широком смысле, в том числе и как неоднородность таких систем или коллективов. Частным случаем случаем неоднородности является разнородность и гетерогенность архитектуры, которая позволяет реализовывать как интегрированное, так и гибридные модели обработки знаний, в рамках интеллектуальных компьютерных систем и их коллективов.]

- *архитектура вычислительной системы, обладающая симметрией*

}

Для определения архитектуры *ассоциативных семантических компьютеров*, в соответствии с выявленными классами и признаками, а также выработанными общими принципами, лежащими в основе таких архитектур, необходимо в рамках соответствующего признакового пространства рассмотреть конкретные множества архитектур и провести сравнительный анализ элементов этих множеств с целью обоснования выбора (оптимальных) вариантов архитектуры *ассоциативных семантических компьютеров*.

На данном этапе работы были выявлены несколько основных вариантов вариантов архитектуры *ассоциативных семантических компьютеров*, более подробно рассмотренные ниже.

§ 6.2.3. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для ostis-систем

В основу предлагаемого подхода к разработке *ассоциативного семантического компьютера* положены идеи, предложенные в работах В.В. Голенкова (см. *Голенков В.В.ГрафоМиМПА-1996дс*) и получившие развитие в работе В.М. Кузьмицкого *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*.

При формализации предметных областей, имеющих достаточно сложную семантическую организацию, перерабатываемые данные естественным образом группируются в некоторые сложные структуры. Эффективность решения задач, связанных с переработкой сложноструктурированных данных, на многопроцессорных вычислительных системах значительно возрастает в том случае, когда структура связей между процессорными элементами вычислительной системы, решающей эту задачу, совпадает со структурой данных, перерабатываемых в ходе ее решения (или, в более общем случае — отображается в структуру перерабатываемых данных простым и естественным образом). При переходе к переработке данных все более сложной структурной и семантической организации (а затем и к переработке знаний) сохранение высокой эффективности вычислительной системы обеспечивается главным образом путем увеличения числа одновременно работающих процессорных элементов и усложнения структуры связей между ними (см. *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*).

Такую тенденцию развития технических средств ЭВМ мы и рассмотрим в качестве основной линии эволюции, создающей предпосылки для появления *ассоциативных семантических компьютеров*. К ней относятся параллельные регулярные спецпроцессоры (векторные, матричные), спецвычислители для решения задач на графах и средства аппаратной поддержки семантических и нейронных сетей. К этой линии примыкают также и ассоциативные процессоры (в которых в роли процессорных элементов выступают ячейки ассоциативной памяти), процессоры баз данных и вычислительные системы, эффективно решающие те или иные классы задач за счет совпадения структуры связей между процессорными элементами со структурой информационного графа алгоритма (систолические вычислители, машины потоков данных) (см. *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*).

Закономерным результатом развития вычислительных систем является переход к системам, изменяющим структуру связей между процессорными элементами в процессе функционирования. Такие системы настраивают свою внутреннюю структуру на структуру перерабатываемых данных и информационные графы алгоритмов решаемых задач и могут решать разные классы задач, сохраняя при этом высокую эффективность.

Так образом развитая ЭВМ, ориентированная на переработку знаний, должна представлять собой в общем случае коллектив спецпроцессоров, ориентированных на максимально эффективное решение тех или иных классов задач, и обладать следующими свойствами:

- Спецпроцессоры представляют собой многопроцессорную вычислительную систему;
- Структура связей между процессорными элементами спецпроцессоров совпадает со структурой данных или (реже) со структурой информационного графа алгоритма решения задачи;
- Связи между процессорными элементами спецпроцессоров имеют перестраиваемую структуру;
- Набор и функции спецпроцессоров определяются для каждой машины переработки знаний конкретно в зависимости от набора предметных областей, на которые эта машина ориентирована, и специфики задач, решаемых в этих областях;
- Выявленный для некоторого семантического процессора набор механизмов переработки знаний должен быть "погружен" в язык представления и переработки знаний. При этом наиболее удобными для этой цели представляются языки семантических сетей;
- Процессорные элементы соответствуют вершинам или фрагментам семантической сети;
- Переработка информации сводится к изменению структуры связей между процессорными элементами, соответствующему изменению конфигурации семантической сети.

В качестве семантического спецпроцессора можно предложить нелинейную (графовую) структурно перестраиваемую (динамическую) процессоро-память, аппаратно реализующую некоторый язык переработки семантических сетей, а саму ЭВМ такого рода можно, таким образом, назвать графодинамическим параллельным ассоциативным компьютером или *ассоциативным семантическим компьютером*.

С учетом сказанного выше, а также общих принципов обработки информации в ostis-системах, описанных в *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*, рассмотрим более конкретно принципы, лежащие в основе реализации *ассоциативных семантических компьютеров*:

- нелинейная память — каждый элементарный фрагмент хранимого в памяти текста может быть логически инцидентен неограниченному числу других элементарных фрагментов этого текста. Таким образом, ячейки памяти, в отличие от обычной памяти, связываются не фиксированными условными связями, задающими фиксированную последовательность (порядок) ячеек в памяти, а логически или даже физически (с использованием технических средств коммутации) проводимыми связями произвольной конфигурации. Эти связи соответствуют дугам, ребрам, гиперребрам записанного в памяти графа (sc-текста);
- структурно-перестраиваемая (реконфигурируемая) память — процесс отработки хранимой в памяти информации сводится не только к изменению состояния элементов, но и к реконфигурации связей между ними. То есть, в ходе переработки информации в структурно-перестраиваемой памяти меняются на только и даже не столько состояния ячеек памяти, как это имеет место в обычной памяти, сколько конфигурация связей между этими ячейками. Т.е. в структурно-перестраиваемой памяти в ходе переработки информации не только перераспределяются метки на вершинах записанного в памяти графа, но и меняется структура самого этого графа;

- в качестве внутреннего способа кодирования знаний, хранимых в памяти *ассоциативного семантического компьютера*, используется универсальный (!) способ нелинейного (графоподобного) смыслового представления знаний — SC-код;
- обработка информации осуществляется коллективом агентов, работающих над общей памятью. Каждый из них реагирует на соответствующую ему ситуацию или событие в памяти (компьютер, управляемый хранимыми знаниями);
- есть программно реализуемые агенты, поведение которых описывается хранимыми в памяти агентно-ориентированными программами, которые интерпретируются соответствующими коллективами агентов;
- есть базовые агенты, которые не могут быть реализованы программно (в частности, это агенты интерпретации агентных программ, базовые рецепторные агенты-датчики, базовые эффекторные агенты);
- все агенты работают над общей памятью одновременно. Более того, если для какого-либо агента в некоторый момент времени в различных частях памяти возникает сразу несколько условий его применения, разные информационные процессы, соответствующие указанному агенту в разных частях памяти могут выполняться одновременно;
- для того, чтобы информационные процессы агентов, параллельно выполняемые в общей памяти не "мешали" друг другу, для каждого информационного процесса в памяти фиксируется и постоянно актуализируется его текущее состояние. То есть каждый информационный процесс сообщает всем остальным о своих намерениях и пожеланиях, которым остальные информационные процессы не должны препятствовать. Реализация такого подхода может выполняться, например, на основе механизма блокировок элементов семантической памяти, рассмотренного в *Главе 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*;
- процессор и память *ассоциативного семантического компьютера* глубоко интегрированы и составляют единую процессоро-память. Процессор *ассоциативного семантического компьютера* равномерно "распределен" по его памяти так, что процессорные элементы одновременно являются и элементами памяти компьютера. То есть каждая ячейка дополняется функциональным (процессорным) элементом, а перестраиваемые связи между ячейками становятся коммутируемыми каналами связи между процессорными элементами. Каждый процессорный элемент при этом имеет свою специальную внутреннюю регистровую память, отражающую важные для данного процессорного элемента аспекты текущего состояния процесса выполнения элементарных операций языка микропрограмм, обеспечивающих интерпретацию языка более высокого уровня (Языка SCP).

  Обработка информации в *ассоциативном семантическом компьютере* сводится к реконфигурации каналов связи между процессорными элементами, следовательно память такого компьютера есть не что иное, как коммутатор (!) указанных каналов связи. Таким образом, текущее состояние конфигурации этих каналов связи и есть текущее состояние обрабатываемой информации. Этот принцип обеспечивает значительное ускорение переработки информации благодаря исключению этапов передачи информации из памяти в процессор и обратно, но оплачивается ценой большой избыточности процессорных (функциональных) средств, равномерно распределяемых по памяти.

§ 6.2.4. Архитектура ассоциативных семантических компьютеров для ostis-систем

⇒ *подраздел**:
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*
- *Пункт 6.2.4.2. Вариант крупнозернистой архитектуры ассоциативных семантических компьютеров*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*

ассоциативный семантический компьютер
⊂ *компьютер с графодинамической ассоциативной памятью*
:= [associative semantic computer]
:= [sc-компьютер]
:= [аппаратно реализованный базовый интерпретатор семантических моделей (sc-моделей) компьютерных систем]
:= [аппаратно реализованная ostis-платформа]
:= [аппаратный вариант ostis-платформы]
:= [ассоциативный семантический компьютер, управляемый знаниями]
:= [компьютер с нелинейной структурно перестраиваемой (графодинамической) ассоциативной памятью, переработка информации в которой сводится не к изменению состояния элементов памяти, а к изменению конфигурации связей между ними]
:= [универсальный компьютер нового поколения, специально предназначенный для реализации семантически совместимых гибридных интеллектуальных компьютерных систем]
:=

[универсальный компьютер нового поколения, ориентированный на аппаратную интерпретацию логико-семантических моделей интеллектуальных компьютерных систем]
:= [универсальный компьютер нового поколения, ориентированный на аппаратную интерпретацию ostis-систем]
:= [ostis-компьютер]
:= [компьютер для реализации ostis-систем]
:= [компьютер, управляемый знаниями, представленными в SC-коде]
:= [компьютер, ориентированный на обработку текстов SC-кода]
:= [компьютер, внутренним языком которого является SC-код]
:= [компьютер, осуществляющий реализацию sc-памяти и интерпретацию scp-программ]
:= [предлагаемый нами компьютер нового поколения, ориентированный на реализацию интеллектуальных компьютерных систем и использующий SC-код в качестве внутреннего языка]
⇒ *разбиение**:
{• *scp-компьютер*
:= [sc-компьютер, обеспечивающий интепретацию scp-программ]
⇒ *обобщенная модель**:
базовая ostis-платформа
• *sc-компьютер с расширенным набором аппаратно реализуемых sc-агентов*
:= [sc-компьютер, обеспечивающий интепретацию scp-программ]
⇒ *обобщенная модель**:
специализированная ostis-платформа
}

scp-компьютер
:= [минимальная конфигурация аппаратно реализованной ostis-платформы, в рамках которой обеспечивается интерпретация scp-программ]
:= [минимальная конфигурация аппаратно реализованной ostis-платформы, в рамках которой аппаратно реализуются только базовые sc-агенты]
⇒ *пояснение**:
[В рамках scp-компьютера аппаратно реализуется (1) sc-память, (2) базовые sc-агенты, обеспечивающие интерпретацию scp-программ, (3) элементарные рецепторные sc-агенты, (4) элементарные эффекторные sc-агенты]

Для уточнения архитектуры *ассоциативных семантических компьютеров* необходимо уточнить:
- Базовую структуру компьютера, и, в частности, его процессоро-памяти;
- Алфавит элементов, хранимых в процессоро-памяти компьютера;
- Систему команд, интерпретируемых компьютером;
- Принципы управления процессом интерпретации указанных команд;
- Систему микропрограмм, обеспечивающих реализацию принципов управления указанным процессом.

Поскольку внутренним языком кодирования информации для *ассоциативного семантического компьютера* является SC-код, то алфавит элементов, хранимых в процессоро-памяти компьютера, совпадает с *Алфавитом SC-кода*, рассмотренным в *Главе 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*. При этом алфавит физически кодируемых синтаксических меток может быть расширен, например, из соображений производительности, по аналогии с тем, как это делается в программном варианте реализации *ostis-платформы* (см. *Главу 6.3. Программная платформа ostis-систем*).

В качестве системы команд для *ассоциативного семантического компьютера* предлагается *Язык SCP*, подробно рассмотренный в *§ 3.3.5. Базовый язык программирования ostis-систем*. Таким образом, как уже сказано в указанном параграфе, *Язык SCP* представляет собой ассемблер для *ассоциативного семантического компьютера*.

Для определения базовой структуры *ассоциативного семантического компьютера* уточним варианты такой структуры, предложенные в работах В.В. Голенкова и В.М. Кузьмицкого (см. *Голенков В.В.ГрафоМиМПА-1996дс*, *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*). В частности, в работе В.М. Кузмицкого предложен переход от крупнозернистых архитектур графодинамических машин к мелкозернистым (см. *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*).

Модели крупнозернистых архитектур имеют в своем составе параллельно функционирующие модули, обладающие следующими свойствами:
- Каждый модуль имеет строго фиксированное функциональное назначение в рамках архитектуры графодинамической машины в целом (так называемое глобальное функциональное назначение).
- Каждый модуль имеет относительно большой объем памяти (количество элементов памяти много больше общего количества модулей).
- Над памятью каждого типа модуля определен свой неэлементарный набор операций, выполняющих некоторые законченные преобразования над достаточно большими фрагментами памяти.

Основным формальным отличием для моделей мелкозернистых архитектур выступает иное соотношение между общим количеством модулей и количеством элементов памяти каждого модуля (емкостью памяти модуля), которое стремится к единице, и уровнем сложности операций модели. Соответственно свойствами моделей мелкозернистых архитектур можно считать следующие (см. *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*):

- Для каждого модуля по отдельности может не просматриваться его функциональное назначение в рамках графодинамической машины в целом. Вместе с тем каждый отдельный модуль в конкретный момент времени может иметь некоторое так называемое локальное функциональное назначение, соответствующее множество которых может уже рассматриваться как имеющее определенное так называемое глобальное функциональное назначение в рамках графодинамической машины в целом.
- Объем (количество элементов) памяти каждого модуля минимален и стремится к единице. Как следствие, общее количество модулей сопоставимо с общим количеством элементов памяти всех модулей.
- Для каждого модуля (в общем случае) набор операций, выполняемых над его памятью, элементарен и ограничен (конечен), так как воздействует лишь на один элемент (или лишь на несколько элементов) памяти и определяется очевидной ограниченностью (конечностью) семантики интерпретации содержимого элемента графовой памяти графодинамической машины.

Целесообразность перехода от крупнозернистых архитектур к мелкозернистым обусловлена соответствующим увеличением степени потенциального параллелизма в процедурах переработки знаний.

С учетом сказанного можно говорить о нескольких вариантах уточнения базовой структуры *ассоциативного семантического компьютера*, каждый из которых имеет определенные преимущества и недостатки. Рассмотрим эти варианты более подробно.

Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры

Одним из логичных и наиболее простых с архитектурной точки зрения вариантов аппаратной реализации ostis-платформы является реализация средств хранения конструкций SC-кода и интерпретации scp-программ на аппаратном уровне по аналогии с тем, как это делается в программных вариантах ostis-платформы на базе современных компьютеров (см. *Главу 6.3. Программная платформа ostis-систем*). В этом случае общая архитектура компьютера остается фон-Неймановской (с явным выделением отдельного процессорного модуля и отдельного модуля памяти). К основным особенностям такого варианта реализации относятся следующие:

- Модуль памяти (реализация *sc-памяти*) представляет собой множество ячеек, каждая из которых может хранить некоторый sc-элемент или может быть пустой. Каждая ячейка такой памяти имеет некоторый уникальный внутренний адрес, аналогичный адресу ячеек фон-Неймановской памяти. В то же время, при обработке информации, хранимой в такой памяти, на уровне языка команд (Языка SCP), в отличие от фон-Неймановской памяти адреса ячеек не учитываются, доступ к sc-элементам осуществляется исключительно по связям инцидентности между ними. Исключение составляют некоторые ключевые sc-элементы, набор которых оговаривается отдельно и доступ к которым осуществляется по какому-либо другому идентификатору, например, системному sc-идентфикатору или основному sc-идентфикатору для какого-либо внешнего языка, но не по адресу. Подразумевается, что если в ячейке хранится некоторый sc-элемент, то в ней хранится информация, характеризующая этот sc-элемент, а именно:
  - синтаксическая метка, задающая тип соответствующего sc-элемента;
  - содержимое внутреннего файл ostis-системы или ссылка на внешнюю файловую систему (если хранится внутренний файл ostis-системы);
  - перечень связей инцидентности данного sc-элемента с другими sc-элементами, что фактически означает хранение набора адресов ячеек памяти, соответствующих sc-элементам, инцидентным данному sc-элементу. Конкретный перечень типов хранимых связей может уточняться в зависимости от реализации. Например, по аналогии с тем, как это сделано в программном варианте реализации ostis-платформы (см. *Главу 6.3. Программная платформа ostis-систем*), целесообразно хранить в ячейке памяти адрес ячейки, соответствующей первому sc-коннектору, инцидентному соответствующему sc-элементу соответствующим типом инцидентности, а в рамках ячейки, соответствующей данному sc-коннектору хранить адрес ячейки, соответствующей следующему sc-коннектору, инцидентного тому же sc-элементу тем же типом инцидентности и так далее. При таком подходе размер каждой ячейки памяти можно сделать фиксированным.
  - метка блокировки sc-элемента с указанием метки соответствующего процесса;
  - метка уровня доступа и любые другие метки при необходимости.
- Процессорный модуль реализует набор команд, соответствующих *атомарным типам scp-операторов*.

- Для связи с внешней средой вводится *терминальный модуль* (см. *Голенков В.В.ГрафоМиМПА-1996дс*, *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*), который в общем случае может быть реализован по разному и задачами которого являются:
  - подготовка (генерация) информации, поступающей из внешней среды для ее последующей ее загрузки в процессорный модуль и модуль памяти;
  - передача (использование, реализация) информации, подготовленной (полученной, представленной) в процессорном модуле и модуле памяти, во внешнюю среду.
- Для хранения содержимых внутренних файлов ostis-системы большого размера может оказаться целесообразным иметь отдельную файловую память, построенную по фон-Неймановским принципам. Тогда в ячейках семантической памяти, соответствующих таким внутренним файлам ostis-системы, будет хранится не непосредственно их содержимое, а адрес этого файла на файловой памяти.
- Для реализации принципов многоагентной обработки информации, предлагаемых в рамках Технологии OSTIS (см. *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*) необходимо реализовать (например, в рамках терминального модуля) подсистему регистрации и обработки событий, которая позволит осуществлять инициирование sc-агентов при возникновении соответствующих событий в памяти.

К достоинствам такого варианта реализации относятся:
- Относительная простота и невысокая трудоемкость реализации по сравнению с разработкой рассмотренного ниже полноценного варианта процессоро-памяти. В частности, при условии наличия стабильно работающего варианта программной реализации ostis-платформы, в котором хотя бы нижний уровень реализован на языках достаточно низкого уровня, таких как C, для упрощения процесса разработки аппаратной архитектуры возможно использование средств автоматизации перехода от программ на C к описанию на языках описания аппаратуры (HDL — hardware description language, например, VSDL и Verilog), также известных как “C to HDL”. Среди популярных средств и языков этого класса можно отметить LegUp (см. *LegUpHLS-2023el*), VHDPlus (см. *VHDPl-2023el*), SystemC (см. *SysteCCP-2023el*), MyHDL для языка Python (см. *MyHDL-2023el*) и множество других.
- Простота интеграции с современными компьютерными системами, в частности, можно рассматривать гибридный вариант, при котором *ассоциативный семантический компьютер* реализуется как отдельный подключаемый модуль для современного компьютера, предназначенный для повышения эффективности обработки sc-конструкций.

Очевидным ключевым недостатком такого варианта является его ориентация на фон-Неймановскую архитектуру со всеми ее недостатками, перечисленными ранее. Кроме того, в таком варианте по умолчанию на аппаратном уровне не закладывается обеспечение параллельной обработки sc-конструкций. Указанный недостаток частично устраняется в рассматриваемом далее варианте крупнозернистой архитектуры *ассоциативных семантических компьютеров*.

Пункт 6.2.4.2. Вариант крупнозернистой архитектуры ассоциативных семантических компьютеров

Целью перехода к крупнозернистой архитектуре *ассоциативных семантических компьютеров* является реализация на аппаратном уровне параллельной обработки sc-конструкций.

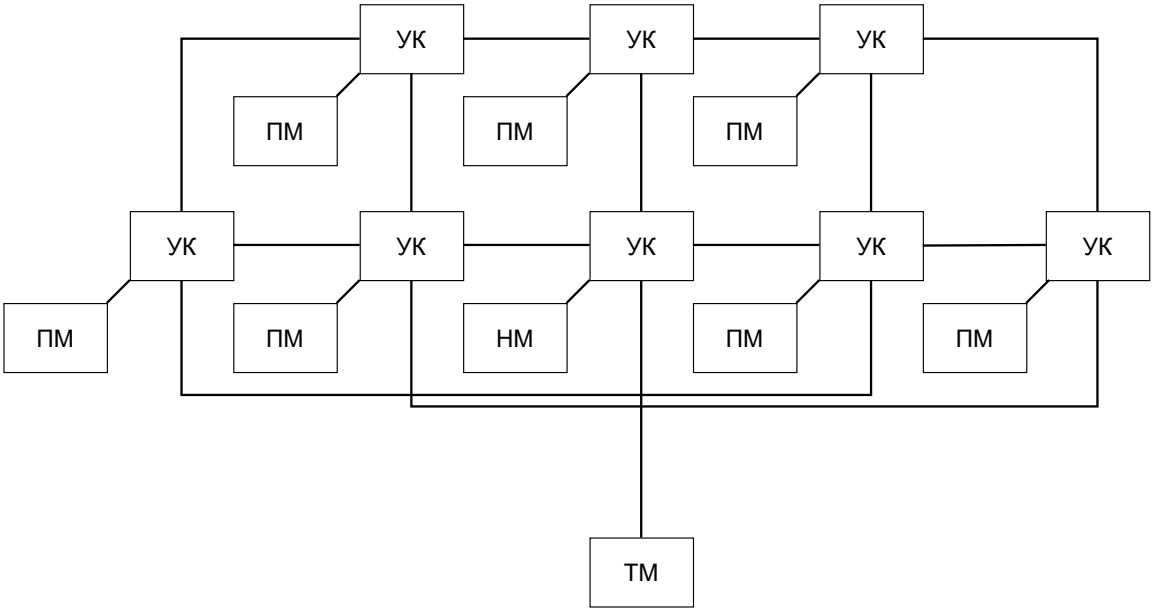
К основным особенностям такого варианта реализации относятся следующие:
- *ассоциативный семантический компьютер* разделяется на несколько однотипных модулей, устроенных аналогично тому, как строится рассмотренный в предыдущем пункте вариант реализации ассоциативного семантического компьютера на базе фон-Неймановской архитектуры. Такие модули будем называть “комбинированными модулями”, поскольку такой модуль имеет свой процессорный модуль и свой модуль памяти (накопительный модуль), отдельно выделяемых общих процессорных модулей не предполагается. Может существовать отдельный общий модуль памяти, в который при необходимости будет записываться информация, которая не поместилась в память конкретного комбинированного модуля.
- По-прежнему выделяется терминальный модуль, обеспечивающий связь системы комбинированных модулей с внешней средой.
- Может отдельно выделяться модуль файловой памяти.
- Число комбинированных модулей относительно невелико (2 — 16), каждый модуль представляет собой достаточно мощное устройство (фактически — отдельный *ассоциативный семантический компьютер*) и, соответственно, для решения задач некоторых классов может оказаться достаточно одного комбинированного модуля.
- В то же время в общем случае для решения задачи необходимо задействовать несколько комбинированных модулей. В этом случае обрабатываемая sc-конструкция распределяется между несколькими модулями, для чего

создаются sc-узлы-копии, позволяющие обеспечить семантическую связь между фрагментами sc-конструкции, хранящимися в разных комбинированных модулях. Для записи таких конструкций было разработано расширение *SC-кода*, названное *SCD-кодом* (Semantic Code Distibuted, см. *Голенков В.В.ГрафоМиМПА-1996дс* , *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*), соответственно конструкции такого языка получили название *scd-конструкций*, их элементы — *scd-элементов* (*scd-узлов*, *scd-дуг*).

- Аналогично для обработки *scd-конструкций* было разработано расширение *Языка SCP*, названное *Языком SCPD*, учитывающее тот факт, что разные фрагменты обрабатываемой конструкции могут физически храниться в разных комбинированных модулях. При этом предполагается, что все элементы *scd-конструкции*, представляющей *scpd-программу* (программу *Языка SCPD*), должны быть расположены в памяти одного комбинированного модуля, но каждая *scpd-программа* может иметь несколько полных копий в разных комбинированных модулях.
- Для синхронизации параллельных процессов обработки информации комбинированные модули обмениваются сообщениями, которые могут содержать как фрагменты обрабатываемых *scd-конструкций*, так и команды *Языка SCDP*. Соответственно *Язык SCPD* по сравнению с *Языком SCP* имеет дополнительные средства, поддерживающие распределенную переработку графовых конструкций (см. *Голенков В.В.ГрафоМиМПА-1996дс*, *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*):
  - В *Язык SCPD* встроены средства, позволяющие распознавать "свой" и "чужой" комбинированный модуль, для этого вводятся операторы работы с идентификаторами модулей;
  - Предусмотрена возможность создания копии *scd-элемента* в памяти другого модуля. Для этой цели вводится группа операторов работы с копиями: создание копии *scd-элемента* в указанном модуле, перенос связей элемента-оригинала на копию, склеивание копий элемента, поиск копии данного элемента в заданном модуле и так далее;
  - Предусматривается возможность явного удаленного вызова *scpd-программы* в указанном процессорном модуле. Для параллельного запуска одинаковых процессов, выполняющихся в разных процессорных модулях, предусмотрены операторы, запускающие программу на выполнение в модулях из указанного списка.
  - Имеются средства межпроцессной и внутрипроцессной синхронизации: операторы формирования сообщения, операторы ожидания сообщения, операторы перевода процесса в режим ожидания завершения выполнения распределенно выполняющихся операторов, операторы ожидания завершения выполнения всех распределенно выполняющихся операторов.
- Для обмена сообщениями каждый комбинированный модуль имеет соответствующие подмодули, позволяющие отправлять и принимать сообщения, а также буфер сообщений для хранения очереди полученных сообщений, ожидающих обработки и сообщений, ожидающих отправки.
- Для интерпретации *scpd-программ* разрабатывается семейство микропрограмм на языке, который в общем случае зависит от выбранных аппаратных компонентов, из которых строятся комбинированные модули.

К описанному варианту реализации *ассоциативных семантических компьютеров* с крупнозернистой архитектурой относится и упомянутая ранее мультитранспьютерная реализация (см. *Голенков В.В.ТермиМПГКИсПиПМСЛОМпМОЭВМ-1994кн* , *Голенков В.В.ГрафоМиМПА-1996дс*). В основу указанной реализация были положены ПК IBM PC 386 (486, Pentium) и 8 транспьютеров T805. На рисунке *Рисунок. Пример реализации крупнозернистой архитектуры* схематично показан такой вариант реализации на 8 транспьютерах, где каждый транспьютер одновременно выполняет роль узла коммутации (“УК”) и процессорного модуля (“ПМ”) или накопительного модуля (“НМ”). Вся система при этом взаимодействует с внешней средой посредством терминального модуля (“ТМ”).

Рисунок. Пример реализации крупнозернистой архитектуры
=



Основным достоинством крупнозернистой архитектуры *ассоциативных семантических компьютеров* является ориентация на аппаратную поддержку параллельной обработки конструкций *SC-кода*. В то же время у такого варианта реализации выделяется ряд недостатков:

- Каждый комбинированный модуль строится по принципам *машины фон Неймана*, соответственно, ее недостатки в полной мере не устраняются;
- Несмотря на сохранение общих принципов *SC-кода* и *Языка SCP*, распределенное хранение и обработка sc-конструкций требует разработки отдельных языковых средств, таких как *SCD-код* и *Язык SCPD*, и их поддержки на базе выбранной аппаратной архитектуры. Кроме того, как видно и рассмотренных выше принципов *Языка SCPD*, при разработке *scpd-программ* необходимо явно учитывать тот факт, что обработка выполняется распределенно.

Следующий шагом с точки зрения иерархии архитектур *ассоциативных семантических компьютеров* является мелкозернистая архитектура ассоциативных семантических компьютеров.

Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров

Как уже было сказано, целесообразность перехода от крупнозернистых архитектур к мелкозернистым обусловлена соответствующим увеличением степени потенциального параллелизма в процедурах переработки знаний. При этом максимально возможный параллелизм, очевидно, будет иметь место при предельной реализации мелкозернистых архитектур, в которых один структурный модуль процессоро-памяти будет соответствовать одному элементу памяти, то есть в нашем случае — одному sc-элементу.

Рассмотрим более детально принципы, лежащие в основе мелкозернистой архитектуры *ассоциативного семантического компьютера*:

- Процессоро-память *ассоциативного семантического компьютера* состоит из однотипных модулей, которые будем называть процессорными элементами sc-памяти или просто *процессорными элементами*. Каждый *процессорный элемент* соответствует одному sc-элементу (хранит один sc-элемент). При этом в каждый момент времени каждый *процессорный элемент* может быть пустым (не хранить никакой sc-элемент) или заполненным, то есть иметь взаимно однозначно соответствующий ему хранимый sc-элемент. На физическом уровне для описания этого факта вводится соответствующий признак, имеющим два значения. Таким образом, каждый *процессорный элемент* "отвечает" только за один sc-элемент и, в отличие от крупнозернистого варианта архитектуры *ассоциативного семантического компьютера*, задача не может быть решена одним *процессорным элементом* и количество таких *процессорных элементов* достаточно велико (соответствует максимальному возможному числу sc-элементов, хранимых в базе знаний некоторой ostis-системы). Опыт разработки прикладных ostis-систем показывает, что в среднем число sc-элементов в базе знаний такой ostis-системы составляет от нескольких сотен тысяч до нескольких миллионов. Ситуация, когда в рамках

процессоро-памяти необходимо представить sc-конструкцию, число элементов которой больше числа *процессорных элементов* на данный момент не рассматривается и требует дополнительного исследования.

- Каждый *процессорный элемент* (по аналогии с ячейкой памяти в случае реализации *ассоциативного семантического компьютера* на фон-Неймановской архитектуре) имеет некоторый уникальный внутренний идентификатор — *адрес процессорного элемента*. Адреса *процессорных элементов*, в отличие от адресов ячеек фон-Неймановской памяти, не обеспечивают непосредственный доступ к *процессорным элементам*, а позволяют однозначно идентифицировать процессорный элемент при обмене сообщениями согласно рассмотренным ниже принципам.
- Каждый процессорный элемент имеет память, в которой хранится
  - синтаксическая метка, задающая тип соответствующего sc-элемента;
  - содержимое внутреннего файла ostis-системы или ссылка на внешнюю файловую систему (если данный процессорный элемент соответствует внутреннему файлу ostis-системы);
  - перечень логических связей данного процессорного элемента с другими, то есть перечень адресов процессорных элементов, связанных с данным процессорным элементом *логическими каналами связи* с указанием типа связи (подробнее о *логических каналах связи* см. ниже);
  - метка блокировки sc-элементов с указанием метки соответствующего процесса;
  - другие метки при необходимости (например, метки уровня доступа к хранимому sc-элементу);
  - волновые микропрограммы, выполняемые данным процессорным элементом в данный момент (подробнее о *волновых микропрограммах* см. ниже), и временные данные для этих микропрограмм, а также очередь микропрограмм при необходимости.
- Процессорные элементы связаны между собой двумя типами каналов связи — *физическими каналами связи* и *логическими каналами связи*.
  - В общем случае число *физических каналов связи* у каждого *процессорного элемента* может быть произвольным, кроме того теоретически *физические каналы связи* между процессорными элементами могут перестраиваться (перекоммутироваться) с течением времени, например, с целью оптимизации времени передачи сообщений между процессорными элементами. Конфигурация *физических каналов связи* не учитывается на уровне логической обработки знаний, как на уровне Языка SCP, так и на уровне языка микропрограмм, обеспечивающих интерпретацию команд Языка SCP, то есть *scp-операторов*. Для упрощения в рамках данной работы будем рассматривать вариант физической реализации sc-памяти, в котором каждый процессорный элемент имеет фиксированное и одинаковое для всех процессорных элементов число *физических каналов связи* (N), при этом конфигурация таких каналов связи с течением времени не меняется. Очевидно, что минимальным значением N является 2, в этом случае мы получим линейную цепочку *процессорных элементов*. При N равном 4 мы получим двумерную "матрицу" *процессорных элементов*, При N равном 6 — трехмерную "матрицу" *процессорных элементов* и так далее. Будем называть “смежными” *процессорные элементы*, непосредственно связанные *физическим каналом связи*.
  - В таком случае можно сказать, что каждый процессорный элемент имеет свой "адрес" (уникальный идентификатор) в некотором многомерном пространстве, число измерений (признаков) которого определяется числом N *физических каналов связи*, связанных с одним *процессорным элементом*. В приведенных выше примерах размерность такого пространства равна N/2, что позволяет предположить, что число N целесообразно делать четным.
  - Каждый *физический канал связи* и каждый *логический канал связи*, таким образом, задается парой *адресов процессорных элементов*.
  - *логические каналы связи* между процессорными элементами формируются динамически и соответствуют *связям инцидентности* между sc-элементами. Таким образом, *логические каналы связи* могут описывать два типа связей инцидентности — *инцидентность обозначений sc-пар с их компонентами* и *инцидентность обозначений ориентированных sc-пар с их вторыми компонентами* (см. *Пункт 2.2.2.1. Синтаксическое Ядро SC-кода*). При этом конфигурация *логических каналов связи* в общем случае никак не связана с конфигурацией *физических каналов связи* — инцидентные sc-элементы могут физически храниться в процессорных элементах, не являющихся смежными. В тоже время очевидно, что в общем случае некоторые *физические каналы связи* могут соответствовать логическим.
  - Кроме связей инцидентности *логические каналы связи* могут соответствовать и другим типам связей между sc-элементами, по аналогии с тем, как это сделано в программном варианте реализации ostis-платформы (см. *Главу 6.3. Программная платформа ostis-систем*). Например, для упрощения реализации алгоритмов поиска в базе знаний и уменьшения объема памяти, которым должен обладать каждый *процессорный элемент*, целесообразно хранить в памяти процессорного элемента адрес только первого sc-коннектора, инцидентного соответствующему sc-элементу соответствующим типом инцидентности, а в рамках процессорного элемента, соответствующего данному sc-коннектору, адрес следующего sc-коннектора, инцидентного тому же sc-элементу тем же типом инцидентности и так далее. При таком подходе количество памяти процессорного элемента, хранящей логические связи между процессорными элементами, можно сделать фиксированным.

- Каждый процессорный элемент может отправлять сообщения (микропрограммы) другим процессорным элементам и принимать сообщения от других процессорных элементов по *логическим каналам связи* и имеет соответствующие рецепторно-эффекторные подмодули. На физическом уровне передача сообщений осуществляется, в свою очередь, по *физическим каналам связи*, конфигурация которых, как было сказано выше, фиксируется и в общем случае не зависит от конфигурации логических каналов связи.
- Таким образом, процессорные элементы формируют однородную процессоро-память, в которой нет отдельно выделяемых модулей, предназначенных только для хранения информации и отдельно выделяемых модулей, предназначенных только для ее обработки.
- Для связи такой процессоро-памяти с внешней средой вводится *терминальный модуль*, который в общем случае может быть реализован по разному и задачами которого являются:
  - подготовка (генерация) информации, поступающей из внешней среды для ее последующей загрузки в процессорные модули;
  - передача (использование, реализация) информации, подготовленной (полученной, представленной) в процессорных модулях, во внешнюю среду.
- Для хранения содержимых внутренних файлов ostis-системы большого размера может оказаться целесообразным иметь отдельную файловую память, связанную с процессоро-памятью и построенную по традиционным фон-Неймановским принципам. Это обусловлено тем, что основная цель построения процессоро-памяти – обеспечение как можно большей параллельности при обработке конструкций SC-кода, в случае же с хранением и обработкой содержимых внутренних файлов ostis-системы, которые по определению являются внешними по отношению к SC-коду информационными конструкциями, целесообразно использовать современные традиционные подходы.

Перечисленные принципы позволяют сформулировать ключевую особенность обработки информации, хранимой в рамках такой процессоро-памяти. В отличие от архитектуры фон-Неймана (и других архитектур, разработанных примерно в то же время, например, Гарвардской архитектуры) и даже от *программного варианта ostis-платформы* в предлагаемой архитектуре процессоро-памяти отсутствует общая память, доступная для всех модулей, осуществляющих обработку информации. Благодаря этому значительно упрощается параллельная обработка информации, однако усложняется реализации набора микропрограмм интерпретации команд обработки информации в такой памяти, поскольку каждый процессорный элемент становится очень "близоруким" и "видит" только те процессорные элементы, которые связаны с ним *логическими каналами связи*.

Таким образом, язык описания микропрограмм интерпретации команд *ассоциативного семантического компьютера* не может быть построен как традиционный язык программирования, например, процедурного типа, поскольку все такие языки предполагают возможность непосредственного адресного или ассоциативного доступа к произвольным элементам памяти. Предлагаемый язык описания микропрограмм предлагается строить по принципам *волновых языков программирования* (см. *Сапатый П.С.ЯзыкВкОНС-1986ст*, *Moldovan D.I..SNAP aVLSIAfAIP-1985art*) и инсерционного программирования (см. *Летичевский А.А..ИнсерП-2003ст*, *Летичевский А.А..ИнсерМ-2012ст*).

В рамках такого языка микропрограмм выделяется два типа волн:
- Волны, передаваемые только по *логическим каналам связи* (например, при поиске инцидентных sc-элементов).
- Волны, передаваемые по всем каналам связи (например, при создании новых логических каналов связи, то есть при генерации новых sc-элементов).

Рассмотрим более детально принципы интерпретации команд (scp-операторов) в рамках рассмотренной процессоро-памяти.
- Каждый *процессорный элемент* может интерпретировать некоторый ограниченный набор микропрограмм. С учетом того, что один процессорный элемент соответствует одному sc-элементу, то множество операций, связанных с преобразованием данного sc-элемента, очень ограничено (сгенерировать sc-элемент указанного типа, удалить sc-элемент, изменить содержимое sc-файла, установить или снять метку блокировки и так далее). Таким образом, важной задачей процессорного элемента будет формирование сообщений для других процессорных элементов и их отправка.
- Каждый процессорный элемент может порождать и хранить в памяти временные данные для микропрограмм. Предполагается, что объем памяти, имеющейся в распоряжении процессорного элемента, достаточен для представления всех необходимых данных для возможного набора микропрограмм, поскольку такие микропрограммы достаточно просты (см. предыдущий принцип). В случае, если по каким-либо причинам переполнение все же происходит, то могут использоваться различные подходы, например, описанные в работе *Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс*.
- Каждый процессорный элемент может сформировать микропрограмму и отправить ее в виде волнового сообщения для выполнения другими процессорными элементами. Передача сообщений происходит по физическим каналам связи. Поскольку конфигурация физических каналов связи в общем случае не связана конфигурацией логических каналов связи, то каждый процессорный элемент самостоятельно принимает решение о необходимости выполнения микропрограммы и передачи ее дальше. Здесь можно провести аналогию с волновым алгоритмом поиска пути в графе (вариант поиска в ширину).

- Часто процессорные элементы будут не выполнять микропрограмму, а передавать ее дальше, таким образом, сами процессорные элементы выполняют также и роль коммутационных элементов, при этом в общем случае каждый процессорный элемент может входить в произвольное число маршрутов при передаче сообщений по логическим каналам связи между процессорными элементами.
- Как и в случае с крупнозернистой архитектурой, у каждого процессорного элемента есть очередь микропрограмм, подлежащих выполнению (входящих сообщений), и очередь микропрограмм, подлежащих отправке (выходящих сообщений). При этом в рамках каждого процессорного элемента также можно говорить о возможности параллельного выполнения каких-либо операций (например, формирование выходящих сообщений и обработку текущего хранимого sc-элемента).

Соответственно, можно говорить об иерархии микропрограмм:
- Микропрограммы по изменению хранимого sc-элемента:
  - Выполнить указанное преобразование содержимого данного sc-узла;
  - Изменить метку типа sc-элемента (если такое изменение не противоречит *Синтаксису SC-кода*);
  - Заменить блокировку данного sc-элемента для указанного процесса (в том числе, снять метку);
  - Удалить sc-элемент.
- Микропрограммы по обработке sc-элементов, хранимых в других (не обязательно смежных процессорных элементах):
  - Сгенерировать инцидентный sc-коннектор (и новый *логический канал связи*), возможно, вместе со смежным sc-элементом;
  - Сгенерировать оба или один sc-элемент, соединяемые данным sc-коннектором;
  - Найти все sc-коннекторы (то есть адреса соответствующих им процессорных элементов) указанного типа, инцидентные данному sc-элементу указанным типом инцидентности;
  - Найти sc-узлы, инцидентные данному sc-коннектору.
- Микропрограммы по управлению процессами выполнения других микропрограмм:
  - Переслать указанную микропрограмму для исполнения из данного процессорного элемента по всем указанным каналам (инцидентным sc-коннекторам указываемого типа) всем смежным sc-элементам указываемого типа;
  - Дождаться выполнения микропрограмм указанного типа, порожденных указанным процессорным элементом и результат их выполнения передать процессорному элементу, запросившему соответствующую информацию.
- И другие.

Очевидно, что при решении конкретной задачи указанные микропрограммы могут комбинироваться в более сложные микропрограммы. Приведенная иерархия на данный момент не является полной и требует дальнейшего уточнения.

Исходя из представленных принципов формируется иерархия языков программирования для предложенной мелкозернистой архитектуры *ассоциативных семантических компьютеров*:
- Язык SCP, не зависящий от реализации ostis-платформы, на котором пишутся программы sc-агентов обработки знаний. Язык SCP является "водоразделом" между платформенно-зависимой частью и платформенно-независимой частью ostis-системы, таким образом, он является самым низкоуровневым языком среди всех возможных платформенно-независимых языков, и одновременно языком высокого уровня с точки зрения ostis-платформы.
- Язык микропрограмм, которыми обмениваются процессорные элементы между собой, и которые исполняются этими процессорными элементами. Фактически на этом языке разрабатывается интерпретатор Языка SCP. Важно отметить, что язык микропрограмм ориентирован на передачу сообщений по *логическим каналам связи* и не учитывает конфигурацию *физических каналов связи*. Для этого вводится еще один язык более низкого уровня.
- Язык для записи программ управления процессами обмена сообщениями (микропрограммами). Введение такого языка необходимо, поскольку, как было сказано, сам по себе язык микропрограмм не учитывает
  - Конфигурацию физических каналов связи. Таким образом, при отправке сообщения по логическому каналу связи необходимо сформировать необходимое число сообщений в зависимости от числа имеющихся физических каналов связи, осуществить кодирование передаваемого сообщения для передачи по физическому каналу связи, передать сообщение с учетом того, что один и тот же физический канал связи может входить в общем случае в произвольное число маршрутов между процессорными элементами, осуществить декодирование сообщения на принимающем процессорном элементе. Все эти задачи требуют разработки соответствующих программ;
  - Организацию очереди входящих и выходящих сообщений внутри *процессорного элемента*, добавление сообщений в очередь, извлечение сообщений из очереди для выполнения и так далее.

Достоинства предложенного мелкозернистого варианта архитектуры *ассоциативных семантических компьютеров*:

- В рамках предложенной мелкозернистой архитектуры, в отличие от крупнозернистой, нет необходимости создания копий sc-элементов, и разработки специальных языков кодирования для полученных конструкция, таких как *SCD-код*, поскольку каждый процессорный элемент хранит один атомарный фрагмент всей хранимой sc-конструкции и число логических связей с другими процессорными элементами не ограничено.
- Приведенная явно выделяемая иерархия языков программирования позволяет исключить на уровне разработки пользовательских программ (на *Языке SCP* и языков более высокого уровня на его основе) необходимость учитывать факт распределенного хранения sc-конструкций и вообще принципы организации ostis-платформы. Другими словами, не требуется разработка таких языков, как *Язык SCPD*.
- Расширяемость архитектуры позволяет легко наращивать число процессорных элементов без существенного снижения производительности, поскольку в предложенной архитектуре нет явно выделяемых процессорных модулей и накопительных модулей, соответственно исключается необходимость передачи информации между такими модулями, кроме того процессорный модуль перестает быть разделяемым ресурсом для большого числа одновременно выполняемых процессов. Все перечисленное позволит в конечном итоге решить проблему, известную как проблема "бутылочного горлышка" архитектуры фон Неймана (см. *Backus J.CanPBLftNS-1978art*).
- Ключевым достоинством предложенной мелкозернистой архитектуры является ее ориентация на максимально возможную поддержку параллельной обработки информации на аппаратном уровне и в конечном итоге возможность реализации любых моделей параллелизма с учетом решаемой задачи. В подтверждение данного тезиса можно привести теорию А-систем, описанную в работе В. Е. Котова и А. С. Нариньяни *Котов В.Е..АсинхВПнОП-1966ст*. По словам самих авторов, данное понятие стоит трактовать как универсальную модель для некоторого класса параллельных систем, которая требует уточнения в случае конкретных реализаций. В частности, в рамках данной теории выделяются процессорные элементы, активация/деактивация которых осуществляется посредством так называемой спусковой функции, принимающей значения 0 и 1. Понятно, что в конкретной реализации в качестве такой функции может быть использован любой признак, имеющий значения истина и ложь, указывающий на то, что тот или процессорный элемент должен быть активирован в следующий момент времени. Авторами показана возможность формализации на основе данной модели любых параллельных алгоритмов, рассмотрена возможность сведения таких алгоритмов к последовательным, варианты синхронизации в рамках такой модели.

  Можно провести очевидную параллель между А-системами и предложенной мелкозернистой архитектурой *ассоциативных семантических компьютеров* с учетом наличия волнового языка микропрограммирования:
  - Процессорным элементам из теории А-систем соответствуют *процессорные элементы* процессоро-памяти;
  - В роли спусковых функций для процессорных элементов выступают микропрограммы, передаваемые волнами от одного процессорного элемента к другому и, соответственно, активизирующие деятельность процессорных элементов.

  Стоит отметить, что несмотря на то, что рассмотренная работа по теории А-систем известна уже более полувека, авторам данной главы не удалось найти попытки реализовать идеи этой теории в аппаратном варианте. На наш взгляд, это обусловлено тем, что уровень развития микроэлектроники на тот момент не соответствовал необходимым для реализации теории А-систем требованиям.

Вместе с перечисленными достоинствами можно выделить ключевой недостаток предложенного мелкозернистого варианта архитектуры *ассоциативных семантических компьютеров*, который заключается в сильной зависимости быстродействия процессоро-памяти от времени передачи волновых микропрограмм от одного процессорного элемента к другому. При этом, поскольку на логическом уровне передача сообщений осуществляется по *логическим каналам связи*, а реально — по *физическим каналам связи*, то быстродействие процессоро-памяти будет зависеть от того, насколько близко соответствует конфигурация *логических каналов связи* конфигурации *физических каналов связи*. Очевидно, что в общем случае взаимно однозначное соответствие этих конфигураций невозможно, поскольку число *физических каналов связи*, инцидентных заданному процессорному элементу, ограничено в отличие от числа *логических каналов связи*. Тем не менее, возможны несколько вариантов оптимизации размещения sc-конструкций в процессоро-памяти:

- при записи ("укладке") sc-конструкции в процессоро-память (в особенности в случае достаточно больших sc-конструкций) можно учитывать семантику записываемых фрагментов, и записывать их таким образом, чтобы те sc-элементы, сообщение к которым будет передаваться от данного sc-элемента с большей вероятностью, находились физически ближе к данному sc-элементу. Так, например, можно учитывать денотационную семантику scp-операторов поиска, которые ориентированы на обработку *трехэлементных sc-конструкций* и *пятиэлементных sc-конструкций*, а также хранить sc-элементы, инцидентные заданному sc-коннектору по возможности ближе к нему;
- Если число логических связей между элементами sc-конструкции не превышает числа доступных физических каналов связи процессорного элемента и sc-граф является планарным (хоть sc-граф не является классическим

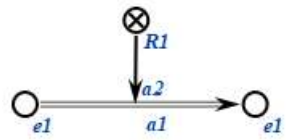
графом, можно говорить о его планарности по аналогии с планарностью классических графов), то возможна запись sc-конструкции в процессоро-память таким образом, чтобы конфигурация *логических каналов связи* взаимно однозначно соответствовала какому-то подмножеству физических каналов связи. Таким образом, актуальной является разработка алгоритмов оптимальной "укладки" sc-графов в процессоро-память для обеспечения последующей эффективности передачи сообщений между процессорными элементами;

- Поскольку конфигурация *логических каналов связи* меняется в процессе обработки sc-конструкций, то целесообразно говорить также о разработке алгоритмов переразмещения ("дефрагментации") уже записанной в процессоро-память sc-конструкции с целью обеспечения последующей эффективности передачи сообщений. Такое переразмещение может выполняться, например, по расписанию в период, когда процессоро-память не используется для решения других задач;
- Кроме того, при наличии аппаратной возможности может выполняться также перекоммутация *физических каналов связи* с целью приближения их конфигурации к конфигурации *логических каналов связи*.

Рассмотрим пример оптимального варианта записи простейшей *пятиэлементной sc-конструкции* в предлагаемую процессоро-память в рамках мелкозернистой архитектуры *ассоциативных семантических компьютеров*.

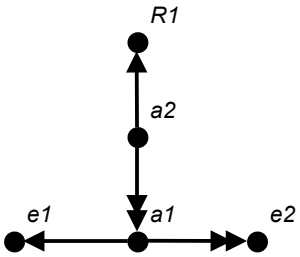
На рисунке *SCg-текст. Пример пятиэлементной sc-конструкции* показана запись некоторой *пятиэлементной sc-конструкции* в SCg-коде.

SCg-текст. Пример пятиэлементной sc-конструкции
=



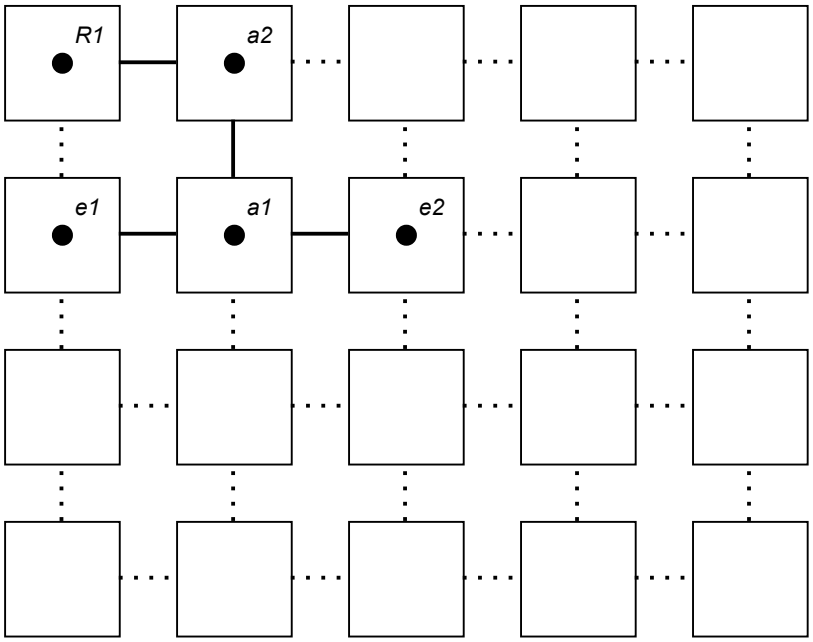
На рисунке *Рисунок. Граф инцидентности для пятиэлементной sc-конструкции* показан граф инцидентности для той же *пятиэлементной sc-конструкции*, который позволяет свести sc-конструкцию к классическому графу с двумя типами связей. Для наглядности синтаксические типы соответствующих sc-элементов на рисунке не показаны.

Рисунок. Граф инцидентности для пятиэлементной sc-конструкции
=



На рисунке *Рисунок. Пример укладки sc-конструкции в процессоро-память* показан один из возможных оптимальных вариантов записи полученного графа инцидентности в процессоро-память. Пунктирными линиями показаны *физические каналы связи* между процессорными элементами, сплошными — *физические каналы связи*, соответствующие *логическим каналам связи*. Отметим, что элемент ***R1*** целесообразно записать в *процессорный элемент*, соседний с *процессорным элементом*, хранящим элемент ***e1*** или элемент ***e2***, как и показано на рисунке. Благодаря этому процессорные элементы, хранящие указанные sc-элементы, оказываются непосредственно связаны *физическим каналом связи*, что упрощает коммуникацию в случае рассылки сообщений по *физическим каналам связи* без учета *логических каналов связи*.

Рисунок. Пример укладки sc-конструкции в процессоро-память
=



Заключение к Главе 6.2.

В главе рассмотрены недостатки доминирующей в настоящее время фон-Неймановской архитектуры компьютерных систем в качестве основы для построения интеллектуальных компьютерных систем нового поколения, проведен анализ современных подходов к разработке аппаратных архитектур, устраняющих некоторые из указанных недостатков, обоснована необходимость разработки принципиально новых аппаратных архитектур, представляющих собой аппаратный вариант реализации ostis-платформ — *ассоциативных семантических компьютеров*.

Предложены общие принципы, лежащие в основе *ассоциативных семантических компьютеров*, рассмотрены три возможных варианта архитектуры таких компьютеров, представлены их достоинства и недостатки.

Дальнейшее развитие предложенных в главе подходов требует решения ряда задач, как технических, так и организационных:

- Разработка волнового языка для записи микропрограмм, которыми обмениваются процессорные элементы между собой, и которые исполняются этими процессорными элементами;
- Разработка языка для записи программ управления процессами обмена микропрограммами и управления очередью микропрограмм;
- Организация активного участия специалистов в области микроэлектроники в уточнении принципов реализации процессорных элементов и процессоро-памяти в целом, уточнение элементной базы и более низкоуровневых архитектурных особенностей *ассоциативных семантических компьютеров*;
- Разработка алгоритмов оптимизации способов записи sc-конструкций в процессоро-память и переразмещения уже записанной sc-конструкции с целью обеспечения последующей эффективности передачи сообщений между процессорными элементами;
- Уточнение типологии информационных процессов в процессоро-памяти, их свойств и соответствующей типологии меток;
- Уточнение принципов реализации многоагентной обработки знаний в рамках процессоро-памяти, в частности, разработка принципов реализации событийной обработки информации в такой памяти.

Глава 6.3.
Программная платформа ostis-систем

⇒ *автор**:
- *Зотов Н.В.*
- *Шункевич Д.В.*

⇒ *аннотация**:
[В главе рассмотрен один из вариантов программной реализации *ostis-платформы* массовой коллективной разработки и эксплуатации интеллектуальных компьютерных систем нового поколения. Детально иллюстрируется пример использования онтологического и компонентного подходов к разработке и описанию подкласса *программных компьютерных систем*, являющихся системами автоматизации проектирования и реализации других *программных компьютерных систем*. Данная глава является *Документацией Программного варианта реализации ostis-платформы*, включающей описание особенностей, аналогов, а также достоинств и недостатков ее реализации.]

⇒ *подраздел**:
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*
- *§ 6.3.2. Принципы, лежащие в основе, и структура предлагаемой Программной платформы ostis-систем*
- *§ 6.3.3. Реализация sc-памяти в Программной платформе ostis-систем*
- *§ 6.3.4. Программный интерфейс Реализации sc-памяти в ostis-платформе*
- *§ 6.3.5. Реализация файловой памяти в Программной платформе ostis-систем*
- *§ 6.3.6. Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*
- *§ 6.3.7. Реализация интерпретатора sc-моделей пользовательских интерфейсов*

⇒ *ключевой знак**:
- *Программный вариант реализации ostis-платформы*
  ⇒ *часто используемый sc-идентификатор**:
  [Программная платформа ostis-систем]
- *Реализация sc-памяти в ostis-платформе*
- *SCin-код*
- *Программный интерфейс Реализации sc-памяти в ostis-платформе*
- *Реализация файловой памяти в ostis-платформе*
- *SCfin-код*
- *Реализация подсистемы взаимодействия ostis-платформы с внешней средой*
- *SC-JSON-код*
- *Реализация интерпретатора sc-моделей пользовательских интерфейсов*

⇒ *ключевое знание**:
- *Спецификация Программного варианта реализации ostis-платформы*
  ⇒ *часто используемый sc-идентификатор**:
  [Документация Программного варианта реализации ostis-платформы]
  ≔ [База знаний Программного варианта реализации ostis-платформы]

⇒ *библиографическая ссылка**:
- *Iliadis A.T.Tower oBPMDM-2019art*
- *Соколов А.П..СистеАПКМ-2021ст*
- *Dillon T..GridSGSotWE-2007art*
- *Dillon T..OntolBSESE-2008art*
- *Ouksel A.M..SemanIiGIS-1999art*
- *Neiva F.W..TowarPItSC-2016art*
- *Lu K..RethiMCfSCtSC-2022art*
- *Hagoort P..SemanU-2009art*
- *Siekmann J.UniveU-1984art*
- *Lenat D.B..CycTPwCS-1990art*
- *Грибова В.В..Платф дРОИС-2016ст*

- *Филиппов.А.А..ЕдинаяОПИАД-2016ст*
- *Ballinger C.tTeradSS-2009art*
- *Lenat D.B.Cyc aLSIiKI-1995art*
- *Lehmann J..DBpedaLSMKBEfW-2015art*
- *Bai J..fPlatf tKGEoLA-2022art*
- *Robinson I..GraphD-2015bk*
- *Neo4jGDPGDMS-2023el*
- *Абрамский М.М..СравнАИРиГБ-2018ст*
- *Vicknair C..aCompa oaGDaaRD-2010art*
- *Климанская Е.В.СовреПИАО-2014ст*
- *Chen C..MPEoRaGD-2022art*
- *Наумов А.Н..СистеУБДиЗ-1991кн*
- *Гаврилова Т.А..БазыЗИСУ-2000кн*
- *Башлыков А.А.СистеУБЗ-2010ст*
- *Башлыков А.А.МетодПСУБЗ-2013ст*
- *Zapata J..Ontol iSEATGKA-2010art*
- *Корончик Д.Н.РеалиХУСС-2013ст*
- *Ивашенко В.П.Модел иАИЗнООСС-2015ст*
- *Shunkevich.D.V.AgentMMaToC-2018art*
- *SoftwIoSNS-el*
- *Корончик Д.Н.РеалиПWСУ-2015ст*
- *Shunkevich D.V..OntolAttDoaSM-2021art*
- *Zotov N.SoftwPfNGICS-2022art*
- *Fortin S.tGraphIP-1996art*
- *Foggia P..aPerfoCoFAfG-2001art*
- *McKay B.D.NautyUGV24-2007art*
- *Cordella L.P..aSubGIAfLG-2004art*
- *Bayer R..PrefiBT-1977art*
- *Belazzougui D..FastPSiLSwA-2010art*
- *Bhumij G..aOverv oWStFoRT-2018art*
- *Tomasseti M.aAnaly otPoWiVP-2021art*
- *Marrs T.Json aWPDIftW-2017bk*
- *Голенков В.В..СтандОТОП-2021кн*
- *Myers B.A..Surve oUIP-1992art*
- *SoftwIoWOUI-el*
- *Корончик Д.Н.СеманТКПП-2011ст*
- *Корончик Д.Н.СеманММПИ*
- *Корончик Д.Н.УнифиСМПИ-2013ст*
- *Корончик Д.Н.ПользИИМП-2014ст*
- *Sadouski M.SemanDoaAUI-2022art*

Введение в Главу 6.3.

До настоящего времени существует обширное множество различных решений в области автоматизации проектирования и разработки *программных компьютерных систем* (см. *Iliadis A.T.Tower oBPMDM-2019art*), позволяющих решать задачи достаточно серьезного уровня. Однако ни одна из таких систем не способна обеспечить *платформенную независимость*, а значит и возможность к более легкой интеграции создаваемых *программных компьютерных систем*. Актуальность проблемы объясняется необходимостью создания *программных компьютерных систем нового поколения*, способных быстро и качественно решать *задачи* любого *вида деятельности*.

Современные *программные компьютерные системы*, а также средства автоматизации проектирования и разработки таких систем, обладают рядом значительных недостатков:

- Проектируемые *программные компьютерные системы* в значительной степени остаются зависимыми от реализации конкретных платформ, на которых они проектируются, что, в свою очередь, приводит к существенным затратам на приведение в соответствие методов и средств проектирования систем в случае их перехода на новые платформы (см. *§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению*).
- Проектирование и разработка конкретной *программной компьютерной системы* ведется при помощи разных методов и моделей проектирования *программных компьютерных систем*. Тем самым, описание целевого

состояния системы и описание текущей реализации могут не соответствовать друг другу, а интеграция таких решений трудно достижима (см. *Соколов А.П..СистеАПКМ-2021ст*).

- Место спецификации *программных компьютерных систем* отводится на второй план, а иногда и вовсе не предусматривается проектом разработки конкретной компьютерной системы. Следовательно, увеличиваются затраты на поддержание процесса перманентного реинжиниринга таких систем (см. *Dillon T..GridSGSotWE-2007art*, *Dillon T..OntolBSESE-2008art*).
- При разработке современных *программных компьютерных систем* отсутствует понимание необходимости разработки и описания методов проектирования этих систем, в том числе описания процесса реализации, направлений использования и так далее. По этой причине новое поколение разработчиков *программных компьютерных систем* не использует уже имеющийся накопленный опыт, а изобретает одни и те же либо похожие решения.
- Отсутствуют единые универсальные инструментальные средства разработки и реинжиниринга других систем, позволяющие не только автоматизировать их проектирование, но и свести к минимуму саму разработку за счет унификации моделей представления этих систем и наличия семантически мощной *комплексной библиотеки многократно используемых компонентов* (см. *Главу 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*).
- Даже узкоспециализированные *программные компьютерные системы* должны обладать высоким *уровнем интеллекта* для расширения возможностей в решении более сложных задач. *программные компьютерные системы нового поколения* в отличие от современных *программных компьютерных систем*, должны оперировать *смыслом* того, что они знают и обрабатывают: они должны понимать друг друга, находить точки соприкосновения и образовывать коллективы для решения *задач* любого класса (см. *Ouksel A.M..SemanIiGIS-1999art* , *Neiva F.W..TowarPItSC-2016art* , *Lu K..RethiMCfSCtSC-2022art* , *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*).
- Для эффективной реализации даже существующих моделей представления знаний и моделей решения трудно формализуемых задач современные компьютеры оказываются плохо приспособленными, что требует разработки принципиально новых платформ и компьютеров, обеспечивающих унификацию представления этих знаний (см. *Hagoort P..SemanU-2009art*, *Siekmann J.UniveU-1984art*).

Обычно системы такого рода проектируются и разрабатываются для решения узких прикладных задач. В результате образуются системы с описанными выше проблемами. Так, большая часть из описанных проблем, к сожалению, не были решены при создании *систем автоматизации проектирования*, описанных в работах *Lenat D.B..CycTPwCS-1990art*, *Грибова В.В..Платф дРОИС-2016ст*, *Филиппов.А.А..ЕдинаяОПИАД-2016ст*.

При разработке спецификации таких систем важной задачей является задача выбора средств и методов проектирования будущей *ostis-платформы*. Она должна обеспечивать:

- *однозначность интерпретации* и представления *sc-моделей ostis-систем*, обеспечиваемые используемым унифицированным языком представления знаний и онтологии проектирования *ostis-платформ*;
- *семантическую совместимость* *sc-моделей ostis-систем* и их компонентов между собой;
- *платформенную независимость* реализуемых и интерпретируемых на ней *sc-моделей ostis-систем*;
- *простоту* и *гибкость расширения* своих функциональных возможностей;
- *функциональную полноту* для создания *sc-моделей ostis-систем* за счет наличия формальной методологии проектирования ее реализации;
- разделение обязанностей между компонентами *ostis-платформы*.

§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем

⇒ *ключевое понятие**:
- *реляционная база данных*
- *реляционная модель данных*
- *графовая база данных*
- *графовая модель данных*

⇒ *ключевой знак**:
- *Платформа SMILA*
- *Платформа Teradata*
- *Платформа SYC*
- *Платформа Semantic Web*
- *Платформа Neo4j*

⇒ *ключевое знание**:
- *Аналоги Программной платформы ostis-систем*
- *Отличие графовых баз данных от реляционных баз данных*

- *Способы программной реализации ostis-платформы*

Реализация хранилищ данных, используемых в подавляющем большинстве *программных компьютерных систем*, основывается на *реляционной модели данных*. Примерами таких систем для обработки *неструктурированных* и *слабоструктурированных данных* являются: *Платформа SMILA* (*SeMantic Information Logistic Architecture*), *Платформа Teradata* (*Teradata Aster Discovery Platform*, см. *Ballinger C.tTeradSS-2009art*), реализующие реляционную, поколоночную и гибридную модели хранения записей в базе данных с массово-параллельной архитектурой, *Платформа CYC* (см. *Lenat D.B.Cyc aLSIiKI-1995art*) и *Платформа Semantic Web* (см. *Lehmann J..DBpedaLSMKBEfW-2015art*). Перечисленные платформы являются *Аналогами Программной платформы ostis-систем*.

В настоящее время информационные системы подвергаются интенсивной *интеллектуализации*. В первую очередь, это вызвано повышением уровня сложности решаемых задач. *интеллектуализация* информационных систем требует от технологий разработки программных систем учета слабоформализуемой, возможно не полностью определенной, нечеткой, темпоральной, пространственно-распределенной информации и, как следствие, получения *структурированных*, *слабоструктурированных* и *неструктурированных данных*. Увеличение числа интеллектуальных задач обработки больших объемов данных во всех сферах деятельности человека приводит к потребности создания универсальных средств хранения, представления и обработки *сложноструктурированной информации*.

Наличие таких задач стимулирует переход от традиционных *реляционных баз данных* к их графовым аналогам. Это объясняется не столько эффективностью организации памяти и обработки данных в *графовых базах данных*, сколько важностью представления конфигураций связей (то есть *смысла*) между ними (см. *Bai J..fPlatf tKGEoLA-2022art*). Подробное изъяснение принципов организации *графовых данных* в *базах данных* можно найти в работе авторов популярной ***Платформы Neo4j*** (см. *Robinson I..GraphD-2015bk*, *Neo4jGDPGDMS-2023el*).

Для общего понимания всей проблемы, связанной с представлением и обработкой данных и знаний, рассмотрим несколько современных реализаций *графовых моделей данных* в виде программных продуктов. Любая из описанных ниже *баз данных* предназначена для удобного хранения и доступа к данным, представленным в виде *графовых структур*. По организации памяти и процессу обработки данных *графовые базы данных* можно классифицировать на следующие виды:

- *базы данных с локальным хранением и обработкой графов* (*Neo4j*, *HyperGraphDB*, *AllegroGraph*);
- *базы данных с распределенным хранением и обработкой данных* (*Horton*, *InfiniteGraph*);
- *базы данных в формате "ключ-значение"* (*Trinity*, *CloudGraph*, *RedisGraph*, *VertexDB*);
- *документо-ориентированные базы данных* (*OrientDB*);
- *надстройки над SQL-ориентированными базами данных* (*Filament*, *G-store*);
- *графовые базы данных с моделью MapReduce* (*Pregel*, *Apache Giraph*, *GraphLab*).

С подробным описанием каждой из представленных *баз данных* можно ознакомиться в работах, посвященных сравнению *реляционных* и *графовых баз данных*: *Абрамский М.М..СравнАИРиГБ-2018cm* , *Vicknair C..aCompra oaGDaaRD-2010art*, *Климанская Е.В.СовреПИАО-2014cm*, *Chen C..MPEoRaGD-2022art*.

Мотивация перехода от *реляционных баз данных* к *графовым базам данных* объясняется преимуществами организации модели памяти и обработки данных в них:

- Производительность обработки данных улучшается на один или более порядков при представлении данных в виде *графов*, что объясняется свойствами самого *графа*. В отличие от *реляционных баз данных*, где производительность запросов с увеличением интенсивности запросов ухудшается по мере увеличения набора данных, производительность *графовой модели данных* остается постоянной, даже когда набор данных растет. Это связано с тем, что обработка данных локализуется в некоторой части *графа*. В результате время выполнения каждого запроса равно пропорционально только размеру части *графа*, пройденной для удовлетворения этого запроса, а не размеру всего *графа*.
- *графовые модели данных* имеют огромную выразительную силу. *графовые базы данных* предлагают чрезвычайно гибкую модель данных и способ их представления. *графы* аддитивны, это обеспечивает гибкость добавления новых связей между данными, новые узлы и новые подграфы к уже существующей структуре *графа*, не нарушая ее целостности и связности.
- Многообразие форм представления данных минимизируется за счет уменьшения количества синтаксических аспектов, учитываемых при хранении данных и использовании их в базах данных, поскольку *графовые модели данных* позволяют записывать различные *виды знаний* одним и тем же способом.

Таким образом, переход к *графовым базам данных* позволяет повысить производительность обработки *сложноструктурированных данных* и улучшить масштабируемость системы.

Не смотря на все преимущества *графовых баз данных* по сравнению с *реляционными базами данных*, *программные компьютерные системы нового поколения* в силу своих свойств (см. *Главу 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*) должны оперировать не просто *данными*, а *знаниями*. Чтобы понимать *смысл* знаний, необходимо представлять эти знания в понятной форме для каждого: и для человека, и для системы. Говоря об унификации представления всех *видов знаний*, важным считается использование *графовых баз данных*

не просто как средств для хранения *структурированных данных*, а для хранения *семантически целостных* и *связанных* между собой знаний.

Стоит также отметить, что акцент ставится не на разработку *программных компьютерных систем* поддержки проектирования других *программных компьютерных систем*, а на разработку *комплексных инструментальных средств поддержки автоматического проектирования интеллектуальных компьютерных систем нового поколения*. Такие инструментальные средства можно сравнивать с *системами управления базами знаний* (см. *Наумов А.Н..СистеУБДиЗ-1991кн* , *Гаврилова Т.А..БазыЗИСУ-2000кн* , *Башлыков А.А.СистеУБЗ-2010ст* , *Башлыков А.А.МетодПСУБЗ-2013ст*).

В основе *ostis-платформы* должны лежать основополагающие принципы:

- Все тексты, представляемые на *SC-коде*, представляют собой *графовые конструкции*. Поэтому задача разработки *Программного варианта реализации ostis-платформы* сводится к разработке средств хранения и обработки таких *графовых конструкций*. Другими словами, *ostis-платформа* должна обеспечивать функционально полную и однозначную интерпретацию хранимых *графовых конструкций*.
- Проектирование *ostis-платформы*, в том числе ее компонентов, должно четко специфицироваться и формулироваться в рамках моделей, методов и средств описания сложных систем, предлагаемых *Технологией OSTIS*. Именно *онтологический подход* к проектированию, эксплуатации и реинжинирингу такого подкласса *программных компьютерных систем* позволит эффективно и универсально разрабатывать другие *ostis-системы* самого различного назначения *Zapata J..Ontol iSEATGKA-2010art*.

Спецификация такого сложного программного объекта, как *ostis-платформа*, должна быть представлена на формальном *языке представления знаний*, в данном случае на *SC-коде*, тексты которого она хранит и обрабатывает. Язык, который должен описывать *программную реализацию ostis-платформы*, должен являться *подъязыком* SC-кода*, то есть должен наследовать все свойства *Синтаксиса* и *Денотационной семантики SC-кода*. Такая модель представления спецификации *программных компьютерных систем* дает безусловно сильные преимущества по сравнению с другими возможными вариантами представления спецификаций:

- Язык, тексты которого система хранит и обрабатывает, и язык спецификации того, как система представляет тексты первого языка в памяти самой себя, являются подмножествами одного и того же языка. Это упрощает не только становление понимания разработчика, который разрабатывает сложную *программную компьютерную систему*, за счет того, что форма представления обрабатываемого этой системой языка и языка ее спецификации унифицирована, но и позволяет открыть для этой системы новые функциональные возможности в познании самой себя. Таким образом, такой подход позволяет повышать качество *интеллектуальной компьютерной системы*, например, *способность к интроспекции*.
- Нельзя проектировать и реализовывать *интеллектуальные компьютерные системы* на *программной компьютерной системе*, которая сама таковой не является. Представление спецификации системы в такой форме позволяет повысить уровень ее *интеллекта*.
- Нет необходимости в создании дополнительных средств для верификации и анализа работы всей системы, поскольку форма представления языка описания системы унифицирована с языком, тексты которого она хранит и обрабатывает.

В таком случае, это позволяет не только уменьшить количество используемых средств при проектировании и реализации *ostis-платформы*, но и позволяет унифицировать информацию, хранимую в *ostis-платформе* и описывающую *ostis-платформу*, с целью использования этой информации в процессе эволюции компонентов *ostis-платформы*. При этом спецификация *ostis-платформы* остается платформенно-независимой, поэтому при смене одного варианта реализации *ostis-платформы* на другой подход к описанию *ostis-платформы* остается одним тем же.

§ 6.3.2. Принципы, лежащие в основе, и структура предлагаемой Программной платформы ostis-систем

⇒ *подраздел**:
- *Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем*
- *Пункт 6.3.2.2. Принципы документирования Программной платформы ostis-систем*

⇒ *ключевой знак**:
- *Программный вариант реализации ostis-платформы*
- *Реализация памяти ostis-платформы*
- *Спецификация Программного варианта реализации ostis-платформы*

⇒ *ключевое знание**:
- *Обоснование разработки Программного варианта реализации ostis-платформы*
- *Структура Программного варианта реализации ostis-платформы*

- *Принципы, лежащие в основе Программного варианта реализации ostis-платформы*
- *Принципы документирования Программного варианта реализации ostis-платформы*
- *Отличие Программного варианта реализации ostis-платформы от других программных компьютерных систем*

Одним из путей, позволяющих осуществлять апробацию, развитие, а в ряде случаев и внедрение новых моделей и технологий вне зависимости от наличия соответствующих аппаратных средств, является разработка программных моделей этих аппаратных средств, которые были бы функционально эквивалентны этим аппаратным средствам, но при этом интерпретировались на базе традиционной аппаратной архитектуры (в данной работе традиционной архитектурой будем считать *архитектуру фон Неймана*, как доминирующую в настоящее время). Очевидно, что производительность таких программных моделей в общем случае будет ниже, чем самих аппаратных решений, однако в большинстве случаев она оказывается достаточной для того, чтобы развивать соответствующую технологию параллельно с разработкой аппаратных средств и осуществления постепенного перевода уже работающих *программных компьютерных систем* с программной модели на аппаратные средства.

Популярность и развитость *графовых баз данных* приводит к тому, что на первый взгляд целесообразным и эффективным кажется разработка *Программного варианта реализации ostis-платформы* на базе одного из таких средств. Однако, существует ряд причин, в силу которых это сделать невозможно. К ним относятся следующие:

- Для обеспечения эффективности хранения и обработки *информационных конструкций* определенного вида (в данном случае — *sc-конструкций*), должна учитываться специфика этих конструкций. В частности, описанные в работе *Корончик Д.Н.РеалиХУСС-2013ст* эксперименты показали значительный прирост эффективности собственного решения по сравнению с существующими на тот момент.
- В отличие от классических *графовых конструкций*, где *дуга* или *ребро* могут быть инцидентны только *узлу графа* (это справедливо и для *rdf-графов*) в *SC-коде* вполне типичной является ситуация, когда *sc-коннектор* инцидентен другому *sc-коннектору* или даже двум *sc-коннекторам* (см. *§ 2.2.2. Синтаксис SC-кода*). В связи с этим существующие средства хранения *графовых конструкций* не позволяют в явном виде хранить *sc-конструкции* (*sc-графы*). Данная проблема также решаема при переходе от *неориентированного графа* к *орграфу* (см. *Ивашенко В.П.Модел иАИЗнООСС-2015ст*).
- В основе обработки информации в рамках *Технологии OSTIS* лежит *многоагентный подход* (см. *§ 1.1.2. Понятие интеллектуальной многоагентной системы*), в рамках которого агенты обработки информации, хранимой в sc-памяти (sc-агенты) реагируют на события, происходящие в sc-памяти и обмениваются информацией посредством спецификации выполняемых ими действий в sc-памяти *Shunkevich.D.V.AgentMMaToC-2018art*. В связи с этим одной из важнейших задач является реализация в рамках *Программного варианта реализации ostis-платформы* возможности подписки на события, происходящие в программной модели sc-памяти, которая на данный момент практически не поддерживается в рамках современных средств хранения и обработки графовых конструкций.
- *SC-код* позволяет описывать также внешние *информационные конструкции* любого рода (изображения, текстовые файла, аудио- и видеофайлы и так далее (см. *Главу 2.1. Информационные конструкции и языки*)), которые формально трактуются как содержимое *sc-элементов*, являющихся знаками *внешних файлов ostis-системы*. Таким образом, компонентом *Программного варианта ostis-платформы* должна быть реализация файловой памяти, которая позволяет хранить указанные конструкции в каких-либо общепринятых форматах. Реализация такого компонента в рамках современных средств хранения и обработки *графовых конструкций* также не всегда представляется возможной.

По совокупности перечисленных причин было принято решение о реализации *Программного варианта реализации ostis-платформы* "с нуля" с учетом особенностей хранения и обработки информации в рамках *Технологии OSTIS*.

Программный вариант реализации ostis-платформы

:= [Реализация sc-машины]

⇒ *часто используемый sc-идентификатор**:
[Программная платформа ostis-систем]

:= [Базовая программная платформа для массового создания интеллектуальных компьютерных систем нового поколения]

:= [Предлагаемый нами программный вариант реализации ассоциативного семантического компьютера]

:= [sc-machine]

∈ *специализированная ostis-платформа*

⇐ *ключевой знак**:

- *§ 6.1.3. Уточнение понятия ostis-платформы*

∈ *web-ориентированный вариант реализации ostis-платформы*

:= [вариант реализации платформы интерпретации sc-моделей компьютерных систем, предполагающий взаимодействие пользователей с системой посредством сети Интернет]

∈ *многопользовательский вариант реализации ostis-платформы*

⇐ *ключевой знак**:

- *§ 6.1.3. Уточнение понятия ostis-платформы*

∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
⇐ *ключевой знак**:
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

∈ *неатомарный многократно используемый компонент ostis-систем*
⇐ *ключевой знак**:
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

∈ *зависимый многократно используемый компонент ostis-систем*
⇐ *ключевой знак**:
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-machine]
⇒ *декомпозиция программной системы**:
{
- *Реализация памяти ostis-платформы*
- *Реализация подсистемы взаимодействия с внешней средой с использованием языков сетевого взаимодействия*
- *Реализация интерпретатора sc-моделей пользовательских интерфейсов*
- *Реализация базового набора платформенно-зависимых sc-агентов и их общих компонентов*
- *Реализация менеджера многократно используемых компонентов ostis-систем*
  ⇐ *ключевой знак**:
  - *§ 5.1.4. Менеджер многократно используемых компонентов ostis-систем*

}
⇒ *зависимости компонента**:
{
- *Реализация памяти ostis-платформы*
- *Реализация подсистемы взаимодействия с внешней средой с использованием языков сетевого взаимодействия*
- *Реализация интерпретатора sc-моделей пользовательских интерфейсов*

}

Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем

Программный вариант реализации ostis-платформы
⇒ *принципы, лежащие в основе**:
- Текущий *Программный вариант реализации ostis-платформы* является web-ориентированным, поэтому с этой точки зрения каждая ostis-система представляет собой web-сайт, доступный онлайн посредством обычного браузера. Такой вариант реализации обладает очевидным преимуществом — доступ к системе возможен из любой точки мира, где есть Интернет, при этом для работы с системой не требуется никакого специализированного программного обеспечения. С другой стороны, такой вариант реализации обеспечивает возможность параллельной работы нескольких пользователей с системой.
- Реализация является *кроссплатформенной* и может быть собрана из исходных текстов в различных *операционных системах*. В то же время, взаимодействие клиентской и серверной части организовано таким образом, что web-интерфейс может быть легко заменен на настольный или мобильный интерфейс, как универсальный, так и специализированный.
- Текущий вариант реализации ostis-платформы является специализированным, то есть не включает *Реализацию интерпретатора Языка SCP* (см. *§ 3.3.5. Базовый язык программирования ostis-систем*). На текущем этапе разработки *Программного варианта реализации ostis-платформы* все функционирующие *ostis-системы* являются *платформенно-зависимыми*. Данная проблема прежде всего связана с недостатками выбранной и реализованной модели управления доступа к sc-памяти, что не позволяет в полной мере создавать распределенные коллективы *sc-агентов*, работающих над sc-памятью.
- Ядром платформы является ***Реализация памяти ostis-платформы***, которая одновременно может взаимодействовать как с *Реализацией интерпретатора sc-моделей пользовательских интерфейсов*, так и с любыми сторонними приложениями по соответствующим языкам сетевого взаимодействия (сетевым протоколам). С точки зрения общей архитектуры *Реализация интерпретатора sc-моделей пользовательских интерфейсов* выступает как один из множества возможных внешних компонентов, взаимодействующих с *Реализацией памяти ostis-платформы* по сети. Текущая *Реализация интерпретатора*

sc-моделей пользовательских интерфейсов является *платформенно-зависимой*, поскольку не в полной мере реализован интерпретатор базового *Языка SCP*.

- Текущая *Реализация памяти ostis-платформы* функционально полна, то есть позволяет хранить и представлять *sc-конструкции*, с помощью которых описывается любая *sc-модель ostis-системы*, внешние *информационные конструкции*, не принадлежащие *SC-коду*, а также предоставлять различные уровни доступа для обработки этих конструкций. В контексте текущего *Программного варианта реализации ostis-платформы Реализация памяти ostis-платформы* состоит из таких компонентов как: *Реализация sc-памяти в ostis-платформе*, внутри которой представляются sc-конструкции *sc-моделей ostis-систем*, *Реализация файловой памяти ostis-платформы*, внутри которой представляются внешние *информационные конструкции*, не принадлежащие *SC-коду*, то есть содержимое *внутренних файлов ostis-системы*, но дополнительно описывающие, поясняющие и детализирующие *sc-конструкции sc-моделей ostis-систем*.
- Текущий *Программный вариант реализации ostis-платформы* включает *Реализацию менеджера многократно используемых компонентов ostis-систем*. Это связано с тем, что текущая *Реализация менеджера многократно используемых компонентов ostis-систем* использует *Реализацию памяти ostis-платформы* для хранения и обработки спецификации устанавливаемых компонентов, вне зависимости от языка их реализации. Полная спецификация текущей *Реализации менеджера многократно используемых компонентов ostis-систем* находится в *§ 5.1.4. Менеджер многократно используемых компонентов ostis-систем*.

Реализация памяти ostis-платформы
:= [Реализация sc-памяти и файловой памяти ostis-платформы]
:= [Предлагаемый нами программный вариант реализации sc-памяти и файловой памяти ostis-платформы]
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
∈ *неатомарный многократно используемый компонент ostis-систем*
∈ *зависимый многократно используемый компонент ostis-систем*
⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-machine/tree/main/sc-memory]
⇒ *декомпозиция программной системы**:
{• *Реализация sc-памяти в ostis-платформе*
• *Реализация файловой памяти ostis-платформы*
}
⇒ *зависимости компонента**:
{• *Библиотека методов и структур данных GLib*
• *Библиотека методов и структур данных C++ Standard Library*
• *Реализация sc-памяти ostis-платформы*
• *Реализация файловой памяти ostis-платформы*
}

Стоит отметить, что при переходе с *Реализации памяти ostis-платформы* на ее аппаратную реализацию *файловую память ostis-системы* целесообразно будет реализовывать на основе традиционной линейной памяти (во всяком случае, на первых этапах развития *ассоциативного семантического компьютера* (см. *Главу 6.2. Ассоциативные семантические компьютеры для ostis-систем*)). Текущий вариант *Реализации памяти ostis-платформы* является открытым и доступен на *SoftwIoSNS-el* . Спецификацию компонентов текущего *Программного варианта реализации ostis-платформы*, а также оценку их производительности можно найти в работах *Корончик Д.Н.РеалиПWСУ-2015cm*, *Shunkevich D.V..OntolAttDoaSM-2021art* и *Zotov N.SoftwPfNGICS-2022art*.

Принципы, лежащие в основе Программного варианта реализации ostis-платформы являются только базовыми, все компоненты, входящие в состав *Программного варианта реализации ostis-платформы* имеют свои особенности реализации, а также аналоги, которые необходимо учитывать при реализации всей *ostis-платформы*.

Пункт 6.3.2.2. Принципы документирования Программной платформы ostis-систем

Перманентный реинжиниринг компонентов текущего *Программного варианта реализации ostis-платформы* обеспечивается открытой командой разработчиков, при этом каждый разрабатываемый компонент документируется согласно общепринятым принципам.

Программный вариант реализации ostis-платформы
⇒ *принципы документирования**:
- Вне зависимости от языка реализации каждого компонента *Программного варианта реализации ostis-платформы*, спецификация каждого компонента включает: (1) спецификацию, непосредственно описанную в исходных файлах самого компонента, описывающую программный интерфейс этого компонента,

(2) а также спецификацию как части базы знаний ostis-платформы, детально описывающую реализацию этого компонента, в том числе используемые алгоритмы. При этом дублирование спецификации компонентов *Программного варианта реализации ostis-платформы* категорически запрещается. Так, например, спецификация, непосредственно находящаяся в исходном файле с реализацией самих компонентов, описывает особенности применения компонентов с точки зрения внешнего или внутреннего (то есть входящего в состав команды) разработчика, а спецификация, являющаяся частью *sc-текста базы знаний Программного варианта реализации ostis-платформы*, дополнительно включает особенности, предлагаемые подходы к реализации, а также достоинства и недостатки входящих в состав компонентов.

- Каждый компонент *Программного варианта реализации ostis-платформы* описывается средствами *Технологии OSTIS*, то есть на *SC-коде*, тексты которого она обрабатывает и хранит. Таким образом, это дает возможности платформе анализировать свое состояние и способствовать поддержанию своего жизненного цикла без участия ее разработчиков. *Программный вариант реализации ostis-платформы* выступает полноценным субъектом, принимающим непосредственное участие в собственной разработке.
- *Спецификация Программного варианта реализации ostis-платформы* представляет собой *sc-язык*, то есть подъязык *SC-кода*, для которого уточнены *Синтаксис* и *Денотационная семантика SC-кода* (см. *§ 2.2.2. Синтаксис SC-кода, § 2.2.1. Базовая денотационная семантика SC-кода*). Этот *sc-язык* можно представить в виде некоторого семейства более частных sc-языков, которые позволяют описывать:
  - то, как *sc-конструкции* представляются внутри *sc-памяти ostis-платформы*;
  - то, как *информационные конструкции*, которые не принадлежат *SC-коду*, представляются внутри *файловой памяти ostis-платформы*;
  - то, как различные подсистемы *ostis-платформы*, взаимодействуют между собой;
  - то, какие методы и соответствующие им агенты взаимодействуют с *sc-памятью ostis-платформы*;
  - то, как представляются и работают различного рода интерпретаторы *sc-моделей ostis-систем* (базы знаний, решателя задач, интерфейса);
  - и так далее.

  Такой подход позволяет без особых препятствий интегрировать описания различных компонентов, входящих в состав *Программного варианта реализации ostis-платформы*, поскольку вся *Спецификация Программного варианта реализации ostis-платформы* является ее базой знаний с четко выделенной иерархией предметных областей и онтологий (то есть *sc-языков*, описывающих ее реализацию).
- Каждый разработчик *Программного варианта реализации ostis-платформы* заботится о перманентной поддержки не только состояния ее компонентов, но и спецификации этих компонентов. Гарантом качественной *Спецификации Программного варианта реализации ostis-платформы* является ее коллектив разработчиков, способных не только понимать детали реализации *ostis-платформы*, но и способствовать к созданию взаимовыгодного сотрудничества для достижения поставленных целей.

Данные принципы можно использовать при описании любых других *программных компьютерных систем*, в том числе тех *программных компьютерных систем*, которые не реализуются на данной *ostis-платформе*.

§ 6.3.3. Реализация sc-памяти в Программной платформе ostis-систем

⇒ *подраздел**:
- *Пункт 6.3.3.1. Спецификация Метаязыка описания представления sc-конструкций в sc-памяти ostis-платформы. SCin-код*
- *Пункт 6.3.3.2. Алфавит и Синтаксис SCin-кода*
- *Пункт 6.3.3.3. Денотационная семантика SCin-кода*
- *Пункт 6.3.3.4. Достоинства и недостатки текущего варианта Реализации sc-памяти в ostis-платформе и Метаязыка описания представления sc-конструкций в sc-памяти ostis-платформы*

⇒ *ключевой знак**:
- *Реализация sc-памяти в ostis-платформе*
- *SCin-код*
  ≔ [sc.in-текст]

⇒ *ключевое знание**:
- *Спецификация Реализации sc-памяти в ostis-платформе*
- *Принципы, лежащие в основе Реализации sc-памяти в ostis-платформе*
- *Ответ на Вопрос. Как кодируются sc-конструкции в текущей Реализации sc-памяти в ostis-платформе*
  ≔ [Спецификация Метаязыка описания представления sc-конструкций в sc-памяти ostis-платформы]
- *Достоинства и недостатки Реализации sc-памяти в ostis-платформе*

Под текущей ***Реализацией sc-памяти ostis-платформы*** понимается компонент программной модели, осуществляющий хранение sc-конструкций и доступ к ним через *программный интерфейс*.

В рамках данного *Программного варианта реализации ostis-платформы sc-память* моделируется в виде набора *сегментов*, каждый из которых представляет собой фиксированного размера упорядоченную последовательность *элементов sc-памяти*, каждый из которых соответствует конкретному *sc-элементу*. В настоящее время каждый сегмент состоит из $2^{16} - 1 = 65535$ *элементов sc-памяти*. Каждый сегмент состоит из набора структур данных, описывающих конкретные *sc-элементы* (элементов sc-памяти). Независимо от типа описываемого sc-элемента каждый *элемент sc-памяти* имеет фиксированный размер (в текущий момент — 36 байт), что обеспечивает удобство их хранения.

Выделение *сегментов sc-памяти* позволяет, с одной стороны, упростить адресный доступ к *элементам sc-памяти*, с другой стороны — реализовать возможность выгрузки части *sc-памяти* из *оперативной памяти* на *файловую систему* при необходимости. Во втором случае сегмент *sc-памяти* становится минимальной (атомарной) выгружаемой частью sc-памяти. Механизм выгрузки сегментов реализуется в соответствии с существующими принципами организации виртуальной памяти в современных *операционных системах*.

Максимально возможное число сегментов ограничивается настройками текущей *Реализации sc-памяти в ostis-платформе* (в настоящее время по умолчанию установлено количество $2^{16} - 1 = 65535$ sc-сегментов, но в общем случае оно может быть другим). Таким образом, технически максимальное количество хранимых *sc-элементов* в текущей реализации составляет около 4.3×10^9 *sc-элементов*. По умолчанию все сегменты физически располагаются в *оперативной памяти*, если объема памяти не хватает, то предусмотрен механизм выгрузки части *sc-сегментов* на жесткий диск (механизм виртуальной памяти).

Текущий вариант *Реализации sc-памяти в ostis-платформе* предполагает возможность сохранения состояния (слепка) *sc-памяти* на жесткий диск и последующей загрузки из ранее сохраненного состояния. Такая возможность необходима для перезапуска системы, в случае возможных сбоев, а также при работе с исходными текстами *базы знаний*, когда сборка из исходных текстов сводится к формированию слепка состояния памяти, который затем помещается в *Реализации sc-памяти в ostis-платформе*.

Реализация sc-памяти в ostis-платформе

:= [Программный вариант реализации графодинамической ассоциативной памяти в Программной платформе ostis-систем]

:= [Предлагаемый нами вариант реализации графодинамической ассоциативной памяти для ostis-систем]

∈ *реализация sc-памяти*

∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*

∈ *атомарный многократно используемый компонент ostis-систем*

∈ *зависимый многократно используемый компонент ostis-систем*

⇐ *программная модель**:

sc-память

⇐ *ключевой знак**:

- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

⇐ *семейство подмножеств**:

сегмент sc-памяти

:= [страница sc-памяти]

⇐ *семейство подмножеств**:

элемент sc-памяти

⇒ *зависимости компонента**:

{• *Библиотека методов и структур данных GLib*

• *Библиотека методов и структур данных C++ Standard Library*

}

⇒ *используемый язык программирования**:

- *C*
- *C++*

⇒ *внутренний язык**:

- *SCin-код*

В общем случае *sc-память* может быть реализована по-разному. Так, например, другой вариант *sc-памяти ostis-платформы* можно реализовать при помощи программной реализации *Платформы Neo4j*. Отличие такого возможного варианта реализации *sc-памяти* от текущего состоит в том, что хранение *графовых конструкций* и управление потоком действий над ними должно осуществляться в большей мере средствами, предоставляемыми *Платформой Neo4j*, в то же время представление *графовых конструкций* должно реализовываться по-своему, поскольку зависит от *Синтаксиса SC-кода*.

Такую модель sc-памяти достаточно просто описывать на *sc-языке*, то есть на подъязыке *SC-кода*. Такой язык позволяет описывать то, как внутри памяти *ostis-платформы* представляются тексты языка, на этом же языке. При этом соблюдается не только унификация представления информации, обрабатываемой *ostis-платформой*, и информации, описывающей саму *ostis-платформу*, но и даются возможности для расширения и использования языка в процессе эволюции *ostis-платформы* и ее компонентов, в том числе в процессе эволюции *Реализации sc-памяти в ostis-платформе*.

Пункт 6.3.3.1. Спецификация Метаязыка описания представления sc-конструкций в sc-памяти ostis-платформы. SCin-код

⇒ *ключевой знак**:
- *Реализация sc-памяти в ostis-платформе*
- *SCin-код*
  ≔ [sc.in-текст]
- *Алфавит SCin-кода^*
- *Синтаксис SCin-кода*
- *элемент sc-памяти*
- *Денотационная семантика SCin-кода*

⇒ *ключевое знание**:
- *Отличие SCin-кода от SC-кода*

Раннее упоминалось об *sc-языках*, которые используются при описании компонентов текущего *Программного варианта реализации ostis-платформы*. Один из таких *sc-языков* описывает то, как *sc-конструкции* хранятся внутри *sc-памяти ostis-платформы*. Такой язык называется ***Метаязыком описания представления sc-конструкций в sc-памяти ostis-платформы***, или, кратко, ***SCin-кодом*** *(Semantic Code interior)*. *sc-память* текстов *SC-кода* (как некоторую абстрактную модель, по которой можно реализовать *sc-память ostis-платформы*) можно рассматривать как подмножество *sc.in-текста*.

SCin-код
≔ [Semantic Code interior]
≔ [Язык описания представления SC-кода внутри sc-памяти ostis-платформы]
≔ [Метаязык описания представления sc-конструкций в sc-памяти ostis-платформы]
⇒ *часто используемый sc-идентификатор**:
[sc.in-текст]
∈ *имя нарицательное*
∈ *абстрактный язык*
∈ *метаязык*
∈ *sc-язык*
⊂ *SC-код*
⊃ *sc-память*

следует отличать*
∋ {• *SC-код*
≔ [Универсальный язык внутреннего смыслового представления знаний в памяти ostis-систем]
• *SCin-код*
≔ [Метаязык описания представления SC-кода в sc-памяти ostis-платформы]
⊂ *SC-код*
}

Пункт 6.3.3.2. Алфавит и Синтаксис SCin-кода

⇒ *ключевой знак**:
- *элемент sc-памяти*
- *элемент sc-памяти, соответствующий sc-узлу*
- *элемент sc-памяти, соответствующий sc-коннектору*
- *элемент sc-памяти, имеющий нулевой sc-адрес*
- *класс элемента sc-памяти^*

- *синтаксический класс элемента sc-памяти^*
- *семантический класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*
- *sc-адрес элемента sc-памяти**

⇒ *ключевое знание**:
- *Синтаксическая классификация элементов SCin-кода*

Одним из замечательных достоинств *SC-кода* является то, что синтаксис любого его *подъязыка* наследует свойства *Синтаксиса SC-кода*. То есть форма представления различных знаний, описываемых его подъязыками, остается одна и та же. В данном случае, *Синтаксис SCin-кода* не является исключением и уточняет семантику *Синтаксиса SC-кода*. При этом *Синтаксис SCin-кода*, как и синтаксис любого другого sc-языка, является частью *Денотационной семантики SCin-кода*. То есть *Синтаксис SCin-кода* описывает семантику того, как формируются sc.in-тексты при помощи *Синтаксических правил SCin-кода*. ***Синтаксис SCin-кода*** задается: (1) *Алфавитом SCin-кода*, (2) отношением инцидентности *sc-адрес элемента sc-памяти**. *Алфавит SCin-кода^*, а также отношение инцидентности *sc-адрес элемента sc-памяти** являются подмножествами *Алфавит SC-кода^* и отношений инцидентности *SC-кода* (*см. Пункт 2.2.1.1. Семантическая классификация sc-элементов по базовым признакам*) соответственно, при этом правила *Синтаксиса SCin-кода* включают правила *Синтаксиса SC-кода* и правила, которые уточняют нюансы *Синтаксиса SCin-кода*.

Алфавит SCin-кода^
:= [синтаксический класс элемента sc-памяти]
:= [синтаксический тип элемента sc-памяти]
:= [Множество классов элементов sc-памяти]
:= [Множество типов элементов sc-памяти]
⇐ *алфавит**:
SCin-код
= {• *элемент sc-памяти, соответствующий sc-узлу*
• *элемент sc-памяти, соответствующий sc-коннектору*
• *элемент sc-памяти, имеющий нулевой sc-адрес*
∈ *синглетон*
}

Алфавит SCin-кода^ состоит из трех синтаксически выделяемых классов элементов sc-памяти: *элемента sc-памяти, соответствующего sc-узлу*, *элемента sc-памяти, соответствующего sc-коннектору*, и *элемента sc-памяти, имеющего нулевой sc-адрес*. Такой алфавит не только позволяет задавать в *sc-памяти* минимальный набор объектов, с которым можно производить вычислительные операции, но и, при необходимости, удобен для расширения. Так, например, даннный алфавит языка можно расширить, добавив в него *элемент sc-памяти, соответствующий внутреннему файлу ostis-системы*, либо *элемент sc-памяти, соответствующий sc-ребру*.

элемент sc-памяти
:= [элемент sc-памяти, соответствующий sc-элементу]
:= [ячейка sc-памяти]
:= [образ sc-элемента в рамках sc-памяти]
:= [структура данных, каждый экземпляр которой в рамках sc-памяти соответствует одному sc-элементу]
:= [sc_element]
∈ *C*
∈ *sc-элемент*
⊃ *sc-элемент*
⇒ *разбиение**:
*Алфавит SCin-кода^*

Отношение ***sc-адрес элемента sc-памяти**** определяется как *взаимно однозначное соответствие*, первым компонентом каждой ориентированной пары которого является некоторый элемент sc-памяти, соответствующей некоторому sc-элементу, а вторым компонентом является sc-адрес этого *элемента sc-памяти*. То есть каждый *элемент sc-памяти* имеет уникальный *sc-адрес* как некоторый идентификатор, с помощью которого определяется уникальность *элемента sc-памяти*.

Каждый ***элемент sc-памяти*** в текущей *Реализации sc-памяти в ostis-платформе* может быть однозначно задан его *sc-адресом*, состоящим из *номера sc-сегмента* и *номера элемента sc-памяти* в рамках sc-сегмента. Таким образом, *sc-адрес* служит уникальными координатами *элемента sc-памяти* в рамках *Реализации sc-памяти*. Для каждого *sc-адреса элемента sc-памяти* можно *взаимно однозначно* поставить в соответствие некоторый *хэш*, полученный в результате применения специальной *хэш-функции** над этим *sc-адресом элемента sc-памяти*. *хэш*

является *неотрицательным целым числом* и является результатом преобразования *номера sc-сегмента sc-памяти si*, в котором располагается *элемент sc-памяти*, и *номера* этого *элемента sc-памяти ei в рамках* этого sc-сегмента *si*. В рамках *sc-памяти* используется единственная *хеш-функция** для получения *хеша sc-адреса элемента sc-памяти* и задается как $f(si, ei) = si << 16|ei\&0xffff$, где операция $<<$ — операция *битового сдвига влево** левого аргумента на количество единиц, заданное правым аргументом, относительно этой операции, операция $|$ — операция *битового сложения**, операция $\&$ — операция *битового умножения**, число $0xffff$ — *Число 65535*, представленное в шестнадцатеричном виде и обозначающее максимальное количество элементов в одном *sc-сегменте sc-памяти*. Числовое выражение *sc-адреса* является *32-битовым целым числом*.

sc-адрес элемента sc-памяти
:= [адрес элемента sc-памяти, соответствующего заданному sc-элементу, в рамках текущего состояния реализации sc-памяти в составе программной модели sc-памяти]
:= [sc_addr]
∈ *C*
:= [ScAddr]
∈ *C++*
∈ *32-битовое целое число*

В рамках любой *реализации sc-памяти* должен существовать набор *синтаксических* и *семантических классов элементов sc-памяти^* (меток), которые:
- задают тип элемента на уровне *ostis-платформы* и не имеют соответствующей *sc-дуги принадлежности* (а точнее — *базовой sc-дуги*), явно хранимой в рамках sc-памяти (ее наличие подразумевается, однако она не хранится явно, поскольку это приведет к бесконечному увеличению числа элементов, которые необходимо хранить в *sc-памяти*);
- могут быть представлены в виде параметров соответствующих *элементов sc-памяти*, то есть множеством таких элементов, каждый из которых имеет "метку", выраженную некоторым числовым значением;
- могут уточнять *класс элементов sc-памяти* с той степенью детализации, которая необходима чтобы, например, при совершении операции поиска с помощью таких *классов элементов sc-памяти^* можно было легко определить класс конкретного из них.

класс элемента sc-памяти^
:= [класс всех синтаксических и семантических классов элементов sc-памяти]
:= [битовая маска, обозначающая синтаксический и семантический класс элемента sc-памяти]
:= [sc_type]
∈ *C*
:= [ScType]
∈ *C++*
⇒ *разбиение**:
{• *синтаксический класс элемента sc-памяти^*
• *семантический класс элемента sc-памяти^*
}

С этой целью, в *SCin-коде* выделяется базовая ***Синтаксическая классификация элементов SCin-кода***. Для того чтобы представлять и хранить любые *sc-конструкции* достаточно иметь только два базовых *класса элементов sc-памяти* (*элемент sc-памяти, соответствующий sc-узлу*, и *элемент sc-памяти, соответствующий sc-коннектору*), при этом остальные *классы элементов sc-памяти* можно добавить в расширенной версии *SCin-кода* и тем самым дореализовать необходимую логику на уровне конкретной реализации *sc-памяти*.

Синтаксическая классификация элементов SCin-кода
⊃=
{

элемент sc-памяти
⇒ *разбиение**:
{• *элемент sc-памяти, соответствующий sc-узлу*
∈ *синтаксический класс элемента sc-памяти^*
• *элемент sc-памяти, соответствующий sc-коннектору*
∈ *синтаксический класс элемента sc-памяти^*
• *элемент sc-памяти, имеющий нулевой sc-адрес*
∈ *синтаксический класс элемента sc-памяти^*
}
}

Стоит отметить, что все ***классы элементов sc-памяти^***, входящие в состав *Синтаксической классификации элементов SCin-кода*, являются синтаксически выделяемыми *классами элементов SCin-кода*, то есть на уровне *Реализации sc-памяти в ostis-платформе* такие программные модели этих элементов представляются по-разному.

Несмотря на то, что отношение *sc-адрес элемента sc-памяти** позволяет полностью описать связи *элементов sc-памяти*, для спецификации представления конструкций SC-кода внутри sc-памяти только одного отношения *sc-адрес элемента sc-памяти** не всегда достаточно, чтобы полностью точно и ясно указывать связи между *элементами sc-памяти*, соответствующими *sc-элементам* этих конструкций. Поэтому на практике при описании представления *sc-конструкций* внутри *sc-памяти* необходимо использовать более частные отношения этого базового отношения, например, такие, как *sc-адрес элемента sc-памяти, соответствующего выходящему sc-коннектору из заданного sc-элемента**, *sc-адрес элемента sc-памяти, соответствующего входящему sc-коннектору в заданный sc-элемент** и *sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-коннектора**.

sc-адрес элемента sc-памяти*

⇒ *разбиение**:

{• *sc-адрес элемента sc-памяти, соответствующего выходящему sc-коннектору из заданного sc-элемента**

• *sc-адрес элемента sc-памяти, соответствующего входящему sc-коннектору в заданный sc-элемент**

• *sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-коннектора**

}

Отношение ***sc-адрес элемента sc-памяти, соответствующего выходящему sc-коннектору из заданного sc-элемента**** определяется как *бинарное ориентированное отношение*, первым компонентом каждой *ориентированной пары* которого является некоторый *элемент sc-памяти, соответствующий некоторому sc-элементу*, из которого выходит заданный *sc-коннектор*, а вторым компонентом этой пары является *sc-адрес элемента sc-памяти, соответствующего* этому выходящему *sc-коннектору*. Частными видами этого отношения являются отношение *sc-адрес элемента sc-памяти, соответствующего начальному выходящему sc-коннектору из заданного sc-элемента**, отношение *sc-адрес элемента sc-памяти, соответствующего следующему выходящему sc-коннектору из заданного sc-элемента** и отношение *sc-адрес элемента sc-памяти, соответствующего предыдущему выходящему sc-коннектору из заданного sc-элемента**.

sc-адрес элемента sc-памяти, соответствующего выходящему sc-коннектору из заданного sc-элемента*

⇒ *разбиение**:

{• *sc-адрес элемента sc-памяти, соответствующего начальному выходящему sc-коннектору из заданного sc-элемента**

• *sc-адрес элемента sc-памяти, соответствующего следующему выходящему sc-коннектору из заданного sc-элемента**

• *sc-адрес элемента sc-памяти, соответствующего предыдущему выходящему sc-коннектору из заданного sc-элемента**

}

Отношение ***sc-адрес элемента sc-памяти, соответствующего входящему sc-коннектору в заданный sc-элемент**** определяется как *бинарное ориентированное отношение*, первым компонентом каждой *ориентированной пары* которого является *некоторый элемент sc-памяти, соответствующий некоторому sc-элементу*, в который входит заданный *sc-коннектор*, а вторым компонентом этой пары является *sc-адрес элемента sc-памяти, соответствующего* этому входящему sc-коннектору. Частными видами этого отношения являются отношение *sc-адрес элемента sc-памяти, соответствующего начальному входящему sc-коннектору в заданный sc-элемент**, отношение *sc-адрес элемента sc-памяти, соответствующего следующему входящему sc-коннектору в заданный sc-элемент** и отношение *sc-адрес элемента sc-памяти, соответствующего предыдущему входящему sc-коннектору в заданный sc-элемент**.

sc-адрес элемента sc-памяти, соответствующего входящему sc-коннектору в заданный sc-элемент*

⇒ *разбиение**:

{• *sc-адрес элемента sc-памяти, соответствующего начальному входящему sc-коннектору в заданный sc-элемент**

• *sc-адрес элемента sc-памяти, соответствующего следующему входящему sc-коннектору в заданный sc-элемент**

• *sc-адрес элемента sc-памяти, соответствующего предыдущему входящему sc-коннектору в заданный sc-элемент**

}

Отношение ***sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-дуги**** определяется как *бинарное ориентированное отношение*, первым компонентом каждой *ориентированной пары* которого является некоторый *элемент sc-памяти, соответствующий некоторому sc-коннектору*, а вторым компонентом является *sc-адрес элемента sc-памяти, соответствующего некоторому* инцидентному этому sc-коннектору sc-элементу. Частными видами этого отношения являются отношение *sc-адрес элемента sc-памяти, соответствующего начальному sc-элементу sc-коннектора** и отношение *sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-коннектора**.

sc-адрес элемента sc-памяти, соответствующего инцидентному sc-элементу sc-коннектора*
⇒ *разбиение**:
{•
*sc-адрес элемента sc-памяти, соответствующего начальному sc-элементу sc-коннектора**
•
*sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-коннектора**
}

sc-адрес элемента sc-памяти никак не учитывается при обработке *базы знаний* на семантическом уровне и необходим только для обеспечения доступа к соответствующей структуре данных, хранящейся в линейной памяти на уровне *Реализации sc-памяти в ostis-платформе*.

В общем случае *sc-адрес элемента sc-памяти, соответствующего заданному sc-элементу*, может меняться, например, при пересборке *базы знаний* из исходных текстов и последующем перезапуске системы. При этом *sc-адрес элемента sc-памяти, соответствующего заданному sc-элементу*, непосредственно в процессе работы системы в текущей реализации меняться не может.

На синтаксические *конструкции SCin-кода*, кроме ограничения самого *SC-кода*, накладываются дополнительные ограничения:

- Для каждого *элемента sc-памяти* взаимно однозначно ставится в соответствие *sc-адрес* этого *элемента sc-памяти*.
- Для каждого *элемента sc-памяти, соответствующего sc-узлу*, существует одна и только одна пара отношения *sc-адрес элемента sc-памяти, соответствующего начальному выходящему sc-коннектору из заданного sc-элемента** и одна и только одна пара отношения *sc-адрес элемента sc-памяти, соответствующего начальному входящему sc-коннектору в заданный sc-элемент**.
- Для каждого *элемента sc-памяти, соответствующего выходящему sc-коннектору из заданного sc-элемента* (*элемента sc-памяти, соответствующего входящему sc-коннектору в заданный sc-элемент*), существует не более чем одна пара отношения *sc-адрес элемента sc-памяти, соответствующего следующему выходящему sc-коннектору из заданного sc-элемента** (*sc-адрес элемента sc-памяти, соответствующего следующему входящему sc-коннектору в заданный sc-элемент**) и не более чем одна пара отношения *sc-адрес элемента sc-памяти, соответствующего предыдущему выходящему sc-коннектору из заданного sc-элемента** (*sc-адрес элемента sc-памяти, соответствующего предыдущему входящему sc-коннектору в заданный sc-элемент**).
- Для каждого *элемента sc-памяти, соответствующего sc-коннектору*, который является вторым компонентом каждой пары отношения *sc-адрес элемента sc-памяти, соответствующего начальному выходящему sc-коннектору из заданного sc-элемента** (*sc-адрес элемента sc-памяти, соответствующего начальному входящему sc-коннектору в заданный sc-элемент**) существует только и только одна пара отношения *sc-адрес элемента sc-памяти, соответствующего следующему выходящему sc-коннектору из заданного sc-элемента** (*sc-адрес элемента sc-памяти, соответствующего следующему входящему sc-коннектору в заданный sc-элемент**).

Пункт 6.3.3.3. Денотационная семантика SCin-кода

⇒ *ключевой знак**:
- *Спецификация элемента sc-памяти, соответствующего sc-узлу*
- *Спецификация элемента sc-памяти, соответствующего sc-коннектору*

⇒ *ключевое знание**:
- *Семантическая классификация элементов SCin-кода*

Для каждого *класса sc-элементов* должна существовать программная модель *класса элементов sc-памяти^*, которая удовлетворяет всем перечисленным требованиям. Поэтому важно, чтобы *Алфавит SCin-кода* изначально был полон, чтобы погрузить не только *sc-конструкции Ядра SC-кода*, но и его расширенных версий. Для этого разработаны ***семантические классы элементов sc-памяти^***, спецификация которых представляется в виде ***Семантической классификации элементов SCin-кода***.

Семантическая классификация элементов SCin-кода
⊃=
{

элемент sc-памяти
⇒ *разбиение**:
Типология элементов sc-памяти по признаку константности^
= {• *элемент sc-памяти, соответствующий sc-константе*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-переменной*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-метапеременной*
∈ *семантический класс элемента sc-памяти*^
}
⇒ *разбиение**:
Типология элементов sc-памяти по признаку постоянности^
= {• *элемент sc-памяти, соответствующий*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий временному sc-элементу*
∈ *семантический класс элемента sc-памяти*^
}
⇒ *разбиение**:
Типология элементов sc-памяти по признаку доступности^
:= [класс уровня доступа к элементу sc-памяти]
= {• *элемент sc-памяти, соответствующий sc-элементу, на котором разрешено право чтения*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-элементу, на котором разрешено право записи*
∈ *семантический класс элемента sc-памяти*^
}
⇒ *включение**:
элемент sc-памяти, соответствующий внутреннему файлу ostis-системы

элемент sc-памяти, соответствующий sc-узлу общего вида
⇒ *разбиение**:
Структурная типология элементов sc-памяти, соответствующих sc-узлам^
= {• *элемент sc-памяти, соответствующий sc-узлу, обозначающему небинарную sc-связку*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-классу*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-узлу, обозначающему класс классов*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-структуре*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-узлу, обозначающему ролевое отношение*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-узлу, обозначающему неролевое отношение*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-узлу, обозначающему первичную сущность*
∈ *семантический класс элемента sc-памяти*^
}
⇒ *разбиение**:
Структурная типология элементов sc-памяти, соответствующих sc-дугам^
= {• *элемент sc-памяти, соответствующий sc-дуге принадлежности*
∈ *семантический класс элемента sc-памяти*^
• *элемент sc-памяти, соответствующий sc-дуге общего вида*
∈ *семантический класс элемента sc-памяти*^
}
⇒ *разбиение**:
Типология элементов sc-памяти, соответствующих sc-дугам принадлежности, по типу обозначаемой принадлежности^
= {• *элемент sc-памяти, соответствующий sc-дуге позитивной принадлежности*
∈ *семантический класс элемента sc-памяти*^

- *элемент sc-памяти, соответствующий sc-дуге нечеткой принадлежности*
  ∈ *семантический класс элемента sc-памяти^*
- *элемент sc-памяти, соответствующий sc-дуге негативной принадлежности*
  ∈ *семантический класс элемента sc-памяти^*

}

}

Все семантически и синтаксически выделяемые *классы элементов sc-памяти^*, а также всевозможные подклассы этих классов являются экземплярами (элементами) sc-класса. Все перечисленные классы на уровне программной *Реализации sc-памяти в ostis-платформе* выражаются в виде битов в битовой строке, описываемой в каждом *элементе sc-памяти*.

В текущий момент *sc-ребра* хранятся так же, как *sc-дуги*, то есть имеют начальный и конечный *sc-элементы*, отличие заключается только в *семантическом классе элемента sc-памяти^*. Это приводит к ряду неудобств при обработке, но *sc-ребра* используются в настоящее время достаточно редко.

Спецификация SCin-кода является объединением спецификацией его *элементов sc-памяти*.

элемент sc-памяти
⇒ *понятие, специфицирующее заданную сущность**:
{}
⊃ *сужение отношения по первому домену(спецификация знака*, элемент sc-памяти, соответствующий sc-узлу)**
⊃ *сужение отношения по первому домену(понятие, специфицирующее заданную сущность знака*, элемент sc-памяти, соответствующий sc-коннектору)**

Каждый *элемент sc-памяти* описывается его синтаксическим типом (меткой), а также независимо от *класса элемента sc-памяти^* указывается *sc-адрес первого входящего в данный sc-элемент sc-коннектора* и *sc-адрес первого выходящего из данного sc-элемента sc-коннектора* (могут быть пустыми, если таких sc-коннекторов нет). Оставшиеся байты в зависимости от *класса элемента sc-памяти* (sc-узел или sc-коннектор) могут использоваться для хранения спецификации *элемента sc-памяти, соответствующего sc-коннектору*. Также *sc-адрес первой sc-дуги, выходящей из данного sc-элемента** и *sc-адрес первой sc-дуги, входящей в данный sc-элемент** в общем случае могут отсутствовать (быть нулевыми, "пустыми"), но размер *элемента sc-памяти* в байтах останется тем же.

Спецификация элемента sc-памяти, соответствующего sc-узлу
⊃=
{

элемент sc-памяти, соответствующий sc-узлу
⇒ *понятие, специфицирующее заданную сущность**:
{• *класс элемента sc-памяти^*
• *класс уровня доступа к элементу sc-памяти^*
• *sc-адрес элемента sc-памяти**
• *sc-адрес первого sc-коннектора, выходящего из данного sc-элемента**
• *sc-адрес первого sc-коннектора, входящего в данный sc-элемент**
}

}

Спецификация элемента sc-памяти, соответствующего sc-коннектору
⊃=
{

элемент sc-памяти, соответствующий sc-коннектору
⇒ *понятие, специфицирующее заданную сущность**:
{• *класс элемента sc-памяти^*
• *класс уровня доступа к элементу sc-памяти^*
• *sc-адрес элемента sc-памяти**
• *sc-адрес элемента sc-памяти, соответствующего начальному sc-элементу sc-коннектора**
• *sc-адрес элемента sc-памяти, соответствующего конечному sc-элементу sc-коннектора**
• *sc-адрес элемента sc-памяти, соответствующего начальному выходящему sc-коннектору из заданного sc-элемента**

- *sc-адрес элемента sc-памяти, соответствующего начальному входящему sc-коннектору в заданный sc-элемент**
- *sc-адрес элемента sc-памяти, соответствующего следующему выходящему sc-коннектору из заданного sc-элемента**
- *sc-адрес элемента sc-памяти, соответствующего следующему входящему sc-коннектору в заданный sc-элемент**
- *sc-адрес элемента sc-памяти, соответствующего предыдущему выходящему sc-коннектору из заданного sc-элемента**
- *sc-адрес элемента sc-памяти, соответствующего предыдущему входящему sc-коннектору в заданный sc-элемент**

}
}

В текущей *Реализации sc-памяти в ostis-платформе* ***классы уровня доступа^*** используются для того, чтобы обеспечить возможность ограничения доступа некоторых *процессов в sc-памяти* к некоторым *элементам sc-памяти*. Каждый *элемент sc-памяти* принадлежит одному из двух классов: классу *элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право чтения* и классу *элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право записи*, каждый из которых выражается *целым числом* от 0 до 255.

Таким образом нулевое значение числовых выражений класса *элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право чтения* и класса *элементов sc-памяти, соответствующих sc-элементам, на которых разрешено право записи* означает, что любой процесс может получить неограниченный доступ к данному *элементу sc-памяти*.

В качестве примера на рисунке SCg-текст. Пример трансляции sc-текста в sc-память ostis-платформы представлены пятиэлементная sc-конструкция (слева) и конструкция в sc-памяти, представленная на SCin-коде (справа).

Пункт 6.3.3.4. Достоинства и недостатки текущего варианта Реализации sc-памяти в ostis-платформе и Метаязыка описания представления sc-конструкций в sc-памяти ostis-платформы

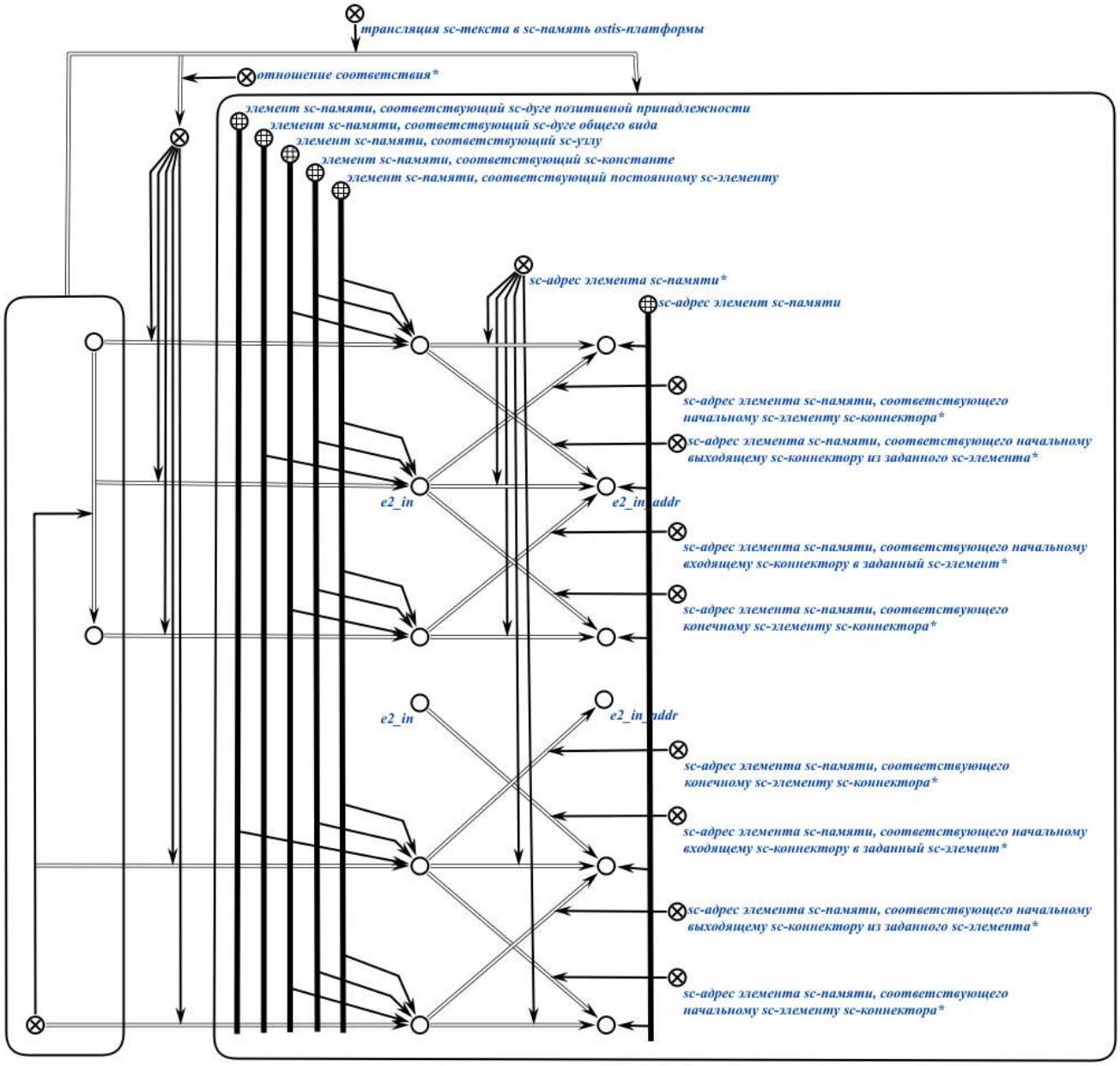
Описанная модель представления в текущей *Реализации sc-памяти в ostis-платформы синтаксических* и *семантических классов sc-элементов* в виде *синтаксических* и *семантических классов элементов sc-памяти^*, которые соответствуют первым, обладает рядом преимуществ:

- *синтаксические* и семантические классы элементов sc-памяти^ могут комбинироваться между собой для получения более частных классов. С точки зрения программной реализации такая комбинация может быть представлена операцией *битовое сложение* классов элементов sc-памяти^* (здесь, в спецификации на *SC-коде* это можно сделать с помощью пересечения соответствующих классов). Так, например, *битовое сложение** классов *элементов sc-памяти, соответствующих sc-узлу* и *sc-константе* в результате образуют новый *класс элементов sc-памяти^* — *элемент sc-памяти, соответствующий константному sc-узлу*.
- Числовые выражения некоторых классов могут совпадать. Это сделано для уменьшения размера *элемента sc-памяти* за счет уменьшения максимального размера числового выражения класса этих *элементов sc-памяти*. Конфликт в данном случае не возникает, поскольку такие классы не могут комбинироваться, например *элемент sc-памяти, соответствующий sc-узлу ролевого отношения* и *элемент sc-памяти, соответствующий sc-дуге нечеткой принадлежности*.
- Важно отметить, что каждому из выделенных *классов элементов sc-памяти* (кроме классов, получаемых путем комбинации других классов) однозначно соответствует порядковый номер бита в линейной памяти, что можно заметить, глядя на соответствующие числовые выражения этих классов. Это означает, что классы элементов не включаются друг в друга (хоть в спецификации это и не так), например, указание принадлежности к классу *элементов sc-памяти, соответствующих sc-дуге позитивной принадлежности* не означает автоматическое указание принадлежности *элементов sc-памяти, соответствующих sc-дуге принадлежности*. На уровне реализации это позволяет сделать операции комбинирования и сравнения меток более эффективными.

Реализация sc-памяти в ostis-платформе учитывает технические аспекты реализации современных *операционных систем*, что дает следующие достоинства:

- Используемый подход адресации *sc-элементов* позволяет загружать сегменты с диска в память, а также выгружать их обратно на диск в любой момент и при этом данная операция не требует дополнительных преобразований. Все содержимое из *оперативной памяти* без изменений попадает на диск. Это дает возможность выгружать неиспользуемые сегменты на диск, что позволяет *sc-памяти* абстрагироваться от имеющихся ресурсов *оперативной памяти* и работать на любых ее объемах;

SCg-текст. Пример трансляции sc-текста в sc-память ostis-платформы
=



- Максимальное количество хранимых *sc-элементов* можно увеличивать путем расширения *sc-адреса элемента sc-памяти*.

С точки зрения программной *Реализации sc-памяти в ostis-платформе*, структура данных для хранения *sc-узла* и *sc-коннектора* остается остается та же, но в ней меняется список полей (компонентов). Кроме того, как можно заметить каждый *элемент sc-памяти* (в том числе, *элемент sc-памяти, соответствующий sc-дуге*) не хранит список *sc-адресов* связанных с ним *элементов sc-памяти*, а хранит *sc-адреса* одного *выходящего* и одного *входящего элементов sc-памяти, соответствующих sc-коннекторам*, каждый из которых в свою очередь хранит *sc-адреса следующего* и *предыдущего элементов, соответствующих sc-коннекторам*, в списке выходящих и входящих *элементов sc-памяти.* Все перечисленное позволяет:

- сделать размер такой структуры фиксированным (в настоящее время 36 байт) и не зависящим от *синтаксического класса* хранимого *элемента в sc-памяти*^;
- с минимальными временными затратами добавлять и удалять инцидентные элементы в и из программной структуры *элемента sc-памяти* соответственно;
- обеспечить возможность работы с sc-элементами без учета их синтаксического класса в случаях, когда это необходимо (например, при реализации поисковых запросов вида “Какие sc-элементы являются элементами данного множества”, “Какие sc-элементы непосредственно связаны с данным sc-элементом” и так далее);

- обеспечить возможность доступа к *элементу sc-памяти* за константное время;
- обеспечить возможность помещения *элемента sc-памяти* в процессорный кэш, что в свою очередь, позволяет ускорить обработку *sc-конструкций*.

§ 6.3.4. Программный интерфейс Реализации sc-памяти в ostis-платформе

⇒ *подраздел**:
- *Пункт 6.3.4.1. Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе*
- *Пункт 6.3.4.2. Программный интерфейс информационно-поисковых методов Реализации sc-памяти в ostis-платформе*
- *Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу*
- *Пункт 6.3.4.4. Общее описание процесса и Пример трансляции sc-текста в sc-память ostis-платформы*
- *Пункт 6.3.4.5. Достоинства и недостатки Программного интерфейса Реализации sc-памяти в ostis-платформе*

⇒ *ключевой знак**:
- *Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы*
- *Программный интерфейс Реализации sc-памяти в ostis-платформе*
- *Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе*
- *Программный интерфейс информационно-поисковых методов Реализации sc-памяти в ostis-платформе*

⇒ *ключевое знание**:
- *Спецификация Программного интерфейса Реализации sc-памяти в ostis-платформе*
- *Принципы, лежащие в основе Программного интерфейса Реализации sc-памяти в ostis-платформе*
- *Ответ на Вопрос. Как создавать, находить, изменять и удалять конструкции в текущей Реализации sc-памяти в ostis-платформе*
- *Достоинства и недостатки Программного интерфейса Реализации sc-памяти в ostis-платформе*

SCin-кода достаточно, чтобы представлять *sc-тексты* внутри *sc-памяти ostis-платформы*. Чтобы совершить трансляцию некоторого *sc-текста* в *sc-память ostis-платформы* необходимо использовать методы текущей *Реализации sc-памяти в ostis-платформе*. Описанные далее методы sc-памяти являются формальной спецификацией текущего ***Программного интерфейса Реализации sc-памяти в ostis-платформе***, с помощью которых можно выполнять действия над *sc-памятью*.

В текущей *Реализации sc-памяти в ostis-платформе* все программные методы реализованы на *языках представления методов C* и *C++*. Текущий *Программный интерфейс Реализации sc-памяти в ostis-платформе* содержит необходимый функционал не только для выполнения действий над *элементами sc-памяти*, но и — над *элементами файловой памяти* (см. *§ 6.3.5. Реализация файловой памяти в Программной платформе ostis-систем*). Данный *программный интерфейс* является одним из языков текущего *Программного варианта реализации ostis-платформы* для выполнения действий над sc-памятью и может быть использован для решения *задач* любой информационной сложности. Так, например, данный программный интерфейс используется в текущих *Реализации Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к этой sc-памяти*, *Реализации менеджера многократных используемых компонентов ostis-систем* и *Реализации интерпретатора логических моделей решения задач*, а также при реализации любых *платформенно-зависимых ostis-систем* любого назначения.

Программный интерфейс Реализации sc-памяти в ostis-платформе
⇐ *программный интерфейс**:
Реализация sc-памяти в ostis-платформе
∈ *программный интерфейс*
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
∈ *атомарный многократно используемый компонент ostis-систем*
∈ *зависимый многократно используемый компонент ostis-систем*
⇒ *зависимости компонента**:
{• *Библиотека методов и структур данных GLib*
• *Библиотека методов и структур данных C++ Standart Library*

}
⇒ *используемый язык представления методов**:
- *C*
- *C++*

⇒ *внутренний язык**:
- *SCin-код*

⊃ *Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе*
:= [информационно-формирующие методы Реализации sc-памяти в ostis-платформе]
:= [подсистема, являющаяся частью Реализации sc-памяти в ostis-платформе, которая позволяет создавать, изменять и удалять конструкции в sc-памяти]
⇐ *программный интерфейс**:
Реализация информационно-порождающей подсистемы Реализации sc-памяти в ostis-платформе
⊂ *Реализация sc-памяти в ostis-платформе*

⊃ *Программный интерфейс информационно-поисковых методов Реализации sc-памяти в ostis-платформе*
:= [информационно-поисковые методы Реализации sc-памяти в ostis-платформе]
:= [подсистема, являющаяся частью Реализации sc-памяти в ostis-платформе, которая позволяет находить конструкции в sc-памяти]
⇐ *программный интерфейс**:
Реализация информационно-поисковой подсистемы Реализации sc-памяти в ostis-платформе
⊂ *Реализация sc-памяти в ostis-платформе*

Логически текущий *Программный интерфейс Реализации sc-памяти в ostis-платформе* разделяется на два программных интерфейса: ***Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе*** и ***Программный интерфейс информационно-поисковых методов Реализации sc-памяти в ostis-платформе***. В первую очередь, такое разделение связано с тем, что реализация методов информационного поиска в текущей *Реализации sc-памяти в ostis-платформе* достаточна сложна и требует намного большего уточнения при описании этой реализации. Также это разделение *программного интерфейса* позволяет выделять и структурировать спецификацию методов *Программного интерфейса Реализации sc-памяти в ostis-платформе* таким образом, чтобы она оставалась однообразной, компактной и простой для внешнего пользователя. Как такового физического разделения в *Программном интерфейсе Реализации sc-памяти в ostis-платформе* не существует, все методы *Программного интерфейса Реализации sc-памяти в ostis-платформе* можно использовать одним и тем же программным образом и являются компонентами ***Библиотеки многократно используемых компонентов Программного варианта реализации ostis-платформы***, то есть могут быть использованы при реализации других компонентов специального назначения.

Пункт 6.3.4.1. Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе

⇒ *ключевой знак**:
- *Метод создания элемента в sc-памяти, соответствующего некоторому sc-узлу заданного класса*
- *Метод создания элемента в sc-памяти, соответствующего некоторому sc-коннектору заданного класса*
- *Метод удаления элемента из sc-памяти*
- *Метод уточнения класса элемента sc-памяти, соответствующего некоторому sc-элементу*
- *Метод получения класса элемента sc-памяти, соответствующего некоторому sc-элементу*

Базовыми методами в Программном интерфейсе Реализации sc-памяти в ostis-платформе являются методы, которые позволяют создавать *конструкции в sc-памяти*, изменять их, а также удалять эти конструкции из нее. Физически разделяются процессы создания *элемента sc-памяти, соответствующего sc-узлу*, и *элемента sc-памяти, соответствующего sc-коннектору*, поскольку соответствующие им *sc-элементы* являются синтаксически разными, а сами элементы имеют разные программные модели внутри *Реализации sc-памяти в ostis-платформе*. Однако процесс удаления этих элементов их sc-памяти никоим образом не отличается. Удаление элемента из sc-памяти предполагает физическое извлечение элемента из линейно-адресуемой *sc-памяти* и освобождение занимаемого места в *оперативной памяти* устройства.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-формирующих методов Реализации sc-памяти в ostis-платформе
∋ *Метод создания элемента в sc-памяти, соответствующего некоторому sc-узлу заданного класса*
∋ *Метод создания элемента в sc-памяти, соответствующего некоторому sc-коннектору заданного класса*
∋ *Метод удаления элемента из sc-памяти*
∋ *Метод уточнения класса элемента sc-памяти, соответствующего некоторому sc-элементу*
∋ *Метод получения класса элемента sc-памяти, соответствующего некоторому sc-элементу*
}

Метод создания элемента в sc-памяти, соответствующего некоторому sc-узлу заданного класса
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScAddr ScMemoryContext::CreateNode(ScType const & type)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *класс элемента sc-памяти^*
⟩
⇒ *класс результата метода**:
• *sc-адрес элемента sc-памяти*
⇒ *класс исключительных ситуаций**:
• *некорректный синтаксический класс для создаваемого элемента в sc-памяти*
• *некорректный семантический класс для создаваемого элемента в sc-памяти*

Метод создания элемента в sc-памяти, соответствующего некоторому sc-коннектору заданного класса
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScAddr ScMemoryContext::CreateConnector(ScType const & type, ScAddr const & begAddr, ScAddr const & endAddr)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *класс элемента sc-памяти^*
• *sc-адрес элемента sc-памяти*
• *sc-адрес элемента sc-памяти*
⟩
⇒ *класс результата метода**:
• *sc-адрес элемента sc-памяти*
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*
• *некорректный синтаксический класс для создаваемого элемента в sc-памяти*
• *некорректный семантический класс для создаваемого элемента в sc-памяти*

При помощи ***Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-узлу*** и ***Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-коннектору*** можно создавать все программные элементы *Алфавита SCin-кода^*, соответствующие sc-элементам *Алфавита SC-кода^* в *sc-памяти ostis-платформы* (см. *Пункт 2.2.1.1. Семантическая классификация sc-элементов по базовым признакам*).

Для создания *элемента в sc-памяти, соответствующего sc-узлу*, необходимо указать соответствующий *семантический класс элемента в sc-памяти^*, являющийся подклассом *класса элемента sc-памяти, соответствующего sc-узлу^* (см. *Пункт 6.3.3.3. Денотационная семантика SCin-кода*). А для создания *элемента в sc-памяти, соответствующего sc-коннектору*, кроме того, что необходимо указать соответствующий *семантический класс элемента в sc-памяти^*, являющийся подклассом *класса элемента sc-памяти, соответствующего sc-коннектору^* (см. *Пункт 6.3.3.3. Денотационная семантика SCin-кода*), также необходимо указать *sc-адреса* инцидентных ему *элементов sc-памяти*: *sc-адрес элемента sc-памяти*, из которого создаваемый коннектор должен выходить, и *sc-адрес элемента sc-памяти*, в который данный коннектор должен входить. Такими *элементами sc-памяти* могут выступать любые *элементы sc-памяти*, которые уже находятся в ней. При этом при помощи *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-узлу* и *Метода создания*

элемента в sc-памяти с заданным классом, соответствующего некоторому sc-коннектору нельзя создавать *элементы sc-памяти, соответствующие sc-коннекторам*, и *элементы sc-памяти, соответствующие sc-узлам*, соответственно (см. *§ 2.2.2. Синтаксис SC-кода*). Также не допускается применять *Метод создания элемента в sc-памяти, соответствующего некоторому sc-коннектору заданного класса* для *sc-адресов*, которые не являются *sc-адресами элементов в sc-памяти*. В случае некорректного использования одного из этих методов внешний пользователь метода получит *исключительную ситуацию* с описанием ошибки. При помощи *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-узлу* можно создавать любые *элементы sc-памяти, соответствующие sc-узлам* или *файлам ostis-системы*, а при помощи *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-коннектору* — любые *элементы sc-памяти, соответствующие sc-коннекторам (sc-ребрам или sc-дугам)*.

При помощи ***Метода создания элемента в sc-памяти, соответствующего некоторому sc-узлу заданного класса*** и ***Метода создания элемента в sc-памяти, соответствующего некоторому sc-коннектору заданного класса*** можно транслировать в sc-память sc-конструкции любой конфигурации. *Общее описание процесса и пример трансляции sc-текста в sc-память ostis-платформы* рассмотрены в *Пункт 6.3.4.4. Общее описание процесса и Пример трансляции sc-текста в sc-память ostis-платформы*.

При реализации прикладных *ostis-систем* очень часто необходимо создавать сложные *конструкции в sc-памяти*. Чтобы упростить разработку и сделать текст разрабатываемого метода, использующего *Программный интерфейс Реализации sc-памяти в ostis-платформе*, более простым и компактным можно использовать *Метод создания конструкции в sc-памяти, изоморфной данному графу-образцу*.

Метод удаления элемента из sc-памяти

∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[bool ScMemoryContext::EraseElement(ScAddr const & addr)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *sc-адрес элемента sc-памяти*
⟩
⇒ *класс результата метода**:
• *логическое значение*
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*

Любой *элемент sc-памяти* может быть удален из нее. Для того, чтобы удалить заданный *элемент из sc-памяти* необходимо применить ***Метод удаления элемента из sc-памяти*** с указанным в качестве аргумента *sc-адресом* заданного *элемента sc-памяти*. Если заданный sc-адрес в действительности является *sc-адресом* некоторого *элемента sc-памяти*, то этот элемент будет удален из *sc-памяти*, а метод закончит работу с результатом логического значения *Истина*. Если указанный sc-адрес является *sc-адресом* некоторого *элемента sc-памяти, соответствующего некоторому sc-элементу*, из которого выходят или в который входят *элементы sc-памяти, соответствующие sc-коннекторам*, то из *sc-памяти* также будут эти *элементы sc-памяти, соответствующие sc-коннекторам*.

Стоит отметить, что *действие удаления элемента из sc-памяти* может привести к внештатным ситуациям в самой *ostis-платформе*. Поэтому кроме *Реализации sc-памяти в ostis-платформе* и *Программного интерфейса Реализации sc-памяти в ostis-платформе* должны быть реализованы средства верификации действий над *sc-памятью* (см. *Главу 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*).

Метод уточнения класса элемента sc-памяти, соответствующего некоторому sc-элементу

∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[bool ScMemoryContext::SetElementSubtype(ScAddr const & addr, ScType const & type)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *sc-адрес элемента sc-памяти*
• *класс элемента sc-памяти^*
⟩
⇒ *класс результата метода**:
• *логическое значение*
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*
• *некорректный синтаксический класс для уточнения класса элемента sc-памяти*
• *некорректный семантический класс для уточнения класса элемента sc-памяти*

На практике частой ситуацией бывает, что пользователю, разработчику или самой системе неизвестны все знания о создаваемом *элементе в sc-памяти* либо об элементах его *семантической окрестности*. Так, например, во время трансляции в *sc-память ostis-платформы* исходных файлов, реализованных на *внешних языках представления знаний*: *SCs-коде* или *SCg-коде* — зачастую происходит уточнение классов уже созданных элементов sc-памяти. Такие действия над *элементами в sc-памяти ostis-платформы* можно осуществить при помощи ***Метода уточнения класса элемента sc-памяти, соответствующего некоторому sc-элементу***. Для этого необходимо задать в качестве аргументов *sc-адрес элемента sc-памяти*, для которого необходимо уточнить класс, и сам *класс элемента sc-памяти*^. В случае успешного выполнения метода в результате пользователем этого метода будет получено логическое значение *Истина*. По правилам *Синтаксиса SC-кода* запрещается указывать несколько *синтаксических классов* для одного и того же *sc-элемента*, также как и запрещается уточнять класс *sc-элемента*, которые не могут быть указаны для *sc-элементов* с заданным *синтаксическим классом*. При попытке совершить такие действия над *элементом sc-памяти* результатом работы *Метода уточнения класса элемента sc-памяти, соответствующего некоторому sc-элементу* будет логическое значение *Ложь*.

Во всех методах, где используются классы обрабатываемых *элементов sc-памяти*, в качестве операций над классами используются операции *Булевой алгебры*: *битовое сложение** (|), *битовое умножение** (&) и *битовое отрицание** (!). Так, при помощи операции *битового умножения** можно определить: является ли заданный класс подклассом *класса* заданного *элемента sc-памяти*^, при помощи операции *битового умножения** — уточнять *класс* заданного *элемента sc-памяти*^, а при помощи операции *битового отрицания** в комбинациях с первыми двумя операциями — проверять корректность задаваемых *классов элементов sc-памяти*^. Так, например, *битовое сложение** классов элементов sc-памяти, соответствующих *sc-коннектору* и sc-переменной в результате образуют новый *класс элементов sc-памяти*^ — *элемент sc-памяти, соответствующий константному sc-узлу*. Таким образом, в результате можно получать любые допустимые комбинации *классов элементов sc-памяти*^.

Метод получения класса элемента sc-памяти, соответствующего некоторому sc-элементу
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScType ScMemoryContext::GetElementType(ScAddr const & addr) const]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *sc-адрес*
⟩
⇒ *класс результата метода**:
• *класс элемента sc-памяти*^
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*

Чтобы получить *класс* заданного *элемента sc-памяти*^ достаточно применить ***Метод получения класса элемента sc-памяти, соответствующего некоторому sc-элементу***, задав в качестве аргумента sc-адрес заданного элемента sc-памяти. В последних двух методах не допускается использовать *sc-адреса*, которые не являются *sc-адресами элементов в sc-памяти*.

Пункт 6.3.4.2. Программный интерфейс информационно-поисковых методов Реализации sc-памяти в ostis-платформе

⇒ *ключевой знак**:
• *Метод создания итератора поиска трехэлементных конструкций в sc-памяти*
• *Метод создания итератора поиска пятиэлементных конструкций в sc-памяти*
• *Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу*
• *Метод создания конструкции в sc-памяти, изоморфной данному графу-образцу*
• *Метод создания программного объекта графа-образца*
⇒ *ключевое понятие**:
• *информационный поиск*
• *изоморфный поиск*
• *граф-образец*

В задачах, решаемых в прикладных ostis-системах, реализуемых на базе текущего *Программного варианта реализации ostis-платформы*, необходимо использовать механизмы поиска уже существующих элементов или *конструкций в sc-памяти*. Такие механизмы являются частью ***Реализации информационно-поисковой подсистемы Реализации sc-памяти в ostis-платформе***, на базе которой можно реализовывать *информационные-поисковые подсистемы* для *платформенно-зависимых* и *платформенно-независимых ostis-систем*. Несмотря на сложность

информационного поиска, текущий *Программный вариант реализации ostis-платформы* позволяет эффективно использовать реализуемые методы *информационного поиска* в задачах, решаемых прикладными ostis-системами. Данную подсистему нельзя реализовать независимо от реализации *ostis-платформы*, то есть ее нельзя сделать *платформенно-независимой*, поэтому необходимо разделять *Реализацию информационно-поисковой подсистемы Реализации sc-памяти в ostis-платформе* и *Реализацию информационно-поисковой подсистемы Метасистемы OSTIS*, которая реализуется на *Языке SCP* и независимо от текущего *Программного варианта реализации ostis-платформы* (см. *Главу 7.2. Метасистема OSTIS*). Сам же *scp-интерпретатор* должен использовать информационно-поисковые методы *Реализации sc-памяти в ostis-платформе*, а *Язык SCP* — предоставлять возможность навигироваться по *базе знаний* любой *ostis-системы* (см. *§ 3.3.5. Базовый язык программирования ostis-систем*).

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
∋ *Метод создания итератора поиска трехэлементных конструкций в sc-памяти*
∋ *Метод создания итератора поиска пятиэлементных конструкций в sc-памяти*
∋ *Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу*
∋ *Метод создания конструкции в sc-памяти, изоморфной данному графу-образцу*
∋ *Метод создания программного объекта графа-образца*
}

Метод создания итератора поиска трехэлементных конструкций в sc-памяти
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[template <typename ParamType1, typename ParamType2, typename ParamType3> std::shared_ptr<TIterator5< ParamType1, ParamType2, ParamType3» Iterator3(ParamType1 const & param1, ParamType2 const & param2, ParamType3 const & param3)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
⟩
⇒ *класс результата метода**:
• *итератор поиска трехэлементных конструкций в sc-памяти*
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*

Метод создания итератора поиска пятиэлементных конструкций в sc-памяти
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[template <typename ParamType1, typename ParamType2, typename ParamType3, typename ParamType4, typename ParamType5> std::shared_ptr<TIterator5<ParamType1, ParamType2, ParamType3, ParamType4, ParamType5» Iterator5(ParamType1 const & param1, ParamType2 const & param2, ParamType3 const & param3, ParamType4 const & param4, ParamType5 const & param5)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
• *параметр Метода создания итератора поиска конструкций в sc-памяти*
⟩
⇒ *класс результата метода**:
• *итератор поиска пятиэлементных конструкций в sc-памяти*
⇒ *класс исключительных ситуаций**:
• *элемент с заданным sc-адресом не существует в sc-памяти*

По правилам *Синтаксиса SC-кода sc-конструкции*, то есть конструкции, состоящие из *sc-элементов*, могут состоять из трех *sc-элементов*, пяти *sc-элементов*, семи sc-элементов и так далее (см. *§ 2.2.2. Синтаксис SC-кода*). В *sc-памяти ostis-платформы* эквивалентом *sc-конструкции* является конструкция, состоящая из *элементов sc-памяти* (***конструкция в sc-памяти***). ***Метод создания итератора поиска трехэлементных конструкций в sc-памяти*** и ***Метод создания итератора поиска пятиэлементных конструкций в sc-памяти*** позволяют создавать итераторы поиска трех- и пяти- элементных *конструкций в sc-памяти ostis-платформы* соответственно. С помощью параметров данных методов можно создавать итераторы любой необходимой конфигурации для поиска *трех-* и пяти- элементных sc-конструкций. Так, например, если необходимо найти все *элементы sc-памяти, соответствующие sc-элементам*, которые принадлежат некоторому *sc-множеству*, для которого соответствует заданный *элемент sc-памяти*, то необходимо при помощи *Метода создания итератора поиска трехэлементных конструкций элементов в sc-памяти* создать итератор поиска, задав в качестве первого аргумента *элемент sc-памяти, соответствующий* заданному *sc-множеству*, а в качестве второго и третьего аргументов — *класс элементов sc-памяти, соответствующих базовой sc-дуге*^ и *класс элементов sc-памяти, соответствующих sc-элементам неуточненного класса*^, соответственно. Для поиска в *sc-памяти* более сложных конструкций, состоящих из семи и более элементов, необходимо комбинировать итераторы поиска трех- и пяти- элементных *конструкций в sc-памяти*, или использовать ***Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу***.

программный интерфейс этих *методов* ограничивается особенностями *языка представления методов C++*. Например, при помощи этих *методов* нельзя создать *итераторы*, задав в качестве аргументов только *классы элементов sc-памяти*^, по которым необходимо найти все *соответствующие конструкции в sc-памяти*, в которых *классы их элементов sc-памяти*^ являются классами, заданными в качестве аргументов для указанных методов, как и впрочем нельзя указать что-то иное в качестве аргументов, кроме *sc-адреса* или *класса элемента sc-памяти*^. Попытка произвести что-то из перечисленного приведет к ошибке "склейки" интерфейса одного из указанных *методов*, указанного в заголовочном файле *C++*, с реализацией с заданными параметрами, указанной в исходном файле *C++*, поскольку компилятор *C++* не сможет найти реализацию для указанного *программного интерфейса*. Таким образом, согласно последней проблеме, параметрами *Метода создания итератора поиска трехэлементных конструкций в sc-памяти* и *Метода создания итератора поиска пятиэлементных конструкций в sc-памяти* могут быть *sc-адрес* и/или *класс элемента sc-памяти*^.

Создаваемые при помощи данных *методов итераторы* также *имеют программный интерфейс*. Результатами обоих *методов* являются разные *итераторы*, но имеющие общий *программный интерфейс*. Такие *итераторы* позволяют работать с *конструкциями в sc-памяти* в тот же момент, когда они были найдены. При помощи *Метода перехода к следующей "подходящей" для заданного итератора конструкции в sc-памяти итератор* обновляет свое внутреннее состояние каждый раз, когда была найдена новая *конструкция в sc-памяти*. Под следующей "подходящей" для заданного итератора *конструкцией в sc-памяти* подразумевается *конструкция в sc-памяти*, элементы которой удовлетворяют конфигурации созданного *итератора поиска конструкций в sc-памяти* и которая не была найдена раннее. Результатом последнего *метода* является логическое значение *Истина*, если следующая "подходящая" для заданного *итератора* конструкция существует в *sc-памяти*. Если "подходящих" для этого *итератора* конструкций нет в *sc-памяти*, то результатом *Метода перехода к следующей подходящей для заданного итератора конструкции в sc-памяти* является логическое значение *Ложь*. Чтобы получить *sc-адрес* некоторого из элементов найденной *конструкции в sc-памяти* необходимо использовать *Метода доступа к sc-адресу элемента заданной конструкции в sc-памяти по номеру позиции этого элемента в заданной конструкции sc-памяти*, задав качестве аргумента целочисленное значение в виде номера позиции искомого элемента в этой конструкции. При этом нумерация позиций элементов в *конструкции sc-памяти* в текущей *Реализации sc-памяти в ostis-платформе* происходит начиная с нуля, а не с единицы, а порядок нумерации задается тем, какой порядок аргументов для создания *итератора* был использован. При попытке указать для данного метода номер позиции, по которому не существует позиции в данной конструкции, результатом этого метода будет *исключительная ситуация* о недопустимой позиции элемента в заданной *конструкции sc-памяти*. Таким образом, диапазон номеров позиции для *трехэлементных конструкций* ограничен от нуля до двух, а для пятиэлементных — от нуля до четырех.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{
⊃ *Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе*

итератор поиска трех- и пяти- элементных конструкций в sc-памяти
⊂ *программный объект*
:= [ScIterator]
∈ *C++*

Программный интерфейс итератора поиска трех- и пяти- элементных конструкций в sc-памяти
⊃=
{
⇐ *программный интерфейс**:
итератор поиска трех- и пяти- элементных конструкций в sc-памяти

Метод перехода к следующей "подходящей" для заданного итератора конструкции в sc-памяти
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[bool Next() const]
∈ *C++*
⇒ *класс результата метода**:
• *логическое значение*

Метод доступа к sc-адресу элемента заданной конструкции в sc-памяти по номеру позиции этого элемента в заданной конструкции
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScAddr Get(size_t idx) const]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *32-битовое целое число*
⟩
⇒ *класс результата метода**:
• *sc-адрес элемента sc-памяти*
⇒ *класс исключительных ситуаций**:
• *недопустимая позиция элемента в заданной конструкции sc-памяти*
}
}
}

Для *Метода создания итератора поиска трехэлементных конструкций в sc-памяти*, как и для *Метода создания итератора поиска пятиэлементных конструкций в sc-памяти* можно использовать различные комбинации параметров, кроме комбинаций, где все параметры являются классами *элементов sc-памяти*. Для простоты и компактности используемых терминов на уровне реализации методов создания итераторов поиска конструкций в sc-памяти вводятся дополнительные обозначения: символом "f" (от английского слова fixed) обозначается факт, что параметром заданного метода создания итератора поиска *конструкций в sc-памяти* указан *sc-адрес* некоторого *элемента sc-памяти*, а символом "a" (от английского слова assign) — *класс элемента sc-памяти*^. Для *Метода создания итератора поиска трехэлементных конструкций в sc-памяти* правильным обозначением искомых конструкций будет комбинация символов "f" и "a" длиной в три символа, а для *Метода создания итератора поиска пятиэлементных конструкций в sc-памяти* — комбинация символов "f" и "a" длиной в пять символов. В *Языке SCP* для указания признака заданности значения у переменной используются соответствующие ролевые отношения: для переменных класса "f" — *scp-операнд с заданным значением'*, для переменных класса "a" — *scp-операнд со свободным значением'* (см. *§ 3.3.5. Базовый язык программирования ostis-систем*).

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{

Метод создания итератора поиска трехэлементных конструкций в sc-памяти

⊃ *Метод создания итератора поиска fff-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    ⟩
⊃ *Метод создания итератора поиска faa-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    ⟩
⊃ *Метод создания итератора поиска aaf-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *sc-адрес элемента sc-памяти*
    ⟩
⊃ *Метод создания итератора поиска faf-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *класс элемента sc-памяти^*
    • *sc-адрес элемента sc-памяти*
    ⟩
⊃ *Метод создания итератора поиска afa-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *класс элемента sc-памяти^*
    • *sc-адрес элемента sc-памяти*
    • *класс элемента sc-памяти^*
    ⟩
}
}

Данные варианты реализации *Метода создания итератора поиска трехэлементных конструкций в sc-памяти* являются достаточными для решения любых поисково-навигационных задач. Возможны варианты реализации ***Метода создания итератора поиска ffa-конструкций в sc-памяти*** и ***Метода создания итератора поиска aff-конструкций в sc-памяти***, но на практике нет необходимости искать третий элемент sc-памяти по известным элементу, соответствующему sc-коннектору, и элементу, соответствующему sc-элементу, из которого данный sc-коннектор выходит или в который данный sc-коннектор входит. Такую задачу можно решить, используя ***Метод создания итератора поиска afa-конструкций в sc-памяти***. Впрочем, реализация *Метода создания итератора поиска пятиэлементных конструкций в sc-памяти* вовсе не обязательна, поскольку все задачи, решаемые при помощи данного метода, можно также решить при помощи *Метода создания итератора поиска трехэлементных конструкций в sc-памяти*, однако реализация *Метода создания итератора поиска пятиэлементных конструкций в sc-памяти* позволяет сделать текст создаваемого метода более компактным по сравнению с методом, в котором бы использовался *Метод создания итератора поиска трехэлементных конструкций в sc-памяти*.

В качестве всех трех аргументов для *Метода создания итератора поиска трехэлементных конструкций в sc-памяти* могут быть указаны:

- *sc-адреса элементов sc-памяти* (например, задача проверки инцидентности всех трех заданных элементов sc-памяти),
- *sc-адрес элемента sc-памяти*, *класс элементов sc-памяти, соответствующих sc-коннекторам^*, которые выходят из заданного в качестве первого аргумента *элемента sc-памяти*, и *sc-адрес элемента sc-памяти*, соответствующего некоторому *sc-элементу*, в который искомые *sc-коннекторы* входят (например, задача поиска всех *элементов sc-памяти, соответствующих sc-коннекторам между sc-элементами*, для которых соответствуют заданные *элементы sc-памяти*),
- *sc-адрес элемента sc-памяти*, *класс элементов sc-памяти, соответствующих sc-коннекторам^*, которые выходят из заданного в качестве первого аргумента *элемента sc-памяти*, и *класс элементов sc-памяти,*

*соответствующих некоторым sc-элементам^*, в которые искомые *sc-коннекторы* входят (например, задача поиска всех *элементов sc-памяти*, *соответствующих sc-коннекторам*, выходящим из *sc-элемента*, для которого соответствует заданный в качестве первого аргумента *элемент sc-памяти*),

- *класс sc-элементов sc-памяти^*, *класс элементов sc-памяти, соответствующих sc-коннекторам^*, которые выходят из заданных в качестве первого аргумента *элементов sc-памяти*, и *sc-адрес элемента sc-памяти, соответствующего некоторому sc-элементу*, в который искомые *sc-коннекторы* входят (например, задача поиска всех *элементов sc-памяти, соответствующих sc-коннекторам*, входящим в *sc-элемент*, для которого соответствует заданный в качестве третьего аргумента *элемент sc-памяти*),
- *класс элементов sc-памяти^*, *sc-адрес элемента sc-памяти, соответствующий sc-коннектору*, который выходит из заданного в качестве первого аргумента *элемента sc-памяти*, и *класс элементов sc-памяти, соответствующих некоторому sc-элементу^*, в который искомый *sc-коннектор* входит (например, задача поиска *элементов sc-памяти, соответствующих sc-элементам*, один из которых является *sc-элементом*, из которого выходит *sc-коннектор*, для которого соответствует заданный *элемент sc-памяти*, а второй из которых является *sc-элементом*, в который входит этот *sc-коннектор*, для которого соответствует заданный *элемент sc-памяти*)
- и так далее.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{

Метод создания итератора поиска пятиэлементных конструкций элементов в sc-памяти
⊃ *Метод создания итератора поиска fffff-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    • *sc-адрес элемента sc-памяти*
    ⟩
⊃ *Метод создания итератора поиска faaaa-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    ⟩
⊃ *Метод создания итератора поиска aaaaf-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *sc-адрес элемента sc-памяти*
    ⟩
⊃ *Метод создания итератора поиска faaaf-конструкций в sc-памяти*
  ∈ *метод*
  ⇒ *классы входных аргументов метода**:
    ⟨• *sc-адрес элемента sc-памяти*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*
    • *класс элемента sc-памяти^*

- *sc-адрес элемента sc-памяти*

⟩

⊃ *Метод создания итератора поиска fafaf-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *sc-адрес элемента sc-памяти*

- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*

⟩

⊃ *Метод создания итератора поиска afaaa-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *класс элемента sc-памяти^*

- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*

⟩

⊃ *Метод создания итератора поиска aaafa-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *класс элемента sc-памяти^*

- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*

⟩

⊃ *Метод создания итератора поиска aafaa-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *класс элемента sc-памяти^*

- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*

⟩

⊃ *Метод создания итератора поиска afaaf-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *класс элемента sc-памяти^*

- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*

⟩

⊃ *Метод создания итератора поиска faafa-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *sc-адрес элемента sc-памяти*

- *класс элемента sc-памяти^*
- *класс элемента sc-памяти^*
- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*

⟩

⊃ *Метод создания итератора поиска afafa-конструкций в sc-памяти*

∈ *метод*

⇒ *классы входных аргументов метода**:

⟨• *класс элемента sc-памяти^*

- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*

- *sc-адрес элемента sc-памяти*
- *класс элемента sc-памяти^*

⟩
}
}

В качестве всех пяти аргументов для ***Метода создания итератора поиска пятиэлементных конструкций в sc-памяти*** могут быть указаны и другие комбинации, не указанные в представленной классификации. Однако в этом нет необходимости, поскольку все задачи, решаемые с помощью таких *итераторов*, можно решить уже существующими *итераторами поиска пятиэлементных конструкций в sc-памяти*.

Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу

⇒ *ключевой знак**:
- *Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу*
- *Метод создания конструкции в sc-памяти, изоморфной данному графу-образцу*
- *Метод создания программного объекта графа-образца*
- *Программный интерфейс программного объекта графа-образца*
- *Программный интерфейс программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу*

⇒ *ключевое понятие**:
- *программный объект графа-образца*
- *программный объект трехэлементной конструкции заданного графа-образца*
- *программный объект конструкции в sc-памяти, изоморфной заданному графу-образцу*

Метод создания итератора поиска трехэлементных конструкций в sc-памяти и *Метод создания итератора поиска пятиэлементных конструкций в sc-памяти*, а также *программный интерфейс итератора поиска трех- и пяти- элементных конструкций в sc-памяти* являются достаточно мощными средствами для решения любых задач *информационного поиска* в прикладных *ostis-системах*. Однако есть *модели решения задач*, в которых необходимо использовать другие методы поиска. Так, например, в задачах *логического вывода* считается удобным решать *задачи*, когда *поиск конструкций* любой необходимой конфигурации *в sc-памяти* сводится к ***изоморфному поиску*** этих конструкций по заданному ***графу-образцу***. Такими *графами-образцами* могут выступать любые *атомарные логические формулы*, входящие в состав в любой другой *неатомарной формулы* (см. *Главу 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*).

изоморфный поиск является одним из способов решения *задачи* поиска подграфа в *графе* (см. *Fortin S.tGraphIP-1996art*). Эта задача заключается в том, чтобы найти все вхождения заданного *графа-образца* в исходном графе. Процесс *изоморфного поиска* может быть реализован с использованием различных алгоритмов. Один из них — *Алгоритм Ульмана*, который основан на использовании матрицы смежности, чтобы определить соответствие между вершинами графов. Другой алгоритм — *Алгоритм VF2*, который использует функцию сравнения для проверки соответствия соответствующих узлов и ребер в двух графах (см. *Foggia P..aPerfoCoFAfG-2001art*). В современной информатике также известны алгоритмы, позволяющие решать *задачу изоморфного поиска* за субэкспоненциальное время (см. *McKay B.D.NautyUGV24-2007art*).

Основополагающим принципом разрабатываемой *Технологии OSTIS* является принцип заимствования наилучших существующих технологий для разработки *ostis-систем*. Однако в силу разных обстоятельств, например, связанных с особенностями *Реализации sc-памяти в ostis-платформе*, а также требований, предъявляемых к sc-агентам, занимающихся логическим выводом, необходимо разрабатывать и апробировать новые решения. В рамках текущей *Реализации sc-памяти в ostis-платформе* разработан принципиально другой алгоритм *изоморфного поиска*, позволяющего за оптимальное время находить графы, изоморфные данному *графу-образцу*.

В общем случае нет необходимости реализовывать *изоморфный поиск* в универсальном виде. Это объясняется тем, что:
- *изоморфный поиск* является NP-полной задачей, что означает, что затраты на ее решение растут экспоненциально с размером входных данных, и пока не существует эффективного алгоритма для решения задачи *изоморфного поиска*;
- в результате определения изоморфизма двух заданных графов может быть обнаружено несколько узлов, которые соответствуют друг другу, но на самом деле не являются изоморфными;
- из-за экспоненциального роста количества возможных вариантов изоморфизма при увеличении размера графа, даже небольшие погрешности при расчете изоморфизма могут привести к сильному искажению результатов;

- для больших графов временные затраты на перебор всех возможных вариантов изоморфизма могут быть очень высокими. Это может снижать эффективность поиска и ограничивать возможности применения *изоморфного поиска* в реальных задачах (см. *Cordella L.P..aSubGIAfLG-2004art*);
- сложность поиска увеличивается с увеличением количества циклов в исходном *графе*, поскольку приводит к большему числу переборов;
- существующие алгоритмы либо медленны, либо нерационально используют память, что приводит *изоморфный поиск* к медленной работе.

Некоторые алгоритмы изоморфного поиска вовсе имеют сложность *O(n!)* и не могут быть использованы для графов большого размера (см. *Fortin S.tGraphIP-1996art*). Несмотря на все проблемы, связанные с *изоморфным поиском*, для удобства решения логических задач в текущем *Программном варианте реализации ostis-платформы* реализован "наиболее подходящий" алгоритм *изоморфного поиска*. Текущий вариант *изоморфного поиска* реализован в *Методе поиска конструкций в sc-памяти, изоморфных данному графу-образцу*. Данный метод позволяет находить *конструкции в sc-памяти*, изоморфные не просто некоторому *графу-образцу*, который представлен в *sc-памяти*, а *программному объекту* этого *графа-образца*, то есть *графу-образцу*, который представлен в удобном и быстром для обработки программном формате.

Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу

∈ *метод*

⇒ *заголовок метода на языке представления методов**:
[ScTemplate::Result HelperSearchTemplate(ScTemplate const & templ, ScTemplateSearchResult & result)**]**
∈ *C++*

⇒ *классы входных аргументов метода**:
⟨• *программный объект графа-образца*
• *кортеж программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу*
⟩

⇒ *класс результата метода**:
• *код ошибки результата поиска конструкций в sc-памяти, изоморфной заданному графу-образцу*

⇒ *класс исключительных ситуаций**:
• *синтаксически некорректный программный объект графа-образца*
• *семантически некорректный программный объект графа-образца*

Метод создания конструкции в sc-памяти, изоморфной данному графу-образцу

∈ *метод*

⇒ *заголовок метода на языке представления методов**:
[ScTemplate::Result HelperGenTemplate(ScTemplate const & templ, ScTemplateGenResult & result)**]**
∈ *C++*

⇒ *классы входных аргументов метода**:
⟨• *программный объект графа-образца*
• *программный объект конструкции sc-памяти, изоморфной заданной графу-образцу*
⟩

⇒ *класс результата метода**:
• *код ошибки результата создания конструкции, изоморфной заданному графу-образцу*

⇒ *класс исключительных ситуаций**:
• *синтаксически некорректный программный объект графа-образца*
• *семантически некорректный программный объект графа-образца*

Метод создания программного объекта графа-образца

∈ *метод*

⇒ *заголовок метода на языке представления методов**:
[ScTemplate::Result HelperBuildTemplate(ScTemplate & templ, ScAddr const & templAddr)**]**
∈ *C++*

⇒ *классы входных аргументов метода**:
⟨• *программный объект графа-образца*
• *sc-адрес элемента sc-памяти*
⟩

⇒ *класс результата метода**:
• *код ошибки результата создания программного объекта по заданному элементу, соответствующему графу-образцу*

⇒ *класс исключительных ситуаций**:
• *синтаксически некорректный программный объект графа-образца*

- *семантически некорректный программный объект графа-образца*

программный объект граф-образца можно сформировать при помощи *Метода создания программного объекта графа-образца*, для этого необходимо в качестве аргументов задать *программный объект графа-образца* как выходный параметр и *sc-адрес элемента sc-памяти, соответствующего sc-структуре атомарной логической формулы* (sc-шаблону). Как и *итератор поиска трех- и пяти- элементных конструкций в sc-памяти*, *программный объект графа-образца* имеет специализированный *программный интерфейс*. Изначально *программный объект графа-образца* задается пустым. При помощи ***Метода добавления трехэлементной конструкции для заданного графа-образца*** и ***Метода добавления пятиэлементной конструкции для заданного графа-образца*** можно расширять заданный *программный объект графа-образца*. Расширение *программного объекта графа-образца* происходит с добавлением трехэлементных конструкций в ***кортеж программных объектов трехэлементных конструкций в заданном графе-образце***, при этом порядок программный объектов конструкций в этом множестве задается последовательностью выполнения *Метода добавления трехэлементной конструкции для заданного графа-образца* и *Метода добавления пятиэлементной конструкции для заданного графа-образца*.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{

программный объект графа-образца
:= [ScTemplate]
∈ *C++*
:= [программный объект атомарной логической формулы]
:= [программный объект sc-шаблона]
⊂ *программный объект*
⇒ *понятие, специфицирующее заданную сущность**:
{• *кортеж программных объектов трехэлементных конструкций в заданном графе-образце*
:= [std::vector<ScTemplateTriple *> m_templateTriples]
• *локальный идентификатор элемента sc-памяти и номера позиций этого элемента в заданном графе-образце**
:= [std::unordered_multimap<std::string, size_t> m_templateItemsNamesToReplacementItemsPositions]
• *упорядоченное по степени приоритета поиска множество множеств номеров программных объектов трехэлементных конструкций в заданном графе-образце*
:= [std::vector<std::unordered_set<size_t» m_priorityOrderedTemplateTriples]
• *локальный идентификатор элемента sc-памяти и sc-адрес этого элемента в заданном графе-образце**
:= [std::map<std::string, ScAddr> m_templateItemsNamesToReplacementItemsAddrs]
• *локальный идентификатор элемента sc-памяти и класс этого элемента в заданном графе-образце**
:= [std::map<std::string, ScType> m_templateItemsNamesToTypes]
}

программный объект трехэлементной конструкции заданного графа-образца
:= [ScTemplateTriple]
∈ *C++*
⊂ *программный объект*
⇒ *понятие, специфицирующее заданную сущность**:
{• *номер программного объекта трехэлементной конструкции в заданном графе-образце*
:= [size_t m_index]
• *кортеж трех элементов программного объекта трехэлементной конструкции*
:= [std::array<ScTemplateItem, 3> m_values]
}

элемент программного объекта трехэлементной конструкции
:= [ScTemplateItem]
∈ *C++*
⊂ *программный объект*

⇒ *понятие, специфицирующее заданную сущность**:
{• *sc-адрес элемента программного объекта трехэлементной конструкции*
⊂ *sc-адрес элемента sc-памяти*
:= [ScAddr m_addrValue]
• *класс элемента программного объекта трехэлементной конструкции*
⊂ *класс элемента sc-памяти^*
:= [ScType m_typeValue]
• *локальный идентификатор программного объекта трехэлементной конструкции*
:= [std::string m_name]
}

Программный интерфейс программного объекта графа-образца
⊃=
{
⇐ *программный интерфейс**:
программный объект графа-образца

Метод добавления трехэлементной конструкции для заданного графа-образца
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScTemplate & Triple(ScTemplateItemValue const & param1, ScTemplateItemValue const & param2, ScTemplateItemValue const & param3)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
⟩
⇒ *класс результата метода**:
• *программный объект графа-образца*
⇒ *класс исключительных ситуаций**:
• *некорректный параметр Метода добавления конструкции для заданного графа-образца*
⊃ *локальный идентификатор не привязан раннее к sc-адресу элемента программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*
⊃ *локальный идентификатор уже использован ранее для другого sc-адреса элемента программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*
⊃ *один и тот же локальный идентификатор одновременно указан для второго и первого (третьего) элементов заданного программного объекта трехэлементной конструкции*
⊃ *класс элемента создаваемого программного объекта трехэлементной конструкции в заданном программном объекте графа-образца является классом элементов в sc-памяти, соответствующих sc-константам*
⊃ *элемент с sc-адресом элемента создаваемого программного объекта трехэлементной конструкции в заданном программном объекте графа-образца не существует в sc-памяти*

Метод добавления пятиэлементной конструкции для заданного графа-образца
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScTemplate & Fiver(ScTemplateItemValue const & param1, ScTemplateItemValue const & param2, ScTemplateItemValue const & param3, ScTemplateItemValue const & param4, ScTemplateItemValue const & param5)]
∈ *C++*
⇒ *устаревший заголовок метода на языке представления методов**:
[ScTemplate & TripleWithRelation(ScTemplateItemValue const & param1, ScTemplateItemValue const & param2, ScTemplateItemValue const & param3, ScTemplateItemValue const & param4, ScTemplateItemValue const & param5)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
• *параметр Метода добавления конструкции для заданного графа-образца*
⟩

⇒ *класс результата метода**:
- *программный объект графа-образца*

⇒ *класс исключительных ситуаций**:
- *некорректный параметр Метода добавления конструкции для заданного графа-образца*
  - ⊃ *локальный идентификатор не привязан раннее к sc-адресу элементу программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*
  - ⊃ *локальный идентификатор уже использован ранее для другого sc-адреса элемента программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*
  - ⊃ *один и тот же локальный идентификатор одновременно указан для второго и первого (третьего) элементов заданного программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*
  - ⊃ *класс элемента создаваемого программного объекта трехэлементной конструкции является классом элементов в sc-памяти, соответствующих sc-константам*
  - ⊃ *элемент с sc-адресом элемента создаваемого программного объекта трехэлементной конструкции в заданном программном объекте графа-образца не существует в sc-памяти*

}
}
}

Чтобы сформировать необходимый *программный объект графа-образца* следует выполнить следующие действия:

- Если *программный объект графа-образца* не создавался раннее, то необходимо создать его.
- Для созданного *программного объекта графа-образца* применить несколько раз *Метод добавления трехэлементной конструкции для заданного графа-образца* (*Метод добавления пятиэлементной конструкции для заданного графа-образца*), задав в качестве трех (пяти) входных параметров ***параметры Метода добавления конструкции для заданного графа-образца*** в зависимости от нужной конфигурации добавляемого *программного объекта трехэлементной (пятиэлементной) конструкции*.

При этом *параметр Метода добавления конструкции для заданного графа-образца* существенно отличается от *параметра Метода создания итератора поиска конструкций в sc-памяти*. Кроме sc-адресов и классов элементов sc-памяти, могут быть указаны локальные идентификаторы этих адресов или классов в созданном графе-образце. Это существенно упрощает процесс формирования *программного объекта графа-образца*, когда необходимо в добавляемой конструкции указать *sc-адрес* или *класс элемента sc-памяти*^, который был уже указан раннее в другой конструкции в заданном *графе-образце*. Таким образом можно сослаться с помощью такого локального идентификатора на параметр уже ранее добавленной конструкции в заданный граф-образец. Кроме того такой способ позволяет получать по таким локальным идентификаторам элементы из найденных по заданному *графу-образцу* конструкций в sc-памяти.

Вне зависимости от того, какие методы были применены для созданного *программного объекта графа-образца* в самой структуре *программного объекта графа образца* для удобства представления и обработки данных создаются только ***программные объекты трехэлементных конструкций***. Каждый элемент в *программном объекте трехэлементной конструкции* кроме sc-адреса, класса и локального идентификатора, имеет свой в рамках этой конструкции номер позиции, заданный в диапазоне от нуля до двух, а также номер позиции в рамках всего графа-образца, вычисляемый как сумма произведения номера *трехэлементной конструкции в заданном графе-образце* и *Числа три* и *номера позиции этого элемента в рамках графа-образца*. Добавление *программного объекта трехэлементной конструкции в заданном программном объекте графа-образца* в *программный объект графа-образца* происходит следующим образом:

- Если указанный в качестве второго параметра аргумент имеет локальный идентификатор в заданном графе-образце и этот локальный идентификатор так же указан для первого или второго аргумента, то завершить работу *Метода добавления трехэлементной конструкции для заданного графа-образца* с исключительной ситуацией о том, что *один и тот же локальный идентификатор одновременно указан для второго и первого (третьего) элементов заданного программного объекта трехэлементной конструкции в заданном программном объекте графа-образца*.
- Если какой-то из параметров задан в качестве класса элемента sc-памяти, соответствующего sc-константе, то завершить работу *Метода добавления трехэлементной конструкции для заданного графа-образца* с исключительной ситуацией о том, что *класс элемента создаваемого программного объекта трехэлементной конструкции в заданном программном объекте графа-образца является классом элементов в sc-памяти, соответствующих sc-константам*.
- Если какой-то из параметров задан в качестве sc-адреса несуществующего элемента в sc-памяти, то завершить работу *Метода добавления трехэлементной конструкции для заданного графа-образца* с исключительной ситуацией о том, что *элемент с sc-адресом элемента создаваемого программного объекта трехэлементной конструкции в заданном программном объекте графа-образца не существует в sc-памяти*.

- Для всех параметров, для которых заданы локальные идентификаторы в заданном графе-образце, а также заданы sc-адреса элементов в sc-памяти, добавить все пары с этими локальными идентификаторами и соответствующими sc-адресами элементов в sc-памяти в отношение *локальный идентификатор элемента sc-памяти и sc-адрес этого элемента в заданном графе-образце**, иначе если известны sc-адреса элементов в sc-памяти для этих локальных идентификаторов в отношении *локальный идентификатор элемента sc-памяти и sc-адрес этого элемента в заданном графе-образце**, указать для заданных параметров известные sc-адреса элементов в sc-памяти.
- Для всех параметров, для которых заданы локальные идентификаторы в заданном графе-образце, а также заданы классы элементов в sc-памяти, добавить все пары с этими локальными идентификаторами и соответствующими классами элементов в sc-памяти в отношение *локальный идентификатор элемента sc-памяти и класс этого элемента в заданном графе-образце**.
- Для всех параметров, для которых заданы только локальные идентификаторы в заданном графе-образце, а также в отношении *локальный идентификатор элемента sc-памяти и номера позиций этого элемента в заданном графе-образце** по этим локальным идентификаторам неизвестны все номера позиций соответствующих элементов в заданном *графе-образце*, то добавить в это отношение все пары с локальными идентификаторами и номерами позиций соответствующих элементов в заданном графе-образце.
- Для полученного *программного объекта трехэлементной конструкции в заданном программном объекте графа-образца* вычислить номер приоритета, необходимый при выполнении *Метода поиска конструкций в sc-памяти, изоморфных данному графу-образцу*:
  - Если в *программном объекте трехэлементной конструкции* для всех элементов указаны *sc-адреса элементов sc-памяти*, то номер приоритета заданной конструкции считается равным нулю (то есть считается самой приоритетной).
  - Если в *программном объекте трехэлементной конструкции* для второго элемента указан *sc-адрес элемента sc-памяти, соответствующего sc-коннектору*, то номер приоритета заданной конструкции считается равным единице (то есть считается второй по приоритету).
  - Если в *программном объекте трехэлементной конструкции* для первого и третьего элементов указаны *sc-адреса элемента sc-памяти*, то номер приоритета заданной конструкции считается равным двойке (то есть считается третьей по приоритету).
  - Если в *программном объекте трехэлементной конструкции* только для третьего элемента указан *sc-адрес элемента sc-памяти*, то номер приоритета заданной конструкции считается равным тройке.
  - Если в *программном объекте трехэлементной конструкции* для первого элемента указан *sc-адрес элемента sc-памяти*, а для третьего элемента — *класс элемента sc-памяти, соответствующего sc-узлу^*, то номер приоритета заданной конструкции считается равным четверке.
  - Если в *программном объекте трехэлементной конструкции* для первого элемента указан *sc-адрес элемента sc-памяти*, а для третьего элемента — *класс элемента sc-памяти, соответствующего sc-коннектору^*, то номер приоритета заданной конструкции считается равным пятерке.
  - Если в *программном объекте трехэлементной конструкции* нет элементов, для которых заданы *sc-адреса элементов sc-памяти*, то номер приоритета заданной конструкции считается равным шестерке (то есть считается самой неприоритетной).

  После определения номера приоритета заданного *программного объекта трехэлементной конструкции в заданном программном объекте графа-образца* добавить данный объект во множество с позицией, равной вычисленному номеру приоритета, *упорядоченного по степени приоритета поиска множества множеств номеров программных объектов трехэлементных конструкций в заданном графе-образце*.
- Полученный *программный объект трехэлементной конструкции заданного графа-образца* добавить в *кортеж программных объектов трехэлементных конструкций в заданном графе-образце*.

Добавление *программного объекта пятиэлементной конструкции* в *программный объект графа-образца* сводится к добавлению двух *программных объектов трехэлементной конструкции* в этот *программный объект графа-образца*, при этом во втором добавляемом *программном объекте трехэлементной конструкции* для третьего элемента указывается только локальный идентификатор второго элемента первого добавляемого *программного объекта трехэлементной конструкции в заданном программном объекте графа-образца* для заданного *программного объекта графа-образца*.

Чтобы найти все *конструкции в sc-памяти*, изоморфные заданному *программному объекту графа-образца* следует выполнить следующие действия:

- Для сформированного *программного объекта графа-образца* применить *Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу*.
- Результатом этого метода будет *кортеж программных объектов всех конструкций в sc-памяти, изоморфных заданному графу-образцу*, который, как и итератор поиска трех- и пяти- элементных конструкций в sc-памяти, имеет свой *программный интерфейс*.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{

кортеж программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу
:= [ScTemplateSearchResult]
∈ *C++*
⊂ *программный объект*

Программный интерфейс кортежа программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу
⊃=
{
⇐ *программный интерфейс**:
кортеж программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу

Метод получения программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по его номеру в кортеже
∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScTemplateSearchResultItem operator[](size_t index) const noexcept(false)]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *32-битовое целое число*
⟩
⇒ *класс результата метода**:
• *программный объект конструкции в sc-памяти, изоморфной заданному графу-образцу*
⇒ *класс исключительных ситуаций**:
• *по заданному номеру нет элемента в кортеже*

Метод получения программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по его номеру в кортеже с предварительной проверкой заданного номера
∈ *метод*
∈ *метод без исключительных ситуаций*
⇒ *заголовок метода на языке представления методов**:
[bool Get(size_t index, ScTemplateSearchResultItem & outItem) const noexcept]
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *32-битовое целое число*
• *программный объект конструкции в sc-памяти, изоморфной заданному графу-образцу*
⟩
⇒ *класс результата метода**:
• *логическое значение*
}

программный объект конструкции в sc-памяти, изоморфной заданному графу-образцу
:= [ScTemplateSearchResultItem]
∈ *C++*
⊂ *программный объект*

Программный интерфейс программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу
⊃=
{
⇐ *программный интерфейс**:
программный объект конструкции в sc-памяти, изоморфной заданному графу-образцу

Метод получения sc-адреса элемента программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по его номеру в этом программном объекте

∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScAddr const & operator[](size_t index) const noexcept(false)**]**
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *32-битовое целое число*
⟩
⇒ *класс результата метода**:
• *sc-адрес элемента sc-памяти*
⇒ *класс исключительных ситуаций**:
• *по заданному номеру нет элемента в программном объекте конструкции в sc-памяти, изоморфной заданному графу-образцу*

Метод получения sc-адреса элемента программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по его номеру в этом программном объекте с предварительной проверкой заданного номера

∈ *метод*
∈ *метод без исключительных ситуаций*
⇒ *заголовок метода на языке представления методов**:
[bool Get(size_t index, ScAddr & outAddr) const noexcept**]**
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *32-битовое целое число*
• *sc-адрес элемента sc-памяти*
⟩
⇒ *класс результата метода**:
• *логическое значение*

Метод получения sc-адреса элемента программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по локальному идентификатору элемента соответствующего программного объекта трехэлементной конструкции в заданном программном объекте графа-образца

∈ *метод*
⇒ *заголовок метода на языке представления методов**:
[ScAddr const & operator[](std::string const & name) const noexcept(false)**]**
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *локальный идентификатор элемента трехэлементной конструкции в заданном графе-образце*
⟩
⇒ *класс результата метода**:
• *sc-адрес элемента sc-памяти*
⇒ *класс исключительных ситуаций**:
• *по заданному локальному идентификатору нет элемента в программном объекте конструкции в sc-памяти, изоморфной заданному графу-образцу*

Метод получения sc-адреса элемента программного объекта конструкции в sc-памяти, изоморфной заданному графу-образцу, по локальному идентификатору элемента соответствующего программного объекта трехэлементной конструкции заданного графа-образца с предварительной проверкой заданного локального идентификатора

∈ *метод*
∈ *метод без исключительных ситуаций*
⇒ *заголовок метода на языке представления методов**:
[bool Get(std::string const & name, ScAddr & outAddr) const noexcept**]**
∈ *C++*
⇒ *классы входных аргументов метода**:
⟨• *локальный идентификатор элемента трехэлементной конструкции в заданном графе-образце*
• *sc-адрес элемента sc-памяти*
⟩
⇒ *класс результата метода**:
• *логическое значение*

}

}
}

Текущий ***Метод поиска конструкций в sc-памяти, изоморфных данному графу-образцу***, состоит из двух этапов: (1) *Этапа предобработки графа-образца*, (2)*Этапа поиска конструкций в sc-памяти, изоморфных заданному графу-образцу*. При этом внутри *Метода поиска конструкций в sc-памяти, изоморфных данному графу-образцу* происходит создание программного *итератора поиска конструкций в sc-памяти, изоморфных данному графу-образцу*, выполняющего весь алгоритм изоморфного поиска.

Библиотека многократно используемых компонентов Программного варианта реализации ostis-платформы
⊃=
{

Программный интерфейс информационно-поисковых методов в Реализации sc-памяти в ostis-платформе
⊃=
{

Программный интерфейс программного объекта графа-образца
⊃=
{

итератор поиска конструкций в sc-памяти, изоморфных данному графу-образцу
:= [ScTemplateSearch]
∈ *C++*
⊂ *программный объект*
⇒ *понятие, специфицирующее заданную сущность**:
{• *локальный идентификатор некоторого элемента в некотором программном объекте трехэлементной конструкции заданного графа-образца и множество всех номеров программных объектов трехэлементных конструкций в заданном графе-образце с этим элементом**
:= [std::map<std::string, std::unordered_set<size_t» m_templateItemsNamesToDependedTemplateTriples]
• *кортеж множеств номеров программных объектов трехэлементных конструкций компонент связности в заданном графе-образце*
:= [std::vector<std::unordered_set<size_t» m_connectivityComponentsTemplateTriples]
• *множество номеров самых приоритетных для поиска программных объектов трехэлементных конструкций компонент связности в заданном графе-образце*
:= [std::vector<size_t> m_connectivityComponentPriorityTemplateTriples]
• *кортеж множеств sc-адресов элементов sc-памяти, соответствующих sc-коннекторам, таких трехэлементных конструкций, которые не изоморфны соответствующим трехэлементным конструкциям заданного графа-образца, номера которых равны номерам позиций множеств в этом ориентированном множестве*
:= [std::vector<std::unordered_set<ScAddr, ScAddrHashFunc<uint32_t»> m_notUsedEdgesInTemplateTriples]
• *кортеж множеств sc-адресов элементов sc-памяти, соответствующих sc-коннекторам, таких трехэлементных конструкций, которые изоморфны соответствующим трехэлементным конструкциям заданного графа-образца, номера которых равны номерам позиций множеств в этом ориентированном множестве*
:= [std::vector<std::unordered_set<ScAddr, ScAddrHashFunc<uint32_t»> m_usedEdgesInTemplateTriples]
• *кортеж множеств sc-адресов элементов sc-памяти, соответствующих sc-коннекторам, являющихся вторыми элементами соответствующих трехэлементных конструкций найденных конструкций в sc-памяти, изоморфных заданному графу-образцу, номера которых равны номерам позиций множеств в этом ориентированном множестве*
:= [std::vector<std::unordered_set<ScAddr, ScAddrHashFunc<uint32_t»> m_usedEdgesInReplacementConstructions]
• *кортеж множеств номеров программных объектов трехэлементных конструкций заданного графа-образца для найденных по нему изоморфных конструкций в sc-памяти, номера которых равны номерам позиций множеств в этом ориентированном множестве*
:= [std::vector<std::unordered_set<size_t» m_checkedTemplateTriplesInReplacementConstructions]
• *номер последней найденной конструкции в sc-памяти по заданному графу-образцу*
:= [size_t m_lastReplacementConstructionIdx]
• *множество номеров всех найденных и изоморфных конструкций в sc-памяти по заданному графу-образцу*
:= [std::unordered_set<size_t> m_foundReplacementConstructions]

```
    }
}
}
}
```

Этап предобработки графа-образца состоит из следующих промежуточных шагов обработки:

- Добавление в отношение *локальный идентификатор некоторого элемента в некотором программном объекте трехэлементной конструкции заданного графа-образца и множество всех номеров программных объектов трехэлементных конструкций в заданном графе-образце с этим элементом** всех пар с локальными идентификаторами элементов в некотором программном объекте трехэлементной конструкции заданного графа-образца и множествами номеров соответствующих им программных объектов трехэлементных конструкций в заданном графе-образце. Во множества таких программных объектов не включаются те *программные объекты трехэлементных конструкций* в заданном графе-образце, номера которых входят в локальный идентификатор самого элемента. Этот локальный идентификатор элемента не является локальным идентификатором этого элемента в рамках всего графа-образца. Такой локальный идентификатор формируется самой системой, а не пользователем *Метода добавления трехэлементной конструкции для заданного графа-образца* (*Метода добавления пятиэлементной конструкции для заданного графа-образца*) и состоит из локального идентификатора элемента в рамках всего графа-образца и номера соответствующего программного объекта трехэлементной конструкции в заданном графе-образце. Зная такой локальный идентификатор элемента некоторого *программного объекта трехэлементной конструкции* графа-образца, можно быстро получать доступам к другим *программным объектам трехэлементных конструкций*, в которых есть данный элемент.
- Удаление из множеств номеров *программных объектов трехэлементных конструкций*, которые являются вторыми компонентами пар отношения *локальный идентификатор некоторого элемента в некотором программном объекте трехэлементной конструкции заданного графа-образца и множество всех номеров программных объектов трехэлементных конструкций в заданном графе-образце с этим элементом** и в которых есть элементы, являющиеся первыми компонентами этих пар, всех таких номеров *программных объектов трехэлементных конструкций*, переход по которым в процессе поиска изоморфных по заданному графу-образцу конструкций может привести к зацикливанию алгоритма поиска конструкций по заданному графу-образцу. Данный этап предобработки сформированного *программного объекта графа-образца* позволяет максимально возможным образом устранить переходы по таким *программным объектам трехэлементных конструкций* заданного графа-образца, существенно усложняющие процесс изоморфного поиска конструкций по заданному графу-образцу. Такими *программными объектами трехэлементных конструкций* являются программные объекты трехэлементных конструкций, например, у которых первыми элементами является sc-адрес одного и того же *элемента sc-памяти, соответствующего sc-структуре*, или *класс элемента sc-памяти^, соответствующего переменной sc-структуре*, или sc-адрес одного и того же *элемента sc-памяти, соответствующего sc-узлу неролевого отношения*. Поскольку сам SC-код позволяет представлять конструкции любой возможной конфигурации, то нельзя точно сказать, какие именно конфигурации конструкций могут приводить к циклическим ситуациям при выполнении алгоритма изоморфного поиска этих конструкций по заданному графу-образцу. Более универсальный алгоритм устранения циклов в графе-образце может привести к существенным дополнительным временным затратам, поскольку может потребовать более глубокого синтаксического анализа в исходном графе-образце, поэтому рекомендуется реализовывать условия, по которым можно определять программные объекты трехэлементных конструкций, переход по которым может привести к циклическим ситуациям в обработке графа-образца. Устранение циклов в графе-образце позволяет алгоритму изоморфного поиска конструкций по заданному графу-образцу более эффективно производить все требуемые операции над графами, поэтому он является ключевым этапом предобработки исходного *программного объекта графа-образца*. Также данный шаг нельзя выполнять вместе с предыдущим шагом, поскольку для устранения всех циклов в графе-образце необходимо знать полностью все возможные переходы по этому графу-образцу.
- Поиск всех компонент связности в заданном графе-образце, то есть не связанных между собой подграфов в этом графе, и добавление всех соответствующих этим компонентам связности *программных объектов трехэлементных конструкций* во множества *кортежа множеств номеров программных объектов трехэлементных конструкций компонент связности в заданном графе-образце*. Таким образом, это позволяет найти даже такие компоненты связности, которые могли быть получены после выполнения второго шага алгоритма предобработки заданного графа-образца, то есть устранения циклов в заданном графе-образце. Разбиение графа-образца на компоненты связности могло быть одним из решений в задаче устранения цикла в заданном графе-образце, однако сам алгоритм изоморфного поиска конструкций по заданному графу-образцу является более совершенным и позволяет за один обход итератора поиска трехэлементных конструкций находить все трехэлементные конструкции для тем трехэлементных конструкций графа-образца, которые имеют один и тот же первый или третий элемент, поэтому разбиение графа-образца на компоненты связности не используется в предыдущем шаге.

- Последним шагом *Этапа предобработки графа-образца* является выделение в найденных на предыдущем шаге *Этапа предобработки графа-образца* компонентах связности самых приоритетных для поиска *программных объектов трехэлементных конструкций* в заданном графе-образце. Самым приоритетным программным объектом трехэлементной конструкции является объект, имеющий номер приоритета, равный нулю, самым неприоритетным — объект, имеющий номер приоритета, равный шести. При этом, если есть несколько *программных объектов трехэлементных конструкций*, которые имеют один и тот же номер приоритета, то самым приоритетным из них считается тот объект, у которого первый (третий) элемент, являющийся sc-адресом элемента sc-памяти, имеет наименьшее количество элементов, соответствующих выходящим (входящим) sc-коннекторам. В результате формируется *множество номеров самых приоритетных для поиска программных объектов трехэлементных конструкций компонент связности в заданном графе-образце*.

Таким образом, *Этап предобработки графа-образца* позволяет существенно упростить обработку графа-образца на этапе поиска изоморфных ему конструкций. Следующий ***Этап поиска конструкций в sc-памяти, изоморфных заданному графу-образцу*** включает следующие шаги:

- Если заданный *кортеж программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу* не пустой, то удалить из него все программные объекты конструкций в sc-памяти.
- Если *множество номеров самых приоритетных для поиска программных объектов трехэлементных конструкций компонент связности в заданном графе-образце* является пустым, то это означает, что заданный *граф-образец* является пустым. В таком случае результатом поиска является пустой *кортеж программных объектов конструкций в sc-памяти, изоморфных заданному графу-образцу* и *Этап поиска конструкций в sc-памяти, изоморфных заданному графу-образцу* завершается с успешным результатом.
- Инициализировать *номер последней найденной конструкции в sc-памяти по заданному графу-образцу* значением, равным нулю. Задать *Номер текущей найденной конструкции в sc-памяти по заданному графу-образцу*, равным *номеру последней найденной конструкции в sc-памяти по заданному графу-образцу*. Задать *множество номеров "равносильных" программных объектов конструкций*, равным множеству номеров самых приоритетных для поиска программных объектов трехэлементных конструкций компонент связности в заданном графе-образце. Задать *множество номеров текущих программных объектов конструкций*, равным *множество номеров "равносильных" программных объектов конструкций*.
- Из *множество номеров "равносильных" программных объектов конструкций* выбрать следующий номер. По полученному номеру из *кортежа программных объектов трехэлементных конструкций в заданном графе-образце* взять соответствующий ему *программный объект трехэлементной конструкции заданного графа-образца* в этом графе-образце.
- Для *выбранного программного объекта трехэлементной конструкции* в заданнои графе-образце найти все такие *программные объекты трехэлементных конструкций*, (1) у элементов которых заданные классы и sc-адреса совпадают с классами и sc-адресами элементов *выбранного программного объекта трехэлементной конструкции* соответственно, при этом либо первые, либо третие их элементы имеют одинаковые локальные идентификаторы в заданном графе-образце или не имеют их вовсе, (2) для которых не были найдены соответствующие замены, то есть множеству, расположенному в *кортеже множеств номеров программных объектов трехэлементных конструкций заданного графа-образца для найденных по нему изоморфных конструкций в sc-памяти, номера которых равны номерам позиций множеств в этом ориентированном множестве* по *номеру текущей найденной конструкции в sc-памяти по заданному графу-образцу*, не принадлежат номера найденных *программных объектов трехэлементных конструкций*, а также номера которых не принадлежат *множеству номеров текущих программных объектов конструкций*. *Множество номеров "равносильных" программных объектов конструкций* сделать равным множеству всех найденных *программные объекты трехэлементных конструкций*.
- Если полученное *множество номеров "равносильных" программных объектов конструкций* является пустым, то завершить данную итерацию алгоритма.
- Если полученное *множество номеров "равносильных" программных объектов конструкций* не является пустым, то из *множества номеров "равносильных" программных объектов конструкций* выбрать случайный номер. По полученному номеру из *кортежа программных объектов трехэлементных конструкций в заданном графе-образце* взять соответствующий ему *программный объект трехэлементной конструкции заданного графа-образца* в этом графе-образце.
- По полученному *программному объекту трехэлементной конструкции* создать *итератор поиска трехэлементных конструкций в sc-памяти*. Создание *итератора поиска трехэлементных конструкций в sc-памяти* происходит при помощи *Метода создания итератора поиска трехэлементных конструкций в sc-памяти*. Параметры метода назначаются элементы заданного *программного объекта трехэлементной конструкции*, при этом вместо *классов элементов sc-памяти, соответствующих sc-переменным* (*классов элементов sc-памяти, соответствующих sc-метапеременным*), используются соответствующие им *классы элементов sc-памяти, соответствующие sc-константам* (*классы элементов sc-памяти, соответствующие sc-переменным*). То есть, например, если у некоторого элемента *программного объекта трехэлементной конструкции* задан *класс элементов sc-памяти, соответствующих переменным sc-узлам*, то вместо него для *Метода создания итератора поиска трехэлементных конструкций в sc-памяти* используется соответ-

ствующий ему *класс элементов sc-памяти, соответствующих константным sc-узлам*. Таким образом, при переходе от *программного объекта трехэлементной конструкции* к *программному итератору поиска трехэлементных конструкций sc-памяти* понижается степень переменности элементов *программного объекта трехэлементной конструкции*: *классы элементов sc-памяти, соответствующих sc-метапеременным* преобразуются в *классы элементов sc-памяти, соответствующих sc-переменным*, а *классы элементов sc-памяти, соответствующих sc-переменным* — в *классы элементов sc-памяти, соответствующих sc-константам*. Если *класс элемента программного объекта трехэлементной конструкции* не является нестрогим подмножеством *класса элементов sc-памяти, соответствующих sc-метапеременным* или *класса элементов sc-памяти, соответствующих sc-переменным*, то степень переменности не понижается.
- С помощью *Метода перехода к следующей "подходящей"для заданного итератора конструкции в sc-памяти* перейти к *конструкции в sc-памяти*, изоморфной заданной конструкции в графе-образце.

Несмотря на широкий спектр задач, которые можно решать при помощи текущей реализации *изоморфного поиска*, существует ряд причин, по которым не следует использовать эту и другие реализации *изоморфного поиска*:
- *изоморфный поиск* позволяет решать широкий спектр задач, если все знания, изоморфные заданному *графу-образцу*, находятся в *базе знаний*, либо отсутствуют. То есть уровень качества *изоморфного поиска* напрямую зависит от состояния *базы знаний*. Чем разнообразнее *фрагменты базы знаний*, тем хуже *изоморфный поиск* в работе;
- Затраты на поиск при больших *графах-образцах* могут расходится с желаниями разработчика или пользователя. Чем больше *граф-образец*, тем больше ситуаций, в которых может быть нестандартное поведение алгоритма *изоморфного поиска*.
- Большую часть задач, решаемых при помощи *изоморфного поиска*, можно и нужно решать при помощи *итераторов поиска трех- и пяти- элементных конструкций*. Чем проще метод решения задач, тем меньше ошибок и внештатных ситуаций можно получить.

Пункт 6.3.4.4. Общее описание процесса и Пример трансляции sc-текста в sc-память ostis-платформы

Любой *sc-конструкции* взаимно-однозначно соответствует *конструкция sc-памяти ostis-платформы*. Погрузка *sc-конструкции* в *sc-память ostis-платформы* означает трансляцию каждого sc-элемента этой *sc-конструкции* и связей инцидентности между этими sc-элементами в *sc-память ostis-платформы*, то есть трансляцию синтаксической структуры sc-конструкции в соответствующее представление внутри *памяти ostis-системы*. В общем случае, алгоритм погрузки любой произвольной sc-конструкции в *sc-память ostis-платформы* состоит из следующих этапов:
- Выделение sc-узлов и их синтаксических и семантических классов в sc-конструкции и создание соответствующих элементов в sc-памяти ostis-платформы при помощи *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-узлу*;
- Выделение всех независимых sc-коннекторов и их синтаксических и семантических классов в sc-конструкции (то есть sc-коннекторов, у которых начальным и конечным sc-элементом не являются другие sc-коннекторы) между sc-узлами, для которых были созданы на шаге 1 соответствующие элементы в sc-памяти, и создание соответствующих элементов в sc-памяти ostis-платформы при помощи *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-коннектору*;
- Возврат в пункт 2, если остались нетранслированные в sc-память sc-коннекторы (то есть sc-коннекторы, для которых не были созданы соответствующие элементы в sc-памяти ostis-платформы);
- Погрузка содержимого всех внутренних файлов ostis-системы в файловую память ostis-платформы (данный подпроцесс рассмотрен в *Пункт 6.3.5.4. Описание процесса и Пример трансляции внешних информационных конструкций в файловую память ostis-платформы*).

На рисунке *SCg-текст. Пример трансляции sc-текста в sc-память ostis-платформы* изображен пример спецификации представления sc-конструкции в *sc-памяти ostis-платформы*. Здесь каждому sc-элементу заданной sc-конструкции ставится в соответствие sc-элемент, обозначающий элемент sc-памяти, соответствующий непосредственно этому sc-элементу. Для каждого sc-элемента, обозначающего элемент sc-памяти некоторого sc-элемента заданной sc-конструкции описываются связи между элементами sc-памяти, а также синтаксические и семантические классы этих элементов.

Пункт 6.3.4.5. Достоинства и недостатки Программного интерфейса Реализации sc-памяти в ostis-платформе

Текущий *Программный интерфейс Реализации sc-памяти в ostis-платформе* позволяет:

- В необходимой и достаточной мере реализовывать платформенно-зависимые подсистемы текущего *Программного варианта реализации ostis-платформы*, практически независимо от *Реализации sc-памяти в ostis-платформе*. То есть текущий *Программный интерфейс Реализации sc-памяти в ostis-платформе* является способом унификации доступа к программной *Реализации sc-памяти в ostis-платформе* и позволяет легко подменять различные варианты реализации sc-памяти на *языке представления методов C++*, при этом сам *Программный интерфейс Реализации sc-памяти в ostis-платформе* практически не меняется или не изменяется вовсе.
- Реализовывать базовые инструментальные средства для проектирования платформенно-независимых ostis-систем, например, *Реализацию scp-интерпретатора*.
- Формировать и расширять *Библиотеку многократно используемых компонентов Программного варианта реализации ostis-платформы* за счет компонентов, использующих методы *Реализации sc-памяти в ostis-платформе* и входящих в состав различных программных расширений текущего *Программного интерфейса Реализации sc-памяти в ostis-платформе*.
- Обеспечить различные уровни доступа к *Реализации sc-памяти в ostis-платформе*, в том числе уровни доступа различных пользователей *Программного варианта реализации ostis-платформы*.

Стоит отметить, что *Программный интерфейс Реализации sc-памяти в ostis-платформе* не может существовать отдельно от текущей *Реализации sc-памяти в ostis-платформе*. Кроме того, он является частью *Реализации sc-памяти в ostis-платформе*, то есть проектируется и разрабатывается согласованно с реализацией самой sc-памяти. Однако при необходимости он может быть использован для различных модификаций или версий текущей *Реализации sc-памяти в ostis-платформе*.

Несмотря на широкий спектр функциональных возможностей текущего *Программного интерфейса Реализации sc-памяти в ostis-платформе*, к его недостатком можно отнести следующее:

- На уровне *Программного интерфейса Реализации sc-памяти в ostis-платформе* никак не ограничивается диапозон классов sc-элементов в sc-памяти, которые могут быть установлены в качестве аргументов, например, для *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-узлу* и *Метода создания элемента в sc-памяти с заданным классом, соответствующего некоторому sc-коннектору*.
- Из-за недостатков текущей реализации агентной архитектуры в *Программном варианте реализации ostis-платформы* нельзя для *Программного интерфейса Реализации sc-памяти в ostis-платформе* использовать *Реализацию sc-памяти в ostis-платформе*, хранящуюся в виде скомпилированного файла. Прежде всего это связано с тем, что платформенно-зависимые агенты реализуются средствами, которые используют создание исходных файлов при сборке всей платформы. Таким образом скомпилированные файлы остаются зависимыми от того устройства, где они были собраны.

§ 6.3.5. Реализация файловой памяти в Программной платформе ostis-систем

⇒ *подраздел**:
- *Пункт 6.3.5.1. Спецификация Метаязыка описания представления конструкций, не принадлежащих SC-коду, в файловой памяти ostis-платформы*
- *Пункт 6.3.5.2. Алфавит и Синтаксис SCfin-кода*
- *Пункт 6.3.5.3. Денотационная семантика SCfin-кода*
- *Пункт 6.3.5.4. Описание процесса и Пример трансляции внешних информационных конструкций в файловую память ostis-платформы*
- *Пункт 6.3.5.5. Достоинства и недостатки Реализации файловой памяти ostis-платформы и Метаязыка описания представления внешних информационных конструкций в файловой памяти ostis-платформы*

⇒ *ключевой знак**:
- *Реализация файловой памяти в ostis-платформе*
- *SCfin-код*

⇒ *ключевое знание**:
- *Спецификация Реализации файловой памяти в ostis-платформе*
- *Принципы, лежащие в основе Реализации файловой памяти в ostis-платформе*
- *Ответ на Вопрос. Как кодируются внешние информационные конструкции в текущей Реализации файловой памяти в ostis-платформе*
  ≔ [Спецификация Метаязыка описания представления внешних информационных конструкций в файловой памяти ostis-платформы]
- *Достоинства и недостатки Реализации файловой памяти в ostis-платформе*

SC-код является универсальным средством для представления любых видов знаний, но (!) не всегда есть необходимость погружать что-либо в *sc-память ostis-платформы*, по крайней мере, на ранних стадиях развития *ostis-платформы*. Это может быть объяснено и тем, что *информационные конструкции, не принадлежащие SC-коду*, достаточно сложны в понимании для неподготовленного *пользователя ostis-системы*. Для решения таких проблем на уровне *Алфавита SC-кода^* вводится дополнительный элемент — *внутренний файл ostis-системы*. С помощью *внутренних файлов ostis-системы* можно представлять, хранить, обрабатывать и визуализировать *информационные конструкции, не принадлежащие SC-коду*.

Поэтому при реализации *sc-памяти* необходимо учитывать необходимость хранения *информационных конструкций, не принадлежащих SC-коду*, с помощью *SC-кода*. Кроме собственно *sc-памяти Реализация sc-памяти в ostis-платформе* включает также *Реализацию файловой памяти ostis-платформы*, предназначенную для хранения содержимого *внутренних файлов ostis-систем*, то есть *внешние информационные конструкции* (линейные тексты, изображения, видео и так далее).

За весь период развития *Программного варианта реализации ostis-платформы* было достаточно много попыток реализовать полно функциональную и быструю файловую память на базе популярных баз данных. Однако из всех этих решений не были учтены потенциальные проблемы при *Реализации информационно-поисковой подсистемы Программного варианта реализации ostis-платформы*. Сейчас *файловая память* реализована своими средствами, в качестве структур данных для хранения *информационных конструкций*, не принадлежащих *SC-коду*, используются префиксные деревья *Bayer R..PrefiBT-1977art* и линейные списки.

Выбор аргументируется тем, что:
- префиксные структуры достаточно просты в понимании и минимальны в своем синтаксисе;
- с помощью префиксных структур достаточно удобно хранить и обрабатывать связи типа "ключ-значение";
- доступ к значению по ключу происходит в худшем случае за длину этого ключа *Belazzougui D..FastPSiLSwA-2010art*;
- за счет того, что префиксы склеиваются, выходит сильный выигрыш в использовании памяти.

Реализация файловой памяти в ostis-платформе
∈ *реализация файловой памяти на основе префиксного дерева*
⇐ *программная модель**:
файловая память ostis-платформы
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
∈ *атомарный многократно используемый компонент ostis-систем*
∈ *зависимый многократно используемый компонент ostis-систем*
⇒ *зависимости компонента**:
{• *Библиотека методов и структур данных GLib*
}
⇒ *язык представления методов**:
• *C*
⇒ *внутренний язык**:
• *SCfin-код*

Пункт 6.3.5.1. Спецификация Метаязыка описания представления конструкций, не принадлежащих SC-коду, в файловой памяти ostis-платформы

Для описания представления *информационных конструкций*, не принадлежащих *SC-коду*, внутри *файловой памяти ostis-платформы* также необходим *sc-язык*. Такой *sc-язык* называется ***Метаязыком описания представления информационных конструкций, не принадлежащих SC-коду, в файловой памяти ostis-платформы***, или, кратко, ***SCfin-кодом*** *(Semantic Code file interior). файловая память* текстов, не принадлежащих *SC-коду* (как абстрактную модель, по которой можно реализовывать соответствующую ей *файловую память ostis-платформы*), можно рассматривать как подмножество *sc.fin-текста*.

SCfin-код
:= [Semantic Code file interior]
:= [Язык описания представления информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти ostis-платформы]
:= [Метаязык описания представления информационных конструкций, не принадлежащих SC-коду, внутри файловой памяти ostis-платформы]
⇒ *часто используемый sc-идентификатор**:
[sc.fin-текст]

∈ *имя нарицательное*
∈ *абстрактный язык*
∈ *метаязык*
∈ *sc-язык*
⊂ *SC-код*
⊃ *файловая память ostis-платформы*

Пункт 6.3.5.2. Алфавит и Синтаксис SCfin-кода

Синтаксис SCfin-кода задается: (1) *Алфавитом SCfin-кода^*, (2) отношением порядка *последовательность в линейном тексте**.

Алфавит SCfin-кода^
:= [синтаксический тип элемента файловой памяти ostis-платформы]
:= [Множество типов элементов файловой памяти ostis-платформы]
⇐ *алфавит**:
SCfin-код
= {• *элемент файловой памяти ostis-платформы, соответствующий подстроке текста линейного языка*
}

Алфавит SCfin-кода^ состоит из одного синтаксически выделяемого типа элементов файловой памяти — *элемента файловой памяти ostis-платформы, соответствующего подстроке текста линейного языка*.

элемент файловой памяти ostis-платформы, соответствующий подстроке текста линейного языка
∈ *sc-элемент*
:= [элемент файловой памяти ostis-платформы]
:= [ячейка файловой памяти ostis-платформы]
:= [образ подстроки информационной конструкции, не принадлежащей SC-коду, в рамках файловой памяти ostis-платформы]

Отношение ***последовательности в линейном тексте**** определяется как бинарное ориентированное отношение порядка, компонентами каждой ориентированной пары которого являются элементы файловой памяти ostis-платформы, соответствующие некоторым подстрокам линейного текста, в результате конкатенации которых образуется подстрока, принадлежащая этому же линейному тексту.

Синтаксис SCfin-кода достаточно прост, поскольку информационные конструкции на нем задаются при помощи *Алфавита SCfin-кода^*, мощность которого равна 1, и единственного отношения инцидентности *последовательности в линейном тексте**. Иерархии синтаксических элементов как таковые не выделяются, поскольку в этом нет необходимости.

На уровне реализации важно выделить семантические классы *элементов файловой памяти ostis-платформы, соответствующих подстроке текста линейного языка*, которые обозначают некоторую префиксную или постфиксную часть целой информационной конструкции.

Пункт 6.3.5.3. Денотационная семантика SCfin-кода

Семантическая классификация элементов SCfin-кода
⊃=
{

элемент файловой памяти ostis-платформы, соответствующий тексту линейного языка
⇒ *разбиение**:
Типология элементов по месту расположения подстроки в линейном тексте
= {• *элемент файловой памяти ostis-платформы, соответствующий префиксной подстроке текста линейного языка*
• *элемент файловой памяти ostis-платформы, соответствующий постфиксной подстроке текста линейного языка*
}

}

На *SCfin-коде* достаточно просто задавать *информационные конструкции* любых линейных текстов. Однако с точки зрения реализуемой модели sc-памяти есть необходимость задавать не столько форму *информационных конструкций*, не принадлежащих *SC-коду*, внутри *файловой памяти ostis-платформы*, сколько связи между этими внутренними *файлами ostis-системы*, являющихся знаками *SC-кода*. Одновременно должны быть реализованы на уровне *sc-памяти* и ***Метод получения файлов ostis-системы, которые содержат заданную внешнюю информационную конструкцию***, и ***Метод получения внешних информационных конструкций из заданных файлов ostis-системы***.

Пункт 6.3.5.4. Описание процесса и Пример трансляции внешних информационных конструкций в файловую память ostis-платформы

На рисунке *SCg-текст. Пример спецификации представления информационных конструкций, не принадлежащих SC-коду, в памяти ostis-системы* показано представление *информационных конструкций, не принадлежащих SC-коду* и соответствия между файлами ostis-системы и информационными конструкциями. С помощью отношения *множество sc-адресов файлов ostis-системы по префиксам их содержимого** задается бинарная ориентированная пара, первым компонентом которой является префиксная структура, элементами которой являются подстроки внешних информационных конструкций, а вторым компонентом — множество соответствующих им sc-адресов файлов ostis-системы, а с помощью отношения *множество постфиксов содержимого файлов ostis-системы по их sc-адресам** задается бинарная ориентированная пара, первым компонентом которой является префиксная структура, элементами которой являются подстроки sc-адресов файлов ostis-системы, представленных в строковом виде, а вторым компонентом — множество соответствующих им постфиксов внешних информационных конструкций префиксной структуры, являющейся первым компонентом каждой пары отношения *множество sc-адресов файлов ostis-системы по префиксам их содержимого**.

Пункт 6.3.5.5. Достоинства и недостатки Реализации файловой памяти ostis-платформы и Метаязыка описания представления внешних информационных конструкций в файловой памяти ostis-платформы

Используемая *Реализация файловой памяти ostis-платформы* полностью оправдывает себя при взаимодействии с системой. Благодаря использованию префиксных структур асимптотические сложности метода получения множества внешних информационных конструкций из заданных *файлов ostis-системы* и метода получения множества файлов ostis-системы по заданным *внешним информационным конструкциям* линейны, так как зависит от длины заданной строки и структуры префиксного дерева.

Среди общих недостатков данной *Реализации файловой памяти ostis-платформы* можно выделить следующие:

- *информационные конструкции*, не принадлежащие SC-коду, пока полностью хранятся в оперативной памяти компьютерного устройства, на котором развернута платформа. Данную проблему можно решить, если в оперативной памяти хранить только первые знаки подстрок информационных конструкций, а остальные части этих подстрок хранить на уровне файлового системы;
- на данный момент информационно-поисковая подсистема реализована не полностью. *Реализация файловой памяти ostis-платформы* позволяет быстро решать задачи поиска внешних информационных конструкций по их префиксным подстрокам, однако не позволяет быстро решать задачи поиска информационных конструкций по любой подстроке, даже для которого задан некоторый шаблон-образец.

Описанные проблемы будут решены в рамках будущей версии *Программного варианта реализации ostis-платформы*.

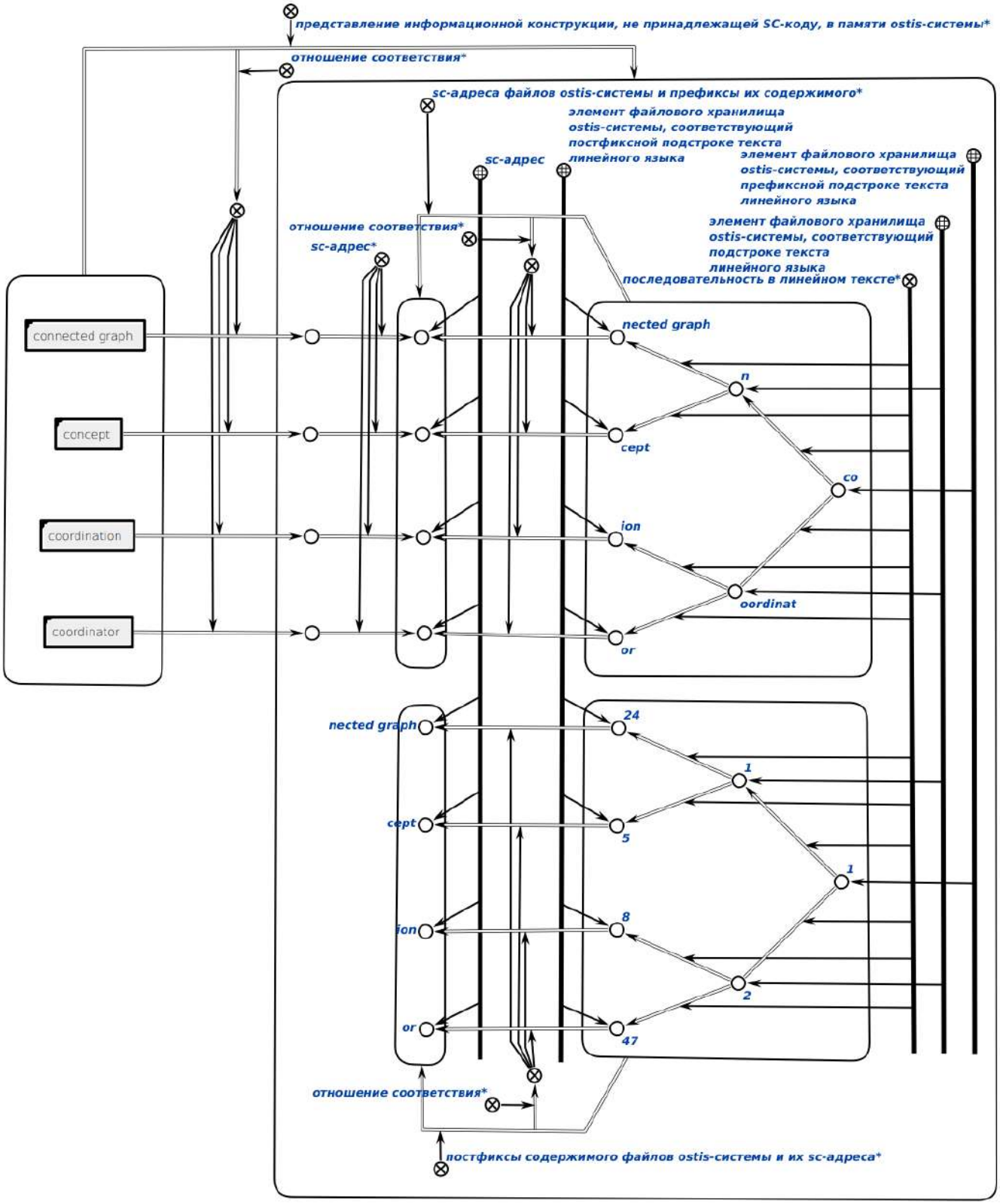
§ 6.3.6. Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе

⇒ *подраздел**:

- *Пункт 6.3.6.1. Спецификация Метаязыка представления сообщений между подсистемами ostis-платформы. SC-JSON-код*
- *Пункт 6.3.6.2. Синтаксис SC-JSON-кода*
- *Пункт 6.3.6.3. Грамматика SC-JSON-кода*

SCg-текст. Пример спецификации представления информационных конструкций, не принадлежащих SC-коду, в памяти ostis-системы

=



- *Пункт 6.3.6.4. Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы*

⇒ *ключевой знак**:
- *Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*
- *JSON*
- *Websocket*
- *HTTP*
- *Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы*
- *Реализация клиентской системы на языке программирования Python*
- *Реализация клиентской системы на языке программирования TypeScript*
- *Реализация клиентской системы на языке программирования C#*
- *Реализация клиентской системы на языке программирования Java*
- *SC-JSON-код*

⇒ *ключевое понятие**:
- *команда на SC-JSON-коде*
- *ответ на команду на SC-JSON-коде*

⇒ *ключевое знание**:
- *Спецификация Реализации подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*
- *Принципы, лежащие в основе Реализации подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*
- *Отличие Websocket от HTTP*
- *Ответ на Вопрос. Как реализуется взаимодействие между подсистемами в ostis-платформе*
  ≔ [Спецификация Метаязыка описания представления сообщений между подсистемами ostis-платформы]

Зачастую, просто *Программного интерфейса Реализации sc-памяти в ostis-платформе* недостаточно, чтобы получать доступ к ее элементам. Это обуславливается тем, что:

- Для реализации задаче-ориентированных компонентов платформы необходимо учитывать функциональные возможности *языков программирования*, используемых для их разработки. Так, например, для реализации интерфейсов по ряду причин лучше использовать *языки программирования* более высокого уровня по сравнению с *C++* (*JavaScript*, *TypeScript* и так далее).
- В силу существующего многообразия форм представления программ необходимо иметь возможность интегрировать уже существующие решения в области современных технологий программирования с существующим *Программным вариантом реализации ostis-платформы*.
- Данный интерфейсе требует того, чтобы клиентское приложение, обращающееся к программной модели *sc-памяти*, фактически составляло с ней единое целое, таким образом исключается возможность построения распределенного коллектива *ostis-систем* на базе реализуемого *Программного варианта реализации ostis-платформы*.

По этим причинам в *Программном варианте реализации ostis-платформы* необходимо иметь дополнительные средства доступа к памяти платформы. Такие средства могут быть реализованы в виде серверной подсистемы текущего *Программного варианта реализации ostis-платформы*, с помощью которой можно общаться с *памятью ostis-платформы* через современные сетевые протоколы.

В общем случае такая подсистема взаимодействия ostis-платформы с внешней средой может быть реализована по-разному. Так, например, до момента реализации текущей подсистемы ранее существовал ее аналог на языке программирования Python, который использовал *протокол HTTP* и бинарное представление команд и ответов. Поэтому можно говорить о большом разнообразии таких подсистем, которые будут составлять множество всевозможных *Реализаций подсистемы взаимодействия с внешней средой с использованием сетевых языков*.

Данная ***Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*** позволяет ostis-системам взаимодействовать с системами из внешней среды на основе общепринятого транспортного формата передачи данных JSON и предоставляет сетевое API для доступа к sc-памяти *Программного варианта реализации ostis-платформы*.

Реализация подсистемы взаимодействия с внешней средой с использованием сетевых языков

⇒ *декомпозиция программной системы**:
{• *Реализация подсистемы взаимодействия с внешней средой с использованием сетевых языков на основе языка JSON*
}

Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе

:= [Подсистема взаимодействия с sc-памятью на основе формата JSON]
:= [Сетевой программный интерфейс Реализации sc-памяти в ostis-платформе]
:= [Предлагаемый нами вариант реализации механизма доступа к sc-памяти ostis-платформы в распределенном коллективе ostis-систем]
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
∈ *неатомарный многократно используемый компонент ostis-систем*
∈ *зависимый многократно используемый компонент ostis-систем*
∈ *клиент-серверная система*
⇒ *используемый язык представления методов**:
- *C*
- *C++*
- *Python*
- *TypeScript*
- *C#*
- *Java*

⇒ *используемый язык**:
- *SC-JSON-код*

⇒ *декомпозиция программной системы**:
{• *Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы*
}
= {• *Реализация клиентской системы на языке программирования Python*
• *Реализация клиентской системы на языке программирования TypeScript*
• *Реализация клиентской системы на языке программирования C#*
• *Реализация клиентской системы на языке программирования Java*
}
}

Взаимодействие с sc-памятью обеспечивается с помощью передачи информации на *SC-JSON-коде* и ведется, с одной стороны, между сервером, являющегося частью ostis-системы, написанным на том же языке реализации этой *ostis-системы* и имеющим доступ к ее sc-памяти, и с другой стороны множеством клиентов, которым известно о наличии сервера в пределах сети их использования. С помощью подсистемы взаимодействия с sc-памятью на основе языка JSON можно взаимодействовать с *ostis-системой* на таком же множестве возможных операций, как и в случае, если бы взаимодействие происходило (непосредственно) напрямую, на том же языке реализации *ostis-платформы*. При этом результат работы отличается только скоростью обработки информации.

Среди эффективных протоколов, используемых при реализации *клиент-серверных систем*, стоит отметить протоколы прикладного уровня стека *TCP/IP* — протоколы ***HTTP*** и ***WebSocket*** *Bhumij G..aOverv oWStFoRT-2018art*. Целесообразным является использование протокола *WebSocket*, это объясняется следующими причинами:

- *WebSocket* выгодно использовать в веб-ориентированных системах, в которых данные, отправляемые сервером, представляются или сохраняются на стороне клиента. В *WebSocket* данные постоянно передаются через одно и то же открытое *соединение*, поэтому коммуникация по протоколу *WebSocket* быстрее чем коммуникация по HTTP *Tomasseti M.aAnaly otPoWiVP-2021art* , *Bhumij G..aOverv oWStFoRT-2018art* . Это очень важно с точки зрения проектирования *Экосистемы OSTIS*, которые могут состоять из десятков тысяч различного вида *ostis-систем*.
- Поскольку в основе *ostis-систем* лежит идея агентно-ориентированной обработки знаний (асинхронной обработки), а память таких систем должна быть одновременно распределенной и общей, то необходимо, чтобы каждая из них (в частности, самостоятельная *ostis-система*) могла общаться с другими *ostis-системами*. Причем такое общение может и должно происходить посредством инициирования событий в памяти этих систем. Отсюда следует однозначный вывод, что протокол *HTTP* не может быть использован в передовых *интеллектуальных системах нового поколения* по причине однонаправленности создаваемого им соединения.

В силу слабой развитости средств для обработки текстов на *SCs-коде* в качестве языка коммуникации систем был разработан строковый язык на базе языка *JSON Marrs T.Json aWPDIftW-2017bk — SC-JSON-код*. Его выбор объясняется гибкостью задания отношений между описываемыми им объектами.

Пункт 6.3.6.1. Спецификация Метаязыка представления сообщений между подсистемами ostis-платформы. SC-JSON-код

Подсистемы в рамках реализуемого программного варианта *специализированной ostis-платформы* общаются с помощью *языка внешнего представления знаний* ***SC-JSON-код***. Данный язык является строковым, то есть линейным, и простым в обработке, поскольку существует большое количество средств для обработки его надъязыка *JSON*.

SC-JSON-код
:= [Semantic JSON-code]
:= [Semantic JavaScript Object Notation code]
:= [Метаязык описания представления сообщений между подсистемами ostis-платформы]
⇒ *часто используемый sc-идентификатор**:
[sc-json-текст]
:= [Предлагаемый нами язык взаимодействия в распределенном коллективе ostis-систем]
∈ *имя нарицательное*
∈ *абстрактный язык*
⊂ *SC-код*
⊂ *JSON*

Пункт 6.3.6.2. Синтаксис SC-JSON-кода

Синтаксис SC-JSON-кода задается: (1) *Алфавитом SC-JSON-кода^*, (2) Грамматикой SC-JSON-кода. В Алфавите SC-JSON-кода^выделяется базовая синтаксическая классификация его элементов.

Синтаксическая классификация элементов SC-JSON-кода
⊃=
{

SC-JSON-код
⇐ *семейство подмножеств**:
sc-json-предложение
⊂ *json-список json-пар*
⇐ *семейство подмножеств**:
*sc-json-пара**
⇐ *декартово произведение**:
⟨• *sc-json-строка*
• *sc-json-объект*
⟩
⇒ *разбиение**:
{• *команда на SC-JSON-коде*
• *ответ на команду на SC-JSON-коде*
}

sc-json-объект
⇒ *разбиение**:
{• *sc-json-список*
• *sc-json-пара*
• *sc-json-литерал*
⇒ *разбиение**:
{• *sc-json-строка*
• *sc-json-число*
}
}
}

Алфавит SC-JSON-кода^ представляет собой множество всех возможных символов в SC-JSON-коде. Поскольку *SC-JSON-код* является линейным строковым языком представления знаний, то его алфавит включает объединение алфавитов всех языков, тексты на которых могут представлять внешние идентификаторы и/или содержимое *файлов ostis-системы*, множество всех цифр и множество всех других специальных символов. Последовательности

знаков алфавита могут образовывать sc-json ключевые слова, *sc-json-пары*, *sc-json-предложения* из *sc-json-пар* и *sc-json-тексты* из *sc-json-предложений*. При этом конструкции на *SC-JSON-коде* строятся по следующим ***Синтаксическим правилам SC-JSON-кода***:

- Каждое правило *Грамматики SC-JSON-кода* описывает корректный с точки зрения *Синтаксиса SC-JSON-кода* порядок sc-json-объектов в sc-json-предложении. Совокупность правил *Грамматики SC-JSON-кода* описывает корректный с точки зрения *Синтаксиса SC-JSON-кода* порядок *sc-json-предложений* в *sc-json-тексте*. Каждое sc-json-предложение является *sc-json-списком*, состоящим из *sc-json-пар* и представляет собой команду или ответ на эту команду.
- Каждая *команда (ответ на команду) на SC-JSON-коде* состоит из заголовка, включающего *sc-json-пары* описания самой команды (ответа на команду), и сообщения, различного для каждого класса команд (ответов на команды). Сообщение *команды (ответа на команду) на SC-JSON-коде* обычно представляет собой список *sc-json-объектов* и может не ограничиваться по мощности.
- Каждая *sc-json-пара* состоит из двух элементов: ключевого слова и некоторого другого *sc-json-объекта*, ассоциируемого с этим ключевым словом. Набор ключевых слов в *sc-json-парах* определяется конкретным классом *команд (ответов на команды) на SC-JSON-коде*. *Sc-json-пара* начинается знаком открывающейся фигурной скобки "{" и заканчивается знаком закрывающейся фигурной скобки "}". Ключевое слово и sc-json-объект, ассоциируемый с ним, разделяются при помощи знака двоеточия ":".
- *sc-json-строки*, записанные в sc-json-текстах, начинаются и заканчиваются знаком двух ковычек “.
- *sc-json-списки*, состоящие не из *sc-json-пар*, начинаются знаком открывающейся квадратной скобки "[" и заканчиваются знаком закрывающейся квадратной скобки "]". *sc-json-объекты* в *sc-json-списках* разделяются запятыми ",".

Пункт 6.3.6.3. Грамматика SC-JSON-кода

Грамматика SC-JSON-кода представляет собой множество всех возможных правил, используемых при построении команд и ответов на них на *SC-JSON-коде*. Каждой команде *SC-JSON-кода* однозначно соответствует правило грамматики *SC-JSON-кода*. Правила *Грамматики SC-JSON-кода* позволяют правильно представлять команды на *SC-JSON-коде*. Каждое правило грамматики *SC-JSON-кода* представляется в виде правила на *Языке описания грамматик ANTLR* и его интерпретации на *естественном языке*.

Грамматика SC-JSON-кода

∋ *ключевой sc-элемент′*:
Правило, задающее синтаксис команд на SC-JSON-коде
⇐ *синтаксическое правило**:
команда на SC-JSON-коде

∋ *ключевой sc-элемент′*:
Правило, задающее синтаксис ответов на команды на SC-JSON-коде
⇐ *синтаксическое правило**:
ответ на команду на SC-JSON-коде

∋ *Правило, задающее синтаксис команды создания элементов в sc-памяти*
⇐ *синтаксическое правило**:
команда создания элементов в sc-памяти

∋ *Правило, задающее синтаксис ответа на команду создания элементов в sc-памяти*
⇐ *синтаксическое правило**:
ответ на команду создания элементов в sc-памяти

Правило, задающее синтаксис команд на SC-JSON-коде, представлено на рисунке *Рисунок. Описание Правила, задающего синтаксис команд на SC-JSON-коде*. Класс *команд на SC-JSON-коде* включает *команду создания элементов в sc-памяти*, *команду получения классов элементов sc-памяти*, *команду удаления элементов из sc-памяти*, *команду обработки ключевых sc-элементов*, *команду обработки содержимого элементов sc-памяти, соответствующих файлам ostis-системы*, *команду поиска конструкций в sc-памяти, изоморфных заданному программному объекту графа-образца*, *команду создания конструкции в sc-памяти, изоморфной заданному программному объекту графа-образца*, и *команду обработки sc-событий*. В *команду на SC-JSON-коде* включаются идентификатор этой команды, тип и сообщение.

Рисунок. Описание Правила, задающего синтаксис команд на SC-JSON-коде
=

```
sc_json_command
 : '{'
     '"id"' ':' NUMBER ','
     sc_json_command_type_and_payload
   '}'
 ;

sc_json_command_type_and_payload
 : sc_json_command_create_elements
 | sc_json_command_check_elements
 | sc_json_command_delete_elements
 | sc_json_command_handle_keynodes
 | sc_json_command_handle_link_contents
 | sc_json_command_search_template
 | sc_json_command_generate_template
 | sc_json_command_handle_events
 ;
```

Правило, задающее синтаксис ответа на команду на SC-JSON-коде описывает синтаксис ответов на команды, описываемых предыдущим правилом. Класс *ответов на команды на SC-JSON-коде* включает *ответ на команду создания элементов в sc-памяти*, *ответ на команду получения классов элементов в sc-памяти*, *ответ на команду удаления элементов из sc-памяти*, *ответ на команду обработки ключевых элементов в sc-памяти*, *ответ на команду обработки содержимого элементов sc-памяти, соответствующих файлам ostis-системы*, *ответ на команду поиска конструкций в sc-памяти, изоморфных заданному программному объекту графа-образца*, *ответ на команду создания конструкции в sc-памяти, изоморфной заданному программному объекту графа-образца*, и *ответ на команду обработки sc-событий*. В *ответ на команду на SC-JSON-коде* включаются идентификатор соответствующей команды, статус обработки ответа и ответное сообщение *Рисунок. Описание Правила, задающего синтаксис ответов на команды SC-JSON-кода на языке ANTLR*.

Рисунок. Описание Правила, задающего синтаксис ответов на команды SC-JSON-кода на языке ANTLR
=

```
sc_json_command_answer
 : '{'
     '"id"' ':' NUMBER ','
     '"status"' ':' BOOL ','
     sc_json_command_answer_payload
   '}'
 ;

sc_json_command_answer_payload
 : sc_json_command_answer_create_elements
 | sc_json_command_answer_check_elements
 | sc_json_command_answer_delete_elements
 | sc_json_command_answer_handle_keynodes
 | sc_json_command_answer_handle_link_contents
 | sc_json_command_answer_search_template
 | sc_json_command_answer_generate_template
 | sc_json_command_answer_handle_events
 ;
```

В сообщении ***команды создания sc-элементов*** представляется список описаний создаваемых элементов в sc-памяти. Такими элементами могут быть элемент sc-памяти, соответствующий sc-узлу, sc-коннектору или файлу ostis-системы. Класс элемента указывается в паре с ключевым словом "el": для элемента sc-памяти, соответствующего sc-узлу, sc-json-класс элемента представляется как "node", для элемента sc-памяти, соответствующего sc-коннектору — "edge", для элемента sc-памяти, соответствующего файлу ostis-системы — "link". Метки классов sc-элементов уточняются в соответствующих им описаниях в сообщении команды в паре с ключевым словом "type". Если создаваемым элементом является элемент sc-памяти, соответствующего файлу ostis-системы, то дополнительно указывается содержимое этого файла ostis-системы в паре с ключевым словом "content", если создаваемым элементом является элемент sc-памяти, соответствующего sc-коннектору, то указываются описания элементов, из которых они выходят, и элементов, в которые они входят. Описание таких элементов состоят из двух пар:

первая пара указывает на способ ассоциации с элементом и представляется как "addr" или "idtf" или "ref" в паре с ключевым словом "type", вторая пара — то, по чему происходит ассоциация с этим элементом: хэш его sc-адреса, локальному идентификатору или номеру в массиве создаваемых элементов — в паре с ключевым словом "value"
Рисунок. Описание Правила, задающего синтаксис команды создания элементов в sc-памяти на языке ANTLR.

Рисунок. Описание Правила, задающего синтаксис команды создания элементов в sc-памяти на языке ANTLR

=

```
sc_json_command_create_elements
 : '"type"' ':' '"create_elements"' ','
  '"payload"' ':'
  '['(
     '{'
        '"el"' ':' '"node"' ','
        '"type"' ':' SC_NODE_TYPE ','
     '}' ','
     |
     '{'
        '"el"' ':' '"link"' ','
        '"type"' ':' SC_LINK_TYPE ','
        '"content"' ':' NUMBER_CONTENT
                      | STRING_CONTENT
     '}' ','
     |
     '{'
        '"el"' ':' '"edge"' ','
        '"type"' ':' SC_EDGE_TYPE ','
        '"src"' ':' (
        '{'
           '"type"' ':' '"ref"' ','
           '"value"' ':' NUMBER
        '}' ','
        |
        '{'
           '"type"' ':' '"addr"' ','
           '"value"' ':' SC_ADDR_HASH
        '}' ','
        |
        '{'
           '"type"' ':' '"idtf"' ','
           '"value"' ':' SC_NODE_IDTF
        '}' ','
        )
        '"trg"' ':' (
        '{'
           '"type"' ':' '"ref"' ','
           '"value"' ':' NUMBER
        '}' ','
        |
        '{'
           '"type"' ':' '"addr"' ','
           '"value"' ':' SC_ADDR_HASH
        '}' ','
        |
        '{'
           '"type"' ':' '"idtf"' ','
           '"value"' ':' SC_NODE_IDTF
        '}' ','
        )
     '}' ','
     |
     scs_text ','
  )*']' ','
 ;
```

Сообщением ***ответа на команду создания элементов в sc-памяти*** является список хэшей sc-адресов созданных элементов в sc-памяти, соответствующих описаниям *команды создания элементы создания в sc-памяти* со статусом 1, в случае успешной обработки команды *Рисунок. Описание Правила, задающего синтаксис ответа на команду создания элементов в sc-памяти на языке ANTLR*.

Рисунок. Описание Правила, задающего синтаксис ответа на команду создания элементов в sc-памяти на языке ANTLR

=

```
sc_json_command_answer_create_elements
 : '"payload"' ':'
  '['
     (SC_ADDR_HASH ',')*
  ']' ','
 ;
```

Множество *команд на SC-JSON-коде* легко расширяемо засчет гибкости и функциональности языка JSON. Множество *ответов на команды на SC-JSON-коде* также легко расширяемо вместе с расширением *команд на SC-JSON-коде*.

Детальное описание синтаксиса команд и ответов на эти команды, а также их примеры можно найти в Стандарте OSTIS (см. *Голенков В.В..СтандОТОП-2021кн*).

Пункт 6.3.6.4. Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы, представляет собой интерпретатор команд и ответов на них *SC-JSON-кода* в программное представление конструкций в памяти ostis-платформы, использующий *Библиотеку программных компонентов для обработки json-текстов JSON for Modern C++* и *Библиотеку кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++*. *Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы* обеспечивается комплексным тестовым покрытием посредством программных фреймворков *Google Tests* и *Google Benchmark Tests*. *Библиотека программных компонентов для обработки json-текстов JSON for Modern C++* имеет богатый, удобный и быстродействующий функционал, необходимый для реализации подобных компонентов ostis-систем, а *Библиотека кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++* позволяет элегантно проектировать серверные системы без использовании избыточных зависимостей и решений. Настройка программного компонента осуществляется с помощью *Программного компонента настройки программных компонентов ostis-систем* и скриптов языков CMake и Bash.

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы

:= [Реализация системы, работающей по принципам Websocket и предоставляющей параллельно-асинхронный многоклиентский доступ к sc-памяти платформы интерпретации sc-моделей при помощи SC-JSON-кода]

:= [sc-json-сервер]

⇒ *часто используемый sc-идентификатор**:
[sc-сервер]

:= [sc-server]

∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*

∈ *атомарный многократно используемый компонент ostis-систем*

∈ *зависимый многократно используемый компонент ostis-систем*

⇒ *используемый язык представления методов**:
- *C*
- *C++*

⇒ *используемый язык**:
- *SC-JSON-код*

⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-machine/sc-tools/sc-server]

⇒ *зависимости компонента**:
{• *Библиотека программных компонентов для обработки json-текстов JSON for Modern C++*
• *Библиотека кросс-платформенных программных компонентов для реализации серверных приложений на основе Websocket WebSocket++*
• *Программный компонент настройки программных компонентов ostis-систем версия*
• *Реализация sc-памяти*
}

Стоит отметить, что текущая *Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы*, не является первой для текущего *Программного варианта реализации ostis-платформы* и заменяет предыдущую ее реализацию, написанную на языке *Python*. Причина такой замены состоит в следующем:

- Предыдущая *Реализация Серверной системы на основе Websocket, обеспечивающей доступ к sc-памяти при помощи команд SC-JSON-кода*, реализованная на языке программирования Python, зависит от библиотеки Boost Python, предоставляемой сообществом по развитию и коллаборации языков C++ и Python. Дело в том, что такое решение требует поддержки механизма интерпретации программного кода на Python на язык C++, что является избыточным и необоснованным, поскольку большая часть программного кода *Программного варианта реализации ostis-платформы* реализована на языках C и C++. Новая реализация описываемой программной системы позволяет избавиться от использования емких и ресурсозатратных библиотек (например, boost-python-lib, llvm) и языка Python;
- При реализации распределенных подсистем важную роль играет скорость обработки знаний, то есть возможность быстро и своевременно отвечать на запросы пользователя. Качество доступа к sc-памяти посредством реализованной *Подсистемы взаимодействия с sc-памятью на основе языка JSON* должно быть соизмеримо с качеством доступа к sc-памяти при помощи специализированного *программного интерфейса*, реализованного на том же языке программирования, что и сама система. Новая реализация позволяет повысить скорость обработки запросов *Подсистемой взаимодействия с sc-памятью на основе языка JSON*, в том числе и обработки знаний, не менее чем в 1,5 раза по сравнению с предыдущим вариантом реализации этой подсистемы.

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы, обладает следующими общими характеристиками:

- Серверная подсистема имеет такой же специализированный *программный интерфей*с, как и *Реализация sc-памяти*, однако взаимодействие с ней при помощи такого интерфейса осуществляется посредством сети. Это обеспечивает возможность взаимодействия клиентских систем, реализованных на разных языках программирования, с одной общей памятью.
- Данную подсистему можно рассматривать как некоторый интерпретатор внешнего языка представления знаний *SC-JSON-код*, на котором могут общаться *ostis-системы*, реализованные на базе *специализированной ostis-платформы*. Каждой команде и ответу на команду этого языка соответствует обработчик (потенциально и вовсе агент), который является частью этого интерпретатора. Сам язык внешнего представления знаний *SC-JSON-код* независим от реализации платформы и используется только как язык внешнего представления знаний, но может быть задействован при реализации других средств и *интерпретаторов sc-моделей ostis-систем*.
- Реализованный программный компонент предоставляет многопользовательский асинхронный доступ к sc-памяти. В ходе тестирования *sc-сервера* выяснилось, что его реализация позволяет обрабатывать запросы не менее чем 1000 клиентских систем. В связи с необходимостью обеспечения параллельного доступа к sc-памяти на уровне реализации программного компонента были добавлены блоки синхронизации. Например, в реализации можно заметить очередь команд на обработку системой — вне зависимости от количества клиентских систем и того, в каком количестве они отправляют команды на обработку, все команды могут становиться в очередь. Такое решение позволяет временно обойти проблемы взаимодействия блоков синхронизации на уровне *sc-памяти* при обработке разных типов команд над ней (поисковых, генеративных, деструктивных и так далее). При этом серверную систему невозможно отключить до тех пор, пока очередь команд имеет какие-нибудь необработанные команды. Также серверная система продолжает работать, если в списке идентификаторов клиентских систем остались неотключенные из них. Необходимость данных функций серверной подсистемы обуславливается необходимостью поддержки атомарности запросов, обрабатываемых системой.
- В процессе тестирования подсистемы были получена оценка ее скорости обработки команд и ответов. При нагрузочном тестировании использовалась тестовая клиентская система, написанная на *C++* и не имеющая функционала обработки текстов *SC-JSON-кода*. В результате тестирования было выяснено, что при отправке 1000 различных команд: *команд создания sc-элементов*, *команд обработки содержимого файлов ostis-системы* и *команд удаления sc-элементов* — время, потраченное на их обработку не превышало 0,2 секунды. При этом в отдельных случаях на обработку 1000 *команд создания sc-элементов* уходило не более 0,14 секунды, *команд удаления sc-элементов* — не более 0,07 секунды, *команд обработки содержимого файлов ostis-системы* — не более 0,27 секунды, *команд поиска sc-конструкций, изоморфных заданному графу-образцу* — не более 0,45 секунды.

Реализация Серверной системы на основе Websocket и JSON, обеспечивающей сетевой доступ к памяти ostis-платформы обеспечивает необходимый и достаточный *программный интерфейс* для взаимодействия с *sc-памятью*. В общем случае ее спецификация включает описание функциональных возможностей не только *Реализации Серверной системы на основе Websocket, обеспечивающей доступ к памяти ostis-платформы*, но и всех клиентских систем взаимодействующих с ней, поскольку зачастую эти клиентские системы включают *специализированный программный интерфейс*, схожий с интерфейсом самой серверной системы, но реализованный на другом *языке программирования*.

§ 6.3.7. Реализация интерпретатора sc-моделей пользовательских интерфейсов

⇒ *подраздел**:
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Пункт 6.3.7.2. Достоинства и недостатки текущего варианта Реализации интерпретатора sc-моделей пользовательских интерфейсов*

⇒ *ключевой знак**:
- *Реализация интерпретатора sc-моделей пользовательских интерфейсов*

⇒ *ключевое знание**:
- *Спецификация Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Принципы, лежащие в основе Реализации интерпретатора sc-моделей пользовательских интерфейсов*

В большинстве случаев разработка *пользовательского интерфейса* в современных системах отнимает большую часть времени, затрачиваемого на разработку всей системы. Однако эффективность использования *программной компьютерной системы* зависит от разрабатываемого *пользовательского интерфейса* (см. *Myers B.A..Surve oUIP-1992art*, *Главу 4.1. Общие принципы организации интерфейсов ostis-систем*).

Наряду с *Реализацией sc-памяти* важной частью *Программного варианта реализации ostis-платформы* является ***Реализация интерпретатора sc-моделей пользовательских интерфейсов***, которая предоставляет базовые средства просмотра и редактирования базы знаний пользователем, средства для навигации по *базе знаний* (задания *вопросов* к *базе знаний*) и может дополняться новыми компонентами в зависимости от задач, решаемых каждой конкретной *ostis-системой*.

Реализация интерпретатора sc-моделей пользовательских интерфейсов
:= [Предлагаемый нами интерпретатор для интерпретации sc-моделей пользовательских интерфейсов ostis-систем]
∈ *многократно используемый компонент ostis-систем, хранящийся в виде файлов исходных текстов*
∈ *неатомарный многократно используемый компонент ostis-систем*
∈ *зависимый многократно используемый компонент ostis-систем*
⇒ *используемый язык представления методов**:
- *JavaScript*
- *TypeScript*
- *Python*
- *HTML*
- *CSS*

⇒ *адрес компонента**:
[https://github.com/ostis-ai/sc-web]
⇒ *зависимости компонента**:
{• *Библиотека стандартных интерфейсных компонентов на языке программирования JavaScript*
- *Библиотека для реализации серверных приложений на языке программирования Python Tornado*
- *Реализация клиентской системы на языке программирования TypeScript*
- *Реализация клиентской системы на языке программирования Python*

}

Важным принципом *Реализации интерпретатора sc-моделей пользовательских интерфейсов* является простота и однотипность подключения любых компонентов *пользовательского интерфейса* (*редакторов*, *визуализаторов*, *переключателей*, *команд меню* и так далее). Для этого реализуется программная прослойка Sandbox, в рамках которой реализуются низкоуровневые операции взаимодействия с серверной частью и которая обеспечивает более удобный *программный интерфейс* для разработчиков компонентов. Текущий вариант *Реализации интерпретатора sc-моделей пользовательских интерфейсов* является открытым и доступен на *SoftwIoWOUI-el*.

Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов

⇒ *ключевой знак**:
- *Панель меню команд пользовательского интерфейса*
- *Компонент переключения языка идентификации отображаемых sc-элементов*
- *Компонент переключения внешнего языка визуализации знаний*
- *Поле поиска sc-элементов по идентификатору*
- *Панель отображения диалога пользователя с ostis-системой*
- *Панель визуализации и редактирования знаний*
- *Визуализатор sc.n-текстов*
- *Визуализатор и редактор sc.g-текстов*

Реализация интерпретатора sc-моделей пользовательских интерфейсов
⇒ *декомпозиция программной системы**:
{• *Панель меню команд пользовательского интерфейса*
• *Компонент переключения языка идентификации отображаемых sc-элементов*
• *Компонент переключения внешнего языка визуализации знаний*
• *Поле поиска sc-элементов по идентификатору*
• *Панель отображения диалога пользователя с ostis-системой*
• *Панель визуализации и редактирования знаний*
⇒ *декомпозиция программной системы**:
{• *Визуализатор sc.n-текстов*
• *Визуализатор и редактор sc.g-текстов*
}
}

Компонент переключения языка идентификации отображаемых sc-элементов является изображением множества имеющихся в системе *естественных языков*. Взаимодействие пользователя с данным компонентом переключает *пользовательский интерфейс* в режим общения с конкретным пользователем с использованием *основных sc-идентификаторов*, принадлежащих данному *естественному языку*. Это значит, что при изображении *sc-идентификаторов sc-элементов* на каком-либо языке, например, *SCg-коде* или *SCn-коде* будут использоваться *основные sc-идентификаторы*, принадлежащие данному *естественному языку*. Это касается как *sc-элементов*, отображаемых в рамках *Панели визуализации и редактирования знаний*, так и любых других sc-элементов, например, классов команд и даже самих *естественных языков*, изображаемых в рамках самого *Компонента переключения языка идентификации отображаемых sc-элементов*.

Компонент переключения внешнего языка визуализации знаний служит для переключения языка визуализации знаний в текущем окне, отображаемом на ***Панели визуализации и редактирования знаний***. В текущей реализации в качестве таких языков по умолчанию поддерживаются *SCg-код* и *SCn-код*, а также любые другие языки, входящие во множество *внешних языков визуализации SC-кода*.

Поле поиска sc-элементов по идентификатору позволяет осуществлять поиск sc-идентификаторов, содержащих подстроку, введенную в данное поле (с учетом регистра). В результате поиска отображается список *sc-идентификаторов*, содержащих указанную подстроку, при взаимодействии с которыми осуществляется автоматическое задание вопроса “Что это такое?”, аргументом которого является либо сам *sc-элемент*, имеющий данный *sc-идентификатор* (в случае, если указанный *sc-идентификатор* является основным или системным, и, таким образом, указанный *sc-элемент* может быть определен однозначно), либо внутренний файл *ostis-системы*, являющийся *sc-идентификатором* (в случае, если данный sc-идентификатор является неосновным).

Панель отображения диалога пользователя с ostis-системой отображает упорядоченный по времени список *sc-элементов*, являющихся знаками действий, которые инициировал *пользователь* в рамках диалога с *ostis-системой* путем взаимодействия с изображениями соответствующих классов *команд* (то есть, если действие было инициировано другим способом, например, путем его явного инициирования через создание дуги принадлежности множеству *инициированных действий* в *sc.g-редакторе*, то на данной панели оно отображено не будет). При взаимодействии *пользователя* с любым из изображенных знаков действий на *Панели визуализации и редактирования знаний* отображается окно, содержащее результат выполнения данного *действия* на том языке визуализации, на котором он был отображен, когда *пользователь* просматривал его в последний (предыдущий) раз. Таким образом, в текущей реализации данная *панель* может работать только в том случае, если инициированное *пользователем* действие предполагает явно представленный в памяти результат данного действия. В свою очередь, из этого следует, что в настоящее время данная панель, как и в целом *Реализация интерпретатора sc-моделей пользовательских интерфейсов*, позволяет работать с системой только в режиме диалога “вопрос-ответ”.

Панель визуализации и редактирования знаний отображает окна, содержащие sc-текст, представленный на некотором языке из множества *внешних языков визуализации SC-кода* и, как правило, являющийся результатом некоторого действия, инициированного *пользователем*. Если соответствующий визуализатор поддерживает возможность редактирования текстов соответствующего *естественного языка*, то он одновременно является также и редактором. При необходимости *пользовательский интерфейс* каждой конкретной *ostis-системы* может быть дополнен *визуализаторами* и *редакторами* различных *внешних языков*, которые в текущей версии *Реализации интерпретатора sc-моделей пользовательских интерфейсов* будут также располагаться на *Панели визуализации и редактирования знаний*. По умолчанию доступны две панели визуализации и редактирования: *Визуализатор sc.n-текстов* и *Визулизатор и редактор sc.g-текстов*.

Панель меню команд пользовательского интерфейса содержит изображения классов команд (как атомарных, так и неатомарных), имеющихся на данный момент в базе знаний и входящих в декомпозицию *Главного меню пользовательского интерфейса* (имеется в виду полная декомпозиция, которая в общем случае может включать несколько уровней *неатомарных классов команд*). Взаимодействие с изображением неатомарного класса команд инициирует команду изображения *классов команд*, входящих в декомпозицию данного *неатомарного класса команд*. Взаимодействие с изображением *атомарного класса команд* инициирует генерацию команды данного класса с ранее выбранными аргументами на основе соответствующей *обобщенной формулировки класса команд* (шаблона класса команд).

Семантические модели описанных компонентов *пользовательского интерфейса* более подробно представлены в работах *Корончик Д.Н.СеманТКПП-2011cm*, *Корончик Д.Н.СеманММПИ*, *Корончик Д.Н.УнифиСМПИ-2013cm*, *Корончик Д.Н.ПользИИМП-2014cm* и *Sadouski M.SemanDoaAUI-2022art*.

Пункт 6.3.7.2. Достоинства и недостатки текущего варианта Реализации интерпретатора sc-моделей пользовательских интерфейсов

Текущая *Реализация интерпретатора sc-моделей пользовательских интерфейсов* имеет большое множество недостатков, а именно:

- Идея платформенной независимости *пользовательского интерфейса* (построения *sc-модели пользовательского интерфейса*) реализована не в полной мере. Полностью описать *sc-модель пользовательского интерфейса* (включая точное размещение, размеры, дизайн компонентов, их поведение и другое) в настоящее время скорее всего окажется затруднительно из-за ограничений производительности, однако вполне возможно реализовать возможность задания вопросов ко всем компонентам интерфейса, изменить их расположение и так далее, однако эти возможности нельзя реализовать в текущем *Программном варианте реализации ostis-платформы*.
- Кроме того, часть интерфейса фактически работает напрямую с sc-памятью с использованием технологии *WebSocket*, а часть — через прослойку на базе *Библиотеки tornado* для *языка программирования Python*, что приводит к дополнительным зависимостям от сторонних библиотек. В последнее время развития текущего *Программного варианта реализации ostis-платформы* данная проблема в большей мере была решена, однако все еще остались компоненты, реализуемые на *Python*.
- Часть компонентов (например, *поле поиска по идентификатору*) реализована сторонними средствами и практически никак не связана с sc-памятью. Это затрудняет развитие *Реализации интерпретатора sc-моделей пользовательских интерфейсов* ориентирована только на ведение диалога с *пользователем* (в стиле вопрос пользователя — ответ системы). Не поддерживаются такие очевидно необходимые ситуации, как выполнение команды, не предполагающей ответа; возникновение ошибки или отсутствие ответа; необходимость задания вопроса системой пользователю и так далее.
- Ограничена возможность взаимодействия пользователя с системой без использования специальных *элементов управления*. Например, можно задать вопрос системе, нарисовав его в *SCg-коде*, но ответ пользователь не увидит, хотя в памяти он будет сформирован соответствующим агентом. Большая часть технологий, использованных при реализации платформы, к настоящему моменту устарела, что затрудняет развитие *ostis-платформы*.
- Не реализован механизм наследования при добавлении новых внешних языков. Например, добавление нового языка даже очень близкого к *SCg-коду* требует физического копирования кода компонента и внесение соответствующих изменений, при этом получаются два никак не связанных между собой компонента, которые начинают развиваться независимо друг от друга.
- Слабый уровень задокументированности текущей *Реализации интерпретатора sc-моделей пользовательских интерфейсов*. Представленная текущая спецификация пока только описывает ключевые моменты *Реализации интерпретатора sc-моделей пользовательских интерфейсов*, но не раскрывает их.

На основе описанных недостатков к будущей реализации предъявляются следующие требования:

- Унифицировать принципы взаимодействия всех компонентов интерфейса с *Реализации sc-памяти в ostis-платформе*, независимо от того, к какому типу относится компонент. Например, список команд меню должен формироваться через тот же механизм, что и ответ на *запрос пользователя*, и *команда редактирования*, сформированная пользователем, и *команда добавления нового фрагмента в базу знаний* и так далее. Необходимо совершенствовать способы использования интерфейса для удобного и комфортного пользования.
- Унифицировать принципы взаимодействия *пользователей* с системой независимо от способа взаимодействия и *внешнего языка*. Например, должна быть возможность задания вопросов и выполнения других команд прямо через SCg/SCn интерфейс. При этом необходимо учитывать принципы редактирования базы знаний, чтобы пользователь не мог под видом задания вопроса внести новую информацию в *согласованную часть базы знаний*.
- Унифицировать принципы обработки *событий*, происходящих при взаимодействии пользователя с компонентами интерфейса — поведение кнопок и других интерактивных компонентов должно задаваться не статически сторонними средствами, а реализовываться в виде агента, который, тем не менее, может быть реализован произвольным образом (не обязательно на *платформенно-независимом* уровне). Любое *действие*, совершаемое *пользователем*, на логическом уровне должно трактоваться и обрабатываться как инициирование *агента*.
- Обеспечить возможность выполнять команды (в частности, задавать вопросы) с произвольным количеством аргументов, в том числе — без аргументов.
- Обеспечить возможность отображения ответа на вопрос по частям, если ответ очень большой и для отображения требуется много времени.
- Каждый отображаемый компонент интерфейса должен трактоваться как изображение некоторого *sc-узла*, описанного в *базе знаний*. Таким образом, пользователь должен иметь возможность задания произвольных вопросов к любым компонентам интерфейса.
- Максимально упростить и задокументировать механизм добавления новых компонентов.
- Обеспечить возможность добавления новых компонентов на основе имеющихся без создания независимых копий. Например, должна быть возможность создать компонент для языка, расширяющего *SCg-код* новыми примитивами, переопределять принципы размещения *sc-текстов* и так далее.
- Свести к минимуму зависимость от сторонних библиотек.
- Свести к минимуму использование протокола *HTTP* (начальная загрузка общей структуры интерфейса), обеспечить возможность равноправного двустороннего взаимодействия серверной и клиентской части.

Очевидно, что реализация большинства из приведенных требований связана не только с собственно вариантом реализации ostis-платформы, но и требует развития теории *sc-моделей пользовательских интерфейсов* и уточнения в рамках нее общих принципов организации *пользовательских интерфейсов ostis-систем*.

Заключение к Главе 6.3.

В дальнейшем развитии *Программного варианта реализации ostis-платформы* важным и правильным будет:
- максимально детализировать спецификацию компонентов проектируемой *ostis-платформы*, в том числе используемых языков внешнего и внутреннего представления знаний, и четко стратифицировать иерархию классов и отношений, используемых при описании компонентов ostis-платформ;
- устранить и учесть недостатки при реализации новых компонентов в проектируемой ostis-платформе, указать возможные варианты их реализации;
- свести зависимость компонентов ostis-платформы от ее реализации к минимуму, то есть, по возможности, реализовать их на *Языке SCP* (например, *интерпретатор sc-моделей пользовательских интерфейсов ostis-систем*);
- оценить качество проектируемой системы и ее компонентов в целом.

Кратко перечислим основные положения Главы 6.3.:
- Текущий *Программный вариант реализации ostis-платформы* является кроссплатформенным, что позволяет:
  - вести разработку и поддерживать состояние ее компонентов вне зависимости от реализации платформ, на которых используются средства их проектирования и разработки,
  - использовать ее для решения задач на любых доступных устройствах.
- Текущий *Программный вариант реализации ostis-платформы* является многопользовательским, то есть позволяет обрабатывать несколько действий одновременно.
- Текущая *Реализация памяти ostis-платформы* является достаточно полной для того, чтобы:
  - взаимно однозначно интерпретировать sc-модели ostis-систем, в том числе внешние информационные конструкции, не принадлежащие SC-коду;
  - разрабатывать платформенно-зависимые компоненты, требующие доступа к sc-памяти (например, *Программный интерфейс Реализации sc-памяти в данной ostis-платформе*).

- Текущий *Программный вариант реализации ostis-платформы* является специализированным, то есть позволяет создавать только платформенно-зависимые ostis-системы.
- На базе текущего *Программного варианта реализации ostis-платформы* реализуются *интерпретатор sc-моделей пользовательских интерфейсов ostis-систем*, *интерпретатор логических моделей решения задач в ostis-системах*, а также *менеджер многократно используемых компонентов ostis-систем*.

Часть 7.
Экосистема OSTIS

:= [Часть 7. Экосистема интеллектуальных компьютерных систем нового поколения и их пользователей]
:= [Часть 7. Экосистема интероперабельных семантически совместимых интеллектуальных компьютерных систем, имеющих высокий уровень самообучаемости и обеспечивающих комплексную автоматизацию всевозможных видов и областей человеческой деятельности]
:= [Часть 7. Экосистема ostis-систем]

⇒ *подраздел**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 7.2. Метасистема OSTIS*
- *Глава 7.3. Структура Экосистемы OSTIS*
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*

Глава 7.1.

Структура и проблемы организации и комплексной автоматизации человеческой деятельности

⇒ *автор**:
- *Голенков В. В.*
- *Гулякина Н. А.*

⇒ *аннотация**:
[В главе рассмотрены принципы автоматизации различных областей *человеческой деятельности* с использованием *интеллектуальных компьютерных систем нового поколения*. Предлагается онтология различных видов деятельности и соответствующих технологий. Детализация указанных принципов осуществляется на примере человеческой деятельности в области *Искусственного интеллекта*.]

⇒ *подраздел**:
- *§ 7.1.1. Структура и текущее состояние деятельности в области Искусственного интеллекта*
- *§ 7.1.2. Проблемы и перспективы комплексной автоматизации всевозможных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения*

⇒ *библиографическая ссылка**:
- *Янковская А.Е..ГибриИСЭДН-2017ст*
- *Палагин А.В..ПроблТиРИ-2013ст*
- *Тарасов В.Б.оМногоСкИОФ-2002кн*
- *Баринов И.И..ФормиСРКпИИ-2021ст*
- *Iliadis A.T.Tower oBPMDM-2019art*
- *Yaghoobirafi K..aAppro fSIiADIS-2022art*
- *Ouksel A.M..SemanIiGIS-1999art*
- *Lanzenberger M..MakinOTKIitSW-2008art*
- *Neiva F.W..TowarPItSC-2016art*
- *Pohl J.Inter atNfISaHP-2004art*
- *Waters J..GlobaIUSS-2009art*
- *Golenkov V.V..ArtifISIaK-2020art*
- *ITAPKIT-2012el*
- *Волков А.И..ПрофеСвОИТ-2015ст*
- *Серенков П.С..КонцеИСкБЗ-2004ст*

Введение в Главу 7.1.

Ключевая проблема современного уровня комплексной автоматизации *человеческой деятельности* заключается в следующем. В настоящее время осуществляется либо полная автоматизация некоторых классов действий, инициируемых соответствующими командами, либо частичная автоматизация некоторых видов *человеческой деятельности*, в рамках которой человек осуществляет управление соответствующими средствами автоматизации. При этом автоматизация решения комплексных задач, которые сводятся к нескольким частично автоматизированным подзадачам, требует "ручного" (неавтоматизированного) управления одновременно несколькими средствами автоматизации.

Принципы, лежащие в основе перехода к более высокому уровню автоматизации *человеческой деятельности* сводятся к следующему. Все средства автоматизации (все сервисы), управляемые в настоящее время людьми, переходят "под управление" ***интероперабельными интеллектуальными компьютерными системами***, которые способны эффективно взаимодействовать между собой и, соответственно, способны полностью автоматизировать решение комплексных задач, требующих использования нескольких средств автоматизации (сервисов) *Yaghoobirafi K..aAppro fSIiADIS-2022art; Ouksel A.M..SemanIiGIS-1999art; Lanzenberger M..MakinOTKIitSW-2008art; Neiva F.W..TowarPItSC-2016art; Pohl J.Inter atNfISaHP-2004art; Waters J..GlobaIUSS-2009art*.

В рамках данной главы рассмотрим структуру текущего состояния и проблемы развития *Человеческой деятельности в области Искусственного интеллекта*. Далее обобщим принципы организации и автоматизации *Человеческой деятельности в области Искусственного интеллекта* на все многообразие видов и областей *человеческой деятельности*.

§ 7.1.1. Структура и текущее состояние деятельности в области Искусственного интеллекта

⇒ *подраздел**:
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.2. Структура деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.3. Текущее состояние и современные проблемы научно-исследовательской деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.4. Текущее состояние унификации интеллектуальных компьютерных систем*
- *Пункт 7.1.1.5. Текущее состояние и современные проблемы развития технологий проектирования интеллектуальных компьютерных систем*
- *Пункт 7.1.1.6. Текущее состояние и современные проблемы развития технологий реализации спроектированных интеллектуальных компьютерных систем, а также их эксплуатации и сопровождения*
- *Пункт 7.1.1.7. Текущее состояние и современные проблемы прикладной инженерной деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.8. Текущее состояние и современные проблемы учебной деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.9. Текущее состояние и современные проблемы организационной деятельности в области Искусственного интеллекта*
- *Пункт 7.1.1.10. Ключевые задачи и методологические проблемы современного этапа развития Искусственного интеллекта*
- *Пункт 7.1.1.11. Комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения*

Введение в § 7.1.1.

В предыдущих главах мы уточнили:
- архитектуру *интеллектуальных компьютерных систем нового поколения*
- то, как осуществляется деятельность (функционирование) *интеллектуальных компьютерных систем нового поколения*
- то, как автоматизируется *прикладная инженерная человеческая деятельность по проектированию интеллектуальных компьютерных систем нового поколения* и поддержке всех последующих этапов их жизненного цикла.

Уточним то, как осуществляется и автоматизируется весь комплекс *человеческой деятельности в области Искусственного интеллекта*.

Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта

Рассмотрим необходимость перехода организации **человеческой деятельности *Искусственного интеллекта*** на принципиально новый уровень, обеспечивающий формирование рынка **семантически совместимых *интеллектуальных компьютерных систем нового поколения***, разрабатываемых на основе принципиально нового комплекса *семантически совместимых* ***технологий Искусственного интеллекта***.

Сейчас актуально исследовать не только *модели решения интеллектуальных задач* в интеллектуальных компьютерных системах различного вида, но также методологические проблемы текущего состояния *Искусственного интеллекта* в целом и пути решения этих проблем.

Анализ современного состояния работ в области *Искусственного интеллекта* показывает то, что указанная научно-техническая дисциплина находится в серьезном методологическом кризисе. Поэтому необходимо:
- Выявление основных причин возникновения указанного кризиса;
- Уточнение основных мер, направленных на его устранение.

Решение рассматриваемых кризисных проблем требует:
- Существенного фундаментального общесистемного переосмысления всего того, *что* мы делаем в области *Искусственного интеллекта* и *как* мы это делаем, то есть требует уточнения характеристик *интеллектуальных*

компьютерных систем, уточнения понятия сообщества, состоящего из *интеллектуальных компьютерных систем* и взаимодействующих с ними пользователей, уточнения требований, предъявляемых к *интеллектуальным компьютерным системам*, а также уточнения методик и средств их создания и использования.

- Осознания того, что *Кибернетика, Информатика* и *Искусственный интеллект* — это общая фундаментальная наука, требующая комплексного подхода к построению общих формальных моделей систем, основанных на обработке информации (*кибернетических систем*), путем *конвергенции* и *интеграции* формальных моделей различных компонентов этих систем (см. *Янковская А.Е..ГибриИСЭДН-2017ст*, *Палагин А.В..ПроблТиРИ-2013ст*). Таким образом, современный этап развития *Искусственного интеллекта* — это переход от накопленного к текущему моменту многообразия моделей решения различного вида задач к преобразованию этого многообразия в стройную систему *семантически совместимых* моделей;
- Осознания того, что сейчас требуется не расширять многообразие точек зрения, а учиться их согласовывать, обеспечивать их *семантическую совместимость*, совершенствуя соответствующие методы.

Обсуждая современную проблематику *конвергенции* различных моделей в области *Искусственного интеллекта* и построения интегрированных *гибридных моделей*, уместно вспомнить «фантастический рассказ Д. А. Поспелова “Соприкосновение”, посвященный контакту различных миров. В нем главный герой популярно излагает свою теорию *концептуальных разломов* <...>. Эта теория напоминает историю долгого периода дифференциации наук, когда различные научные дисциплины развивались независимо, словно параллельные миры, лишь изредка соприкасаясь друг с другом, а отдельные ученые, получая все более узкую специализацию, мало что знали о достижениях даже своих "близких собратьев". К счастью, в последние годы все чаще и чаще возникают новые области контакта между отдельными дисциплинами, происходит взаимопроникновение идей, установление аналогий между полученными результатами и тенденциями развития. Во многом это объясняется появлением и широким внедрением во все сферы жизни общества передовых информационных и коммуникационных технологий <...>. Современные технологии опираются на достижения многих научно-технических дисциплин, среди которых на первый план выходят синтетические науки нового поколения — науки об искусственном».

⇐ *цитата**:
Тарасов В.Б.оМногоСкИОФ-2002кн/с.13

Анализируя современное состояние работ в области *Искусственного интеллекта (ИИ)*, следует констатировать то, что *концептуальный разлом* между различными направлениями *Искусственного интеллекта* является очевидным фактом. Это подтверждается следующей цитатой из книги В. Б. Тарасова *Тарасов В.Б.оМногоСкИОФ-2002кн* «вновь, как и на заре ИИ, актуальными становятся формирование единых методологических основ ИИ, разработка теоретических проблем создания интеллектуальных систем новых поколений, развитие нетрадиционных аппаратно-программных средств. Здесь большие перспективы связаны с использованием идей и принципов синергетики в ИИ. Сам термин "синергетика" происходит от слова "синергия", означающего совместное действие, сотрудничество. По мнению "отца синергетики" Г. Хакена, такое название вполне подходит для современной теории сложных самоорганизующихся систем по двум причинам: а) исследуются совместные действия многих элементов развивающейся системы; б) для отыскания общих принципов самоорганизации требуется объединение усилий представителей различных дисциплин».

⇐ *цитата**:
Тарасов В.Б.оМногоСкИОФ-2002кн/с.14

Для того, чтобы убедиться в наличии *концептуального разлома* между различными направлениями *Искусственного интеллекта*, достаточно просто перечислить основные направления работы конференций по тематике *Искусственного интеллекта*, обращая внимание на то, что многие из них развиваются независимо от других:

- синергетические модели самоорганизации интеллектуальных компьютерных систем;
- гибридные интеллектуальные компьютерные системы;
- коллаборативные интеллектуальные компьютерные системы;
- мягкие вычисления, интеллектуальные вычисления;
- моделирование не-факторов;
- неклассические, многозначные, модальные, псевдофизические, индуктивные, нечеткие логики и приближенные рассуждения, логические программы;
- нечеткие множества, отношения, графы, алгоритмы;
- функциональные программы, нечеткие алгоритмы, генетические алгоритмы, продукционные модели;
- нейросетевые модели;
- параллельные асинхронные модели децентрализованного решения задач;
- обработка сигналов;
- мультисенсорная конвергенция, сенсо-моторная координация;
- модели ситуационного управления.

Преодоление *концептуального разлома* между различными направлениями исследований в области *Искусственного интеллекта* — это, своего рода "прыжок" через "концептуальную пропасть", который требует особой концентрации усилий. Через пропасть нельзя перепрыгнуть двумя прыжками.

Если кратко охарактеризовать **текущее состояние** всего комплекса работ в области ***Искусственного интеллекта***, то это — **иллюзия благополучия**. Происходит активное локальное развитие самых различных направлений *Искусственного интеллекта* (*неклассические логики*, *формальные онтологии*, *искусственные нейронные сети*, *машинное обучение*, *мягкие вычисления*, *многоагентные системы* и так далее), но комплексного повышения уровня *интеллекта* современных *интеллектуальных компьютерных систем* не происходит. Для этого прежде всего требуется сближение и *интеграция* всех направлений *Искусственного интеллекта* и соответствующее построение ***Общей формальной теории интеллектуальных компьютерных систем***, а также превращение современного многообразия *инструментальных средств* (frameworks) разработки различных компонентов *интеллектуальных компьютерных систем* в единую ***Технологию комплексного проектирования и поддержки всего жизненного цикла интеллектуальных компьютерных систем***, гарантирующую совместимость всех разрабатываемых компонентов *интеллектуальных компьютерных систем*, а также совместимость самих *интеллектуальных компьютерных систем* как самостоятельных субъектов (агентов, акторов), взаимодействующих между собой в рамках комплексных систем автоматизации сложных видов коллективной *человеческой деятельности* (умных домов, умных больниц, умных школ, умных производственных предприятий, умных городов и так далее). Таким образом, эпиграфом текущего состояния работ в области *Искусственного интеллекта* является известное высказывание из Экклезиаста: “Время разбрасывать камни и время собирать камни — всему свое время”.

«К сожалению, в современных дискуссиях по теме ИИ (Искусственного интеллекта) научные споры часто подменяются завышенными ожиданиями от скорого внедрения ИИ и значительным сужением темы ИИ, которая оказалась сведена лишь к *машинному обучению* на основе *искусственных нейронных сетей*. <...> При этом за бортом Национальной стратегии пока остались *онтологии*, *базы знаний*, *методы рассуждений* и *принятия решений*, *методы синтеза и анализа сложных конструкций*, умные кибер-физические системы, *цифровые двойники*, *автономные системы*, системы анализа как "больших", так и "малых" данных. <...>

Признавая всю важность *машинного обучения* на базе *искусственных нейронных сетей*, научные и практические результаты мирового уровня следует искать на стыке разных дисциплин в ***конвергенции*** различных технологий ИИ и ***интеграции*** полипредметных *знаний*. В этой связи формализация *знаний* в виде *онтологий* и *баз знаний* в рамках *Semantic Web* рассматривается как одно из фундаментальных направлений для создания *Искусственного интеллекта*. Действительно, какой же может быть *интеллект* без использования *знаний* современных учебников, на основе чего ИИ будет понимать *контекст ситуации*, делать *выводы* и *принимать решения*? <...>

Еще одной ключевой сферой ИИ, не нашедшей отражения в Российской стратегии по ИИ, является *распределенное принятие решений*, которое все больше становится коллективным для стремительно развивающихся систем умного Интернета вещей и автономных систем управления, начиная с беспилотных автомобилей, самолетов, кораблей и так далее.

Компанией Гартнер 2020 год был объявлен годом "автономных вещей", которые по мнению компании уже прошли большую эволюцию от "цифровых" к "умным". Ожидается, что на следующем этапе автономные вещи, обладающие собственным *искусственным интеллектом*, "заговорят" друг с другом и в научную повестку войдут вопросы ***семантической интероперабельности*** систем *Искусственного интеллекта*, которые будут не только обмениваться данными, но и вести переговоры для согласования решений. Дорожная карта научных исследований по *Искусственному интеллекту* США в качестве ключевых выделяет такие направления, как *связность* систем *Искусственного интеллекта* (Integrated Intelligence) и их *осмысленное взаимодействие* (Meaningful Interaction), наряду с разными видами *самообучения* в системах (Self-Aware Learning).»

⇐ *цитата**:
Баринов И.И..ФормиСРКпИИ-2021ст/с. 264-265

Ключевой причиной методологических проблем современного состояния *Искусственного интеллекта* и серьезным вызовом для специалистов в этой области является проклятие ***Вавилонского столпотворения*** (см. *Iliadis A.T.Tower oBPMDM-2019art*), которое преследует нас на всех уровнях:

- на уровне внутренней организации *решения задач* в *интеллектуальных компьютерных системах;*
- на уровне взаимодействия *интеллектуальных компьютерных систем* как между собой, так и с пользователями;
- на уровне взаимодействия ученых, работающих в области *Искусственного интеллекта*, что препятствует созданию *Общей формальной теории и стандарта интеллектуальных компьютерных систем*, а также *Технологии комплексного проектирования и поддержки всего жизненного цикла интеллектуальных компьютерных систем*
- на уровне взаимодействия между учеными, инженерами, разрабатывающими прикладные *интеллектуальные компьютерные системы*, преподавателями ВУЗов, которые готовят специалистов в области *Искусственного интеллекта*, а также студентами, магистрантами и аспирантами.

Сложность разрабатываемых в настоящее время *интеллектуальных компьютерных систем* и технологий *Искусственного интеллекта* достигла такого уровня, что для их разработки требуются не просто большие творческие коллективы, но и существенное повышение квалификации и качества этих коллективов. Как известно, квалификация коллектива разработчиков определяется не только квалификацией его членов, но также эффективностью и

атмосферой их взаимодействия. Известно также, что качество любой технической системы является отражением качества того коллектива, который эту систему разработал. Может ли коллектив достаточно квалифицированных специалистов, многие из которых не обладают высоким уровнем *интероперабельности*, разработать интеллектуальную компьютерную систему с высоким уровнем *интероперабельности*, а тем более технологию комплексной поддержки всего жизненного цикла *интеллектуальных компьютерных систем* такого уровня? Очевидный ответ на этот вопрос и очевидная сложность создания работоспособных творческих коллективов указывают на основной вызов, адресованный специалистам в области *Искусственного интеллекта* в настоящее время. Таким образом, требования, предъявляемые к *интеллектуальным компьютерным системам нового поколения* и определяющие их способность к индивидуальному и коллективному решению комплексных сложных задач, должны предъявляться и к разработчикам этих систем, а также к разработчикам любых других сложных объектов, поскольку все сложные виды и области человеческой деятельности являются коллективными и творческими.

Создание быстро развивающегося рынка семантически совместимых *интеллектуальных компьютерных систем* — это основная цель, адресованная специалистам в области *Искусственного интеллекта*, требующая преодоления ***Вавилонского столпотворения*** во всех его проявлениях, формирования высокой культуры договороспособности и унифицированной, согласованной формы представления коллективно накапливаемых, совершенствуемых и используемых знаний. Ученые, работающие в области *Искусственного интеллекта*, должны обеспечить ***конвергенцию*** результатов различных направлений *Искусственного интеллекта* и построить *Общую формальную теорию интеллектуальных компьютерных систем*, а также *Комплексную технологию проектирования семантически совместимых интеллектуальных компьютерных систем,* включающую соответствующие стандарты *интеллектуальных компьютерных систем* и их компонентов. Инженеры, *разрабатывающие прикладные интеллектуальные компьютерные системы*, должны сотрудничать с учеными и участвовать в развитии *Комплексной технологии проектирования семантически совместимых интеллектуальных компьютерных систем*, и поддержки всех последующих этапов жизненного цикла этих систем.

Разобщенность различных направлений исследований в области *Искусственного интеллекта* является главным препятствием создания *Комплексной технологии проектирования семантически совместимых интеллектуальных компьютерных систем*, а также *Технологии комплексной поддержки* всех последующих этапов жизненного цикла *интеллектуальных компьютерных систем*.

Пункт 7.1.1.2. Структура деятельности в области Искусственного интеллекта

Для того, чтобы рассмотреть проблемы дальнейшего развития *деятельности* в области *Искусственного интеллекта* и, в частности, проблемы комплексной автоматизации этой *деятельности*, необходимо уточнить структуру указанной *деятельности*.

Человеческая деятельность в области *Искусственного интеллекта* направлена на исследование и создание *интеллектуальных компьютерных систем* различного вида и различного назначения. Объектами исследования в области *Искусственного интеллекта* являются:

- *индивидуальные интеллектуальные компьютерные системы* (в частности, когнитивные агенты);
- *многоагентные интеллектуальные компьютерные системы* (в частности, сообщества, состоящие из *индивидуальных интеллектуальных компьютерных систем*);
- человеко-машинные сообщества, состоящие из *интеллектуальных компьютерных систем* и их пользователей.

Основными целями человеческой деятельности в области *Искусственного интеллекта* являются:

- Построение формальной теории *интеллектуальных компьютерных систем* (искусственных *интеллектуальных систем*);
- Создание технологий (методик и средств), обеспечивающих *проектирование, реализацию, сопровождение и эксплуатацию интеллектуальных компьютерных систем*;
- Переход на принципиально новый уровень комплексной автоматизации всевозможных *видов человеческой деятельности*, который основан на массовом применение *интеллектуальных компьютерных систем* и который предполагает:
  - не только наличие *интеллектуальных компьютерных систем,* способных понимать друг друга и согласовывать свою деятельность,
  - но и учет общей структуры *человеческой деятельности,* осуществляемой в условиях нового уровня ее автоматизации (деятельности smart-общества), которая должна быть "понятна"используемым *интеллектуальным компьютерным системам* и которая потребует существенного переосмысления современной организации *человеческой деятельности*.

Искусственный интеллект как *область человеческой деятельности* включает в себя следующие *направления деятельности*:

- ***Научно-исследовательскую деятельность в области Искусственного интеллекта***,
в процессе которой осуществляется конкуренция различных точек зрения и подходов к построению формальных моделей различных компонентов *интеллектуальных компьютерных систем*. Конечной целью такой деятельности является постоянно развиваемая *Общая теория интеллектуальных компьютерных систем*, объектами исследования которой являются *интеллектуальные компьютерные системы* и их формальные *логико-семантические модели*, включающие в себя формальные модели различного вида *знаний*, входящих в состав *баз знаний* интеллектуальных компьютерных систем, а также различные *модели решения задач* (логические модели различного вида, нейросетевые, генетические, продукционные, функциональные и другие).
- ***Разработку Стандарта интеллектуальных компьютерных систем***, включающую в себя перманентную эволюцию этого стандарта и поддержку целостности каждой его версии. Текущая версия *Стандарта интеллектуальных компьютерных систем* — это согласованная (общепризнанная) на текущий момент часть *Общей теории интеллектуальных компьютерных систем.*
- ***Разработку технологии проектирования интеллектуальных компьютерных систем***, которая включает в себя семейство методик проектирования, а также методов и средств автоматизации *проектирования* различных *компонентов интеллектуальных компьютерных систем* и *интеллектуальных компьютерных систем* в целом. Результатом проектирования *интеллектуальных компьютерных систем* является полная формальная логико-семантическая модель этой системы.
- ***Разработку технологии реализации спроектированных интеллектуальных компьютерных систем***, а также технологий эксплуатации и сопровождения *интеллектуальных компьютерных систем*. В основе технологии реализации (производства) спроектированных *интеллектуальных компьютерных систем* лежит *универсальный интерпретатор формальных логико-семантических моделей интеллектуальных компьютерных систем*, являющихся результатом проектирования указанных систем. Указанный универсальный интерпретатор может быть реализован либо в виде *программной системы* на современных компьютерах, либо в виде *универсального компьютера нового поколения*, ориентированного на интерпретацию формальных *логико-семантических моделей интеллектуальных компьютерных систем*.
- ***Прикладную инженерную деятельность в области Искусственного интеллекта***, то есть непосредственное проектирование, реализацию и сопровождение, включающее в себя обновление (реинжиниринг), осуществляемое в ходе эксплуатации, конкретных *интеллектуальных компьютерных систем.*
- ***Учебную деятельность в области Искусственного интеллекта***, направленную на подготовку специалистов области *Искусственного интеллекта* и на перманентное повышение квалификации действующих специалистов в этой области. Без эффективной организации учебной деятельности в области *Искусственного интеллекта* быстрый прогресс в этой области невозможен. Непосредственное включение учебной деятельности в общую структуру человеческой деятельности в области *Искусственного интеллекта* обусловлено следующими обстоятельствами:
  - необходимостью глубокой *конвергенции* между различными направлениями и видами деятельности в области *Искусственного интеллекта* и соответствующей спецификой требований, предъявляемых к специалистам в этой области — каждый такой специалист должен быть достаточно компетентен и в научно-исследовательской деятельности в области *Искусственного интеллекта*, и в разработке технологий (методик и средств) *проектирования интеллектуальных компьютерных систем*, и в разработке технологий *воспроизводства* (реализации) спроектированных *интеллектуальных компьютерных систем*, а также технологий их *эксплуатации* и *сопровождения*, и в прикладной *инженерной деятельности в области Искусственного интеллекта*;
  - высокими темпами эволюции результатов в области *Искусственного интеллекта*, что делает необходимой организацию обучения соответствующих специалистов путем их непосредственного подключения не к учебным (упрощенным) проектам, а к реальным проектам, реализуемым в текущий момент. Иначе подготовленные специалисты будут иметь квалификацию "вчерашнего дня";
  - существенным расширением объемов работ в области *Искусственного интеллекта* и острой необходимостью массовой подготовки соответствующих специалистов.

Сложность ***Подготовки молодых специалистов в области Искусственного интеллекта*** заключается не только в высокой степени наукоемкости этой области, но и в том, что формирование у них соответствующих знаний и навыков осуществляется в условиях быстрого морального старения текущего состояния технологий *Искусственного интеллекта*, существенные изменения в которых происходят за время обучения студентов и магистрантов. Поэтому надо учить не текущему уровню развития *Искусственного интеллекта*, а тому уровню развития, который будет достигнут через пять и более лет.

При подготовке молодых специалистов в области *Искусственного интеллекта* необходимо формировать у них:

- культуру формализации (математическую культуру);
- системную культуру (в частности, умение осуществлять качественную стратификацию сложных динамических систем);

  - технологическую культуру (в частности, умение отличать то, что следует унифицировать и то, унификация чего ограничивает направление эволюции заданного класса сложных систем);
  - технологическую дисциплину;
  - культуру коллективного творчества (в частности, первоначальную *интероперабельность*);
  - высокую *познавательную активность* и мотивацию;
  - умение сочетать индивидуальную творческую свободу и самостоятельность с обеспечением совместимости своих результатов с результатами коллег, то есть сочетать свободу в создании (порождении) новых смыслов при согласованности (совместимости) форм их представления — о понятиях, терминах и синтаксисе не спорят, а договариваются.
- ***Организационную деятельность в области Искусственного интеллекта***, направленную на создание инфраструктуры для качественного выполнения всех остальных видов деятельности в области *Искусственного интеллекта*, а именно:
  - для обеспечения глубокой *конвергенции* между различными направлениями и видами деятельности в области *Искусственного интеллекта* и, в частности, между теорией, технологиями и инженерной практикой в этой области;
  - для обеспечения баланса между тактикой и стратегией в развитии деятельности в области *Искусственного интеллекта* как ключевой основы существенного повышения уровня автоматизации всевозможных видов *человеческой деятельности* и перехода к *smart-обществу*.

Рассмотренная декомпозиция *человеческой деятельности* в области *Искусственного интеллекта* по *видам деятельности* не является традиционным признаком декомпозиции *научно-технических дисциплин*. Обычно декомпозиция *научно-технических дисциплин* осуществляется по содержательным направлениям, которые соответствуют декомпозиции *технических систем*, исследуемых и разрабатываемых в рамках этих *научно-технических дисциплин*, то есть соответствуют выделению в этих *технических системах* различного вида компонентов. Для *Искусственного интеллекта* такими направлениями являются:
- исследование и разработка формальных моделей и языков представления знаний;
- исследование и разработка баз знаний;
- исследование и разработка логических моделей обработки знаний;
- исследование и разработка искусственных нейронных сетей;
- исследование и разработка подсистем компьютерного зрения;
- исследование и разработка подсистем обработки естественно-языковых текстов (синтаксический анализ, понимание, синтез);
- и многие другие.

Важность декомпозиции *Искусственного интеллекта* по *видам деятельности* определяется тем, что выделение различных *видов деятельности* позволяет четко ставить задачу на разработку средств автоматизации этих *видов деятельности*.

Приведем общую структуру *Человеческой деятельности в области Искусственного интеллекта*.

Человеческая деятельность в области Искусственного интеллекта
:= [Искусственный интеллект (как научно-техническая дисциплина)]
∈ *научно-техническая дисциплина*
:= [Искусственный интеллект (как научно-техническая дисциплина)]
:= [Человеческая деятельность в Предметной области интеллектуальных компьютерных систем]
∈ *деятельность*
⇒ *декомпозиция**:
  {• *Интегральная деятельность по поддержке жизненного цикла всевозможных интеллектуальных компьютерных систем*
    ⇒ *декомпозиция**:
    поддержка жизненного цикла интеллектуальных компьютерных систем
      ∈ *вид деятельности*
      ⊃ *поддержка жизненного цикла ostis-систем*
      ⇒ *обобщенная декомпозиция**:
        {• *проектирование интеллектуальных компьютерных систем*
        • *производство интеллектуальных компьютерных систем*
        • *начальное обучение интеллектуальных компьютерных систем*
        • *мониторинг качества интеллектуальных компьютерных систем*
        • *восстановление требуемого уровня качества интеллектуальных компьютерных систем*
        • *реинжиниринг интеллектуальных компьютерных систем*
        • *обеспечение безопасности интеллектуальных компьютерных систем*
        • *эксплуатация интеллектуальных компьютерных систем конечными пользователями*

}

- *Поддержка жизненного цикла Общей теории интеллектуальных компьютерных систем*
  ∈ *научно-исследовательская деятельность*
- *Поддержка жизненного цикла Стандарта интеллектуальных компьютерных систем*
  ∈ *стандартизация*
  ⇒ *часть**:
  Поддержка жизненного цикла Стандарта ostis-систем
- *Поддержка жизненного цикла Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем*
  ∈ *поддержка жизненного цикла технологий*
  ≔ [создание и сопровождение технологий]
  ⇒ *часть**:
  Поддержка жизненного цикла Технологии OSTIS
- *Поддержка жизненного цикла кадровых ресурсов для Человеческой деятельности в области Искусственного интеллекта*
- *Поддержка жизненного цикла системы комплексной организации взаимодействия между всеми направлениями Человеческой деятельности в области Искусственного интеллекта*
  ∈ *поддержка жизненного цикла метасистем комплексного управления поддержкой и обеспечением поддержки жизненного цикла сущностей соответствующего класса*

}

Технология поддержки жизненного цикла интеллектуальных компьютерных систем

⇒ *вид деятельности**:
поддержка жизненного цикла интеллектуальных компьютерных систем

⇒ *декомпозиция**:

{• *Технология проектирования интеллектуальных компьютерных систем*
  ⇒ *вид деятельности**:
  проектирование интеллектуальных компьютерных систем
- *Технология производства интеллектуальных компьютерных систем*
  ⇒ *вид деятельности**:
  производство интеллектуальных компьютерных систем
- *Технология начального обучения интеллектуальных компьютерных систем (адаптации к конкретной деятельности)*
  ⇒ *вид деятельности**:
  начальное обучение интеллектуальных компьютерных систем
- *Технология мониторинга качества интеллектуальных компьютерных систем*
  ⇒ *вид деятельности**:
  мониторинг качества интеллектуальных компьютерных систем
  ≔ [плановое тестирование и диагностика интеллектуальных компьютерных систем]
- *Технология восстановления требуемого уровня качества интеллектуальных компьютерных систем в ходе их эксплуатации*
  ≔ [Технология выявления и исправления потенциально опасных ситуаций и событий в деятельности интеллектуальных компьютерных систем (ошибок, противоречий, и так далее)]
  ⇒ *вид деятельности**:
  восстановление требуемого уровня качества интеллектуальных компьютерных систем
- *Технология реинжиниринга интеллектуальных компьютерных систем*
  ≔ [Технология совершенствования, модернизации, обновления интеллектуальных компьютерных систем]
  ⇒ *вид деятельности**:
  реинжиниринг интеллектуальных компьютерных систем
- *Технология обеспечения безопасности интеллектуальных компьютерных систем*
  ⇒ *вид деятельности**:
  обеспечение безопасности интеллектуальных компьютерных систем
- *Технология эксплуатации интеллектуальных компьютерных систем конечными пользователями*
  ⇒ *вид деятельности**:
  эксплуатация интеллектуальных компьютерных систем конечными пользователями

}

Пункт 7.1.1.3. Текущее состояние и современные проблемы научно-исследовательской деятельности в области Искусственного интеллекта

В настоящее время научные исследования в области *Искусственного интеллекта* активно развиваются по широкому спектру различных направлений (*модели представления знаний*, различного вида *логики* — дедуктивные, индуктивные, абдуктивные, четкие, нечеткие, различного вида *искусственные нейронные сети*, машинное обучение, принятие решений, целеполагание, планирование поведения, ситуационное поведение, многоагентные системы, компьютерное зрение, распознавание, интеллектуальный анализ данных, мягкие вычисления и многое другое).

Однако:

- Отсутствует согласованность систем *понятий* в разных направлениях *Искусственного интеллекта* и, как следствие, отсутствует *семантическая совместимость* и *конвергенция* этих направлений, в результате чего существенно затруднена работа в направлении построения *Общей теории интеллектуальных систем* с высоким уровнем формализации. Существование и продолжающееся увеличение "высоты барьеров"между различными направлениями исследований в области *Искусственного интеллекта* проявляется в том, что специалист, работающий в рамках какого-либо направления *Искусственного интеллекта*, посещая заседания "не своей"секции на конференции по *Искусственному интеллекту*, мало что там может понять и, соответственно, извлечь полезного для себя;
- Отсутствует мотивация и осознание острой необходимости в указанной *конвергенции* между различными направлениями *Искусственного интеллекта*;
- Отсутствует реальное движение в направлении построения *Общей теории интеллектуальных систем*, поскольку отсутствует соответствующая мотивация и осознание острой практической необходимости в этом;
- Отсутствует строгое и согласованное уточнение понятия *интеллектуальной компьютерные системы*. До сих пор для этого используется Тест Тьюринга. Поверхностная трактовка Теста Тьюринга породила различные имитации интеллекта в стиле "светского" разговора или разговора на "завалинке". На самом деле должна учитываться содержательная, целевая установка диалога, в рамках которого интеллект *интеллектуальной компьютерной системы* определяется как ее нетривиальный вклад в коллективное решение некоторой интеллектуальной (творческой) задачи.

Пункт 7.1.1.4. Текущее состояние унификации интеллектуальных компьютерных систем

Стандарты в самых различных областях являются важнейшим видом знаний, обеспечивающих согласованность различных видов массовой деятельности. Но для того, чтобы стандарты не тормозили прогресс, они должны постоянно совершенствоваться.

Стандарты должны эффективно и грамотно использоваться. Поэтому оформление стандартов в виде текстовых документов не удовлетворяет современным требованиям.

Стандарты должны быть оформлены в виде интеллектуальных справочных систем, которые способны отвечать на самые различные вопросы. Таким образом стандарты целесообразно оформлять в виде баз знаний, соответствующих интеллектуальных справочных систем. При этом указанные интеллектуальные справочные системы могут осуществлять координацию деятельности разработчиков стандартов, направленной на совершенствование этих стандартов (см. *ITAPKIT-2012el; Волков А.И..ПрофеСвОИТ-2015ст; Серенков П.С..КонцеИСкБЗ-2004ст*).

С семантической точки зрения каждый стандарт есть иерархическая онтология, уточняющих структуру и систем понятий соответствующих им предметных областей, которая описывает структуру и функционирование либо некоторого класса технических или иных искусственных систем, либо некоторого класса организаций, либо некоторого вида деятельности.

Очевидно, что для построения интеллектуальной справочной системы по стандарту и интеллектуальной системы поддержки коллективного совершенствования стандарта необходима формализация стандарта, в виде соответствующей формальной онтологии.

Конвергенция различных видов деятельности, а также конвергенция результатов этой деятельности требует глубокой семантической конвергенции (семантической совместимости) соответствующих стандартов, для чего также настоятельно необходима формализация стандартов.

Следует также заметить, что важнейшей методологической основой формализации стандартов и обеспечения их семантической совместимости и конвергенции является построение иерархической системы формальных онтологий и соблюдение *Принципа Бритвы Оккама*.

В настоящее время необходимость унификации и стандартизации *интеллектуальных компьютерных систем* не осознана, что существенно тормозит создание *комплексной технологии Искусственного интеллекта*.

Пункт 7.1.1.5. Текущее состояние и современные проблемы развития технологий проектирования интеллектуальных компьютерных систем

Современная *технология Искусственного интеллекта* представляет собой целое семейство всевозможных частных технологий, ориентированных на разработку различного вида компонентов *интеллектуальных компьютерных систем,* реализующих самые различные модели представления и обработки информации, а также ориентированных на разработку различных классов *интеллектуальных компьютерных систем*.

Однако:

- Высока трудоемкость разработки *интеллектуальных компьютерных систем*;
- Необходима высокая квалификация разработчиков;
- Современные *технологии Искусственного интеллекта* принципиально не обеспечивают разработки таких *интеллектуальных компьютерных систем,* в которых устраняются недостатки современных *интеллектуальных компьютерных систем* и, в частности, обеспечивается достаточно высокий уровень интероперабельности;
- Совместимость технологий *проектирования различных классов интеллектуальных компьютерных систем Искусственного интеллекта* практически отсутствует и, как следствие, не обеспечивается *семантическая совместимость* и взаимодействие разрабатываемых *интеллектуальных компьютерных систем*, поэтому системная интеграция *интеллектуальных компьютерных систем* осуществляется вручную;
- Отсутствует *комплексная технология проектирования интеллектуальных компьютерных систем*;
- Нет совместимости между существующими *частными технологиями проектирования различных компонентов интеллектуальных компьютерных систем* (баз знаний, решателей задач, интеллектуальных интерфейсов). Есть инструментальные средства по разработке компонентов, но "склеивать"(соединять, интегрировать) разработанные компоненты надо вручную, поскольку нет комплексных инструментальных средств, позволяющих разрабатывать *интеллектуальные компьютерные системы* в целом.

Пункт 7.1.1.6. Текущее состояние и современные проблемы развития технологий реализации спроектированных интеллектуальных компьютерных систем, а также их эксплуатации и сопровождения

Был сделан целый ряд попыток разработки *компьютеров нового поколения*, ориентированных на использование *в интеллектуальных компьютерных системах.* Но все они оказались неудачными, так как не были ориентированы на все многообразие *моделей решения задач* в *интеллектуальных компьютерных системах*. В этом смысле они не были *универсальными компьютерами* для *интеллектуальных компьютерных систем.*

Разрабатываемые *интеллектуальные компьютерные системы* могут использовать самые различные комбинации *моделей решения интеллектуальных задач* (логических моделей, соответствующих различного вида логикам, нейросетевых моделей различного вида, моделей целеполагания, синтеза планов, моделей управления сложными объектами, моделей понимания и синтеза текстов естественного языка и так далее). Однако, современные традиционные (фон-неймановские) *компьютеры* не в состоянии достаточно производительно интерпретировать все многообразие указанных *моделей решения задач*. При этом разработка специализированных *компьютеров*, ориентированных на интерпретацию какой-либо одной *модели решения задач* (нейросетевой модели или какой-либо логической модели) проблему не решает, так как в *интеллектуальной компьютерной системе* необходимо использовать сразу несколько разных *моделей решения задач*, причем в различных сочетаниях.

В настоящее время отсутствует комплексный подход к технологическому обеспечению всех этапов жизненного цикла *интеллектуальных компьютерных систем* — не только к поддержке проектирования и реализации (сборки, производства) *интеллектуальных компьютерных систем*, но также и к технологической поддержке сопровождения, реинжиниринга и эксплуатации *интеллектуальных компьютерных систем*.

Семантическая недружественность *пользовательского интерфейса* и отсутствие встроенных интеллектуальных справочных систем, позволяющих запрашивать информацию об элементах интерфейса и возможностях системы, приводят к низкой эффективности эксплуатации всех возможностей *интеллектуальной компьютерной системы*.

Пункт 7.1.1.7. Текущее состояние и современные проблемы прикладной инженерной деятельности в области Искусственного интеллекта

Накоплен достаточно большой опыт разработки *интеллектуальных компьютерных систем* самого различного назначения — систем автоматизации медицинской диагностики, а также диагностики сложных технических си-

стем, интеллектуальных обучающих, информационно-справочных и help-систем, систем естественно-языкового общения, интеллектуальных компьютерных персональных ассистентов, интеллектуальных корпоративных систем, интеллектуальных систем ситуационного управления различного рода сложными объектами, систем интеллектуального анализа больших данных, систем технического зрения и анализа сцен, интеллектуальных порталов знаний, интеллектуальных систем автоматизации проектирования различного вида сложных объектов, интеллектуальных систем автоматизации подготовки к производству спроектированной продукции различного вида, интеллектуальных автоматизированных систем управления производства различного вида продуктов, а также многих других *интеллектуальных компьютерных систем.*

Однако:

- Уровень эффективности практического использования научных результатов в области *Искусственного интеллекта* явно не соответствует современному уровню развития самих этих научных результатов. Для того чтобы повысить уровень эффективности практического использования указанных научных результатов, необходимы совместные усилия и ученых, создающих новые модели решения интеллектуальных задач, и разработчиков технологий проектирования и реализации *интеллектуальных компьютерных систем*, и разработчиков прикладных *интеллектуальных компьютерных систем.*
- Отсутствует четкая систематизация многообразия *интеллектуальных компьютерных систем*, соответствующая систематизации автоматизируемых *видов человеческой деятельности*;
- Отсутствует *конвергенция* ***интеллектуальных компьютерных систем*****,** обеспечивающих автоматизацию *областей человеческой деятельности*, принадлежащих одному и тому же *виду человеческой деятельности*;
- Отсутствует *семантическая совместимость* (семантическая унификация, взаимопонимание) между *интеллектуальными компьютерными системами*, основной причиной чего является отсутствие согласованной системы общих используемых *понятий*;
- Анализ проблем комплексной автоматизации всех *видов человеческой деятельности* убеждает в том, что дальнейшая *автоматизация человеческой деятельности* требует не только повышения уровня *интеллекта* соответствующих *интеллектуальных компьютерных* систем, но и к существенному повышению уровня их способности:
  - устанавливать свою *семантическую совместимость* (взаимопонимание) как с другими *компьютерными системами*, так и со своими пользователями;
  - поддерживать эту *семантическую совместимость* в процессе собственной эволюции, а также эволюции пользователей и других *компьютерных систем*;
  - координировать свою деятельность с пользователями и другими *компьютерными системами* при коллективном решении различных задач;
  - участвовать в распределении работ (подзадач) при коллективном решении различных задач.

Важно подчеркнуть то, что реализация вышеперечисленных способностей создаст возможность для существенной и даже полной автоматизации *системной интеграции компьютерных систем* в комплексы взаимодействующих *интеллектуальных компьютерных систем* и автоматизации реинжиниринга таких комплексов. Такая автоматизация системной интеграции и ее реинжиниринга:

- даст возможность комплексам компьютерных систем самостоятельно адаптироваться к решению новых задач;
- существенно повысит эффективность эксплуатации таких комплексов компьютерных систем, так как реинжиниринг системной интеграции компьютерных систем, входящих в такой комплекс, часто востребован (например, при реконструкции предприятий);
- существенно сокращает число ошибок по сравнению с "ручным" (неавтоматизированным) выполнением *системной интеграции* и ее *реинжиниринга*, которые, к тому же, требует высокой квалификации.

Таким образом следующий этап повышения уровня автоматизации *человеческой деятельности* настоятельно требует создания таких *интеллектуальных компьютерных систем*, которые могли бы сами (без системного интегратора) объединяться для совместного решения сложных задач.

Пункт 7.1.1.8. Текущее состояние и современные проблемы учебной деятельности в области Искусственного интеллекта

Многие ведущие университеты осуществляют подготовку специалистов в области *Искусственного интеллекта*. При этом необходимо отметить следующие особенности и проблемы текущего состояния этой деятельности:

- Поскольку *деятельность в области Искусственного интеллекта* сочетает в себе и высокую степень наукоемкости и высокую степень сложности инженерных работ, подготовка специалистов в этой области требует одновременного формирования у них как научно-исследовательских навыков и знаний, так и инженерно-практических навыков и знаний, а также системной и технологической культуры и стиля мышления. С точки зрения методики и психологии обучения сочетание фундаментальной научной и инженерно-практической подготовки специалистов является достаточно сложной педагогической задачей;

- Отсутствует *семантическая совместимость* между различными учебными дисциплинами, что приводит к "мозаичности" восприятия информации;
- Отсутствует системный подход к подготовке молодых специалистов в области *Искусственного интеллекта*;
- Нет персонификации обучения, а также установки на выявление, раскрытие и развитие индивидуальных способностей;
- Отсутствует целенаправленное формирование мотивации к творчеству;
- Нет формирования навыков работы в реальных коллективах разработчиков. Отсутствует адаптация к реальной практической деятельности;
- Любая современная технология (в том числе и технология *Искусственного интеллекта*) должна иметь высокие темпы своего развития, поскольку без этого невозможно поддерживать высокий уровень ее конкурентоспособности. Но для быстро развиваемой технологии требуется:
  - не просто высокая квалификация кадров, использующих и развивающих технологию;
  - но и высокие темпы повышения уровня этой квалификации, так как без этого невозможно эффективно использовать и развивать быстро меняющуюся технологию.

Из этого следует, что *учебная деятельность в области Искусственного интеллекта* и соответствующая ей технология должна быть не просто важной частью *деятельности* в области *Искусственного интеллекта*, а частью, глубоко интегрированной во все остальные *виды деятельности* в области *Искусственного интеллекта*. Так, например, каждая *интеллектуальная компьютерная система* должная быть ориентирована не только на обслуживание своих конечных пользователей, не только на организацию целенаправленного взаимодействия со своими разработчиками, которые постоянно совершенствуют эту систему, и не только на обеспечение минимального "порога вхождения" для новых конечных пользователей и разработчиков, но и на организацию постоянного и персонифицированного повышения квалификации каждого своего конечного пользователя и разработчика в условиях постоянных изменений, вносимых в указанную *интеллектуальную компьютерную систему.* Для этого эксплуатируемая *интеллектуальная компьютерная система* должна "знать", что в ней изменилось, на что она способна и как эти способности инициировать (содержание и форма, соответствующих пользовательских команд).

Когда мы говорим о *конвергенции* и *интеграции* в области *Искусственного интеллекта*, речь идет не только о конвергенции между *интеллектуальными компьютерными системами*, но также и между различными видами и областями *человеческой деятельности*. Таким образом, *учебная деятельность*, направленная на подготовку специалистов в области *Искусственного интеллекта*, органически входит в состав *деятельности в области Искусственного интеллекта*, а важнейшим направлением повышения эффективности этой деятельности является ее *конвергенция* и *интеграция* с другими видами *деятельности в области Искусственного интеллекта*.

Пункт 7.1.1.9. Текущее состояние и современные проблемы организационной деятельности в области Искусственного интеллекта

Острая потребность в существенном повышении уровня автоматизации в самых различных областях *человеческой деятельности* (в промышленности, медицине, транспорте, образовании, строительстве и во многих других), а также современные результаты в развитии *технологий Искусственного интеллекта* привели к существенному расширению работ по созданию *прикладных интеллектуальных компьютерных систем* и к появлению большого количества коммерческих организаций, ориентированных на разработку таких приложений.

Однако:

- Не так просто обеспечить баланс тактических и стратегических направлений развития всех видов деятельности в области *Искусственного интеллекта* (научно-исследовательской деятельности, развития технологии проектирования и производства интеллектуальных компьютерных систем, разработки прикладных систем, образовательной деятельности), а также баланс между всеми перечисленными *видами деятельности*;
- В настоящее время отсутствует глубокая *конвергенция* различных *видов деятельности* в области *Искусственного интеллекта* (в первую очередь, конвергенция развития технологий *Искусственного интеллекта* и разработки различных прикладных *интеллектуальных компьютерных систем*), что существенно затрудняет развитие каждого из этих видов деятельности и, в частности, существенно затрудняет при разработке интеллектуальных решателей задач интеграцию различных моделей решения задач (например, логических моделей, нейросетевых моделей, моделей обработки текстов естественных языков, моделей обработки сигналов — аудиосигналов, изображений);
- Высокий уровень наукоемкости работ в области *Искусственного интеллекта* предъявляет особые требования к квалификации сотрудников и к их способности работать в составе *творческих коллективов*.

Пункт 7.1.1.10. Ключевые задачи и методологические проблемы современного этапа развития Искусственного интеллекта

К числу **ключевых задач** современного этапа развития *Искусственного интеллекта* следует отнести:

- Построение ***Общей формальной теории интеллектуальных компьютерных систем***, в рамках которой была бы обеспечена совместимость всех направлений *Искусственного интеллекта*, всех моделей представления знаний, всех моделей решения задач, всех компонентов *интеллектуальных компьютерных систем*. Это предполагает:
  - Уточнение требований, предъявляемых к *интеллектуальным компьютерным системам нового поколения* — уточнение свойств *интеллектуальных компьютерных систем*, определяющих высокий уровень их *интеллекта*;
  - ***Конвергенцию*** и *интеграцию* всевозможных видов *знаний* и всевозможных *моделей решения задач* в рамках каждой *интеллектуальной компьютерной системы*.
- Создание ***инфраструктуры***, обеспечивающей интенсивное перманентное развитие *Общей формальной теории интеллектуальных компьютерных систем* в самых различных направлениях, гарантирующее сохранение логико-семантической целостности этой *теории* и совместимости всех направлений ее развития;
- На основе *Общей формальной теории интеллектуальных компьютерных систем* построение ***Технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения***, обладающих высоким уровнем *интероперабельности* и совместимости;
- Создание ***инфраструктуры***, обеспечивающей интенсивное перманентное развитие *Комплексной технологии разработки и эксплуатации интеллектуальных компьютерных систем нового поколения* в самых различных направлениях, гарантирующее сохранение целостности этой *технологии* и совместимости всех направлений ее развития;
- Разработку ***компьютеров нового поколения***, ориентированных на высокопроизводительную интерпретацию *логико-семантических моделей интеллектуальных компьютерных систем нового поколения;*
- Создание ***Глобальной экосистемы интеллектуальных компьютерных систем нового поколения***, ориентированной на комплексную автоматизацию различных видов человеческой деятельности.

Эпицентром современных методологических проблем развития *человеческой деятельности* в области *Искусственного интеллекта* является ***конвергенция*** и *глубокая интеграция* всех видов, направлений и результатов этой *деятельности*. Уровень взаимосвязи, взаимодействия и ***конвергенции*** между различными видами и направлениями деятельности в области *Искусственного интеллекта* в настоящее время явно недостаточен. Это приводит к тому, что каждая из них развивается обособленно, независимо от других. Речь идет о ***конвергенции*** между такими направлениями *Искусственного интеллекта*, как представление знаний, решение интеллектуальных задач, интеллектуальное поведение, понимание и так далее, а также между такими *видами человеческой деятельности в области Искусственного интеллекта*, как научные исследования, разработка технологий, разработка приложений, образование, бизнес. Почему на фоне уже достаточно длительного интенсивного развития научных исследований в области *Искусственного интеллекта* до сих пор не создан рынок *интеллектуальных компьютерных систем* и комплексная технология *Искусственного интеллекта*, обеспечивающая разработку широкого спектра *интеллектуальных компьютерных систем* самого различного назначения и доступной широкому контингенту инженеров. Потому что сочетание высокого уровня наукоемкости и прагматизма этой проблемы требует для ее решения принципиально нового подхода к организации взаимодействия ученых, работающих в области *Искусственного интеллекта*, разработчиков средств автоматизации проектирования *интеллектуальных компьютерных систем,* разработчиков средств реализации *интеллектуальных компьютерных систем*, включая средства аппаратной поддержки *интеллектуальных компьютерных систем*, разработчиков прикладных *интеллектуальных компьютерных систем*. Такое целенаправленное взаимодействие должно осуществляться как в рамках каждой из этих форм деятельности в области *Искусственного интеллекта*, так и между ними. Таким образом, основной тенденцией дальнейшего развития теоретических и практических работ в области *Искусственного интеллекта* является ***конвергенция*** как самых разных видов (форм и направлений) *человеческой деятельности* в области *Искусственного интеллекта*, так и самых разных продуктов (результатов) этой деятельности. Необходимо ликвидировать барьеры между различными видами и продуктами деятельности в области *Искусственного интеллекта* в целях обеспечения их совместимости и интегрируемости.

Конвергенция разрабатываемых *интеллектуальных компьютерных систем* преобразует набор индивидуальных (автономных) *интеллектуальных компьютерных систем* различного назначения в коллектив активно взаимодействующих *интеллектуальных компьютерных систем* для совместного (коллективного) решения сложных (комплексных) задач и для перманентной поддержки совместимости между всеми *интеллектуальными компьютерными системами*, входящими в коллектив, в процессе индивидуальной эволюции каждой из этих систем.

Конвергенция конкретных искусственных сущностей (например, технических систем) есть стремление к их унификации (в частности, к стандартизации), то есть стремление к минимизации многообразия форм решения аналогичных практических задач — стремление к тому, чтобы все, что можно сделать одинаково, делалось одинаково,

но без ущерба требуемого качества. Последнее очень важно, так как безграмотная стандартизация может привести к существенному торможению прогресса. Ограничение многообразия форм не должно приводить к ограничению содержания, возможностей. Образно говоря, "словам должно быть тесно, а мыслям — свободно".

Методологически ***конвергенция*** искусственно создаваемых сущностей (артефактов) сводится (1) к выявлению (обнаружению) принципиальных сходств между этими сущностями, которые часто весьма закамуфлированы и их трудно "увидеть" и (2) к реализации обнаруженных сходств одинаковым образом (в одинаковой форме, в одинаковом "синтаксисе"). Образно говоря, от "семантической" (смысловой) эквивалентности требуется перейти и к "синтаксической" эквивалентности. Кстати, в этом как раз и заключается суть *смыслового представления информации*, целью которого является создание такой языковой среды (смыслового пространства), в рамках которого (1) семантически эквивалентные информационные конструкции полностью совпадали бы, а (2) ***конвергенция*** информационных конструкций сводилась бы к выявлению изоморфных фрагментов этих конструкций.

К числу общих методологических проблем современного этапа развития *Искусственного интеллекта* можно отнести:

- Отсутствие массового осознания того, что создание рынка *интеллектуальных компьютерных систем нового поколения*, обладающих *семантической совместимостью* и высоким уровнем *интероперабельности*, а также создание комплексов (экосистем), состоящих из таких *интеллектуальных компьютерных систем* и обеспечивающих автоматизацию различных *видов человеческой деятельности*, невозможно, если коллективы разработчиков таких систем и комплексов существенно не повысят уровень *социализации* **всех** своих сотрудников. Уровень качества коллектива разработчиков, то есть уровень квалификации сотрудников и уровень согласованности их деятельности, должен превышать уровень качества систем, разрабатываемых этим коллективом. Особое значение рассматриваемая проблема согласованности деятельности специалистов в области *Искусственного интеллекта* имеет для построения *Общей формальной теории интеллектуальных компьютерных систем нового поколения*, а также *Комплексной технологии разработки и эксплуатации интеллектуальных компьютерных систем нового поколения*;
- Далеко не всеми учеными, работающими в области *Искусственного интеллекта*, принимается прагматичность, практическая направленность *Искусственного интеллекта*;
- Не всеми принимается необходимость ***конвергенции*** различных направлений *Искусственного интеллекта* и необходимость их интеграции в целях построения *Общей формальной теории интеллектуальных компьютерных систем*;
- Не всеми принимается необходимость ***конвергенции*** различных видов деятельности в области *Искусственного интеллекта*;
- Важным препятствием для ***конвергенции*** результатов научно-технической деятельности является сформировавшийся в науке и технике акцент на выявлении не сходств, а отличий. Чтобы убедиться в этом достаточно обратить внимание на то, что уровень научных результатов оценивается научной новизной, которая может имитироваться новизной не по существу, а по форме представления (например, с помощью новых понятий или даже новых терминов). Результаты в технике, например, в патентах также оцениваются отличиями от предшествующих технических решений. Но для ***конвергенции*** нужны другие акценты — не поиск отличий, а выявление неочевидных сходств и превращение их в очевидные сходства, представленные в одинаковой форме;
- Нет движения к построению *Комплексной технологии проектирования, реализации, сопровождения, реинжиниринга и эксплуатации интеллектуальных компьютерных систем*. Речь идет о комплексном подходе к технологическому обеспечению всех этапов жизненного цикла *интеллектуальных компьютерных систем*;
- Нет активного развития работ по созданию *Глобальной экосистемы интеллектуальных компьютерных систем нового поколения*;
- В основе современной организации и автоматизации *человеческой деятельности* лежит "*Вавилонское столпотворение*" постоянно расширяемого многообразия *языков*. Имеются в виду не только *естественные языки*, но и *формальные язык*и, направленные на точное представление *знаний* различного вида. Многообразие различных *специализированных языков* пронизывает всю *человеческую деятельность* — во многих областях *человеческой деятельности* для решения различных видов *задач*, для разработки различных *моделей решения задач* создаются *специализированные языки*. Примером этого является многообразие *языков программирования*. *Специализированные языки* могут и должны появляться, но только как *подъязыки* более общих *языков*, синтаксис каждого из которых совпадает с *синтаксисом* всех соответствующих ему *подъязыков*. При этом в рамках *Общей формальной теории интеллектуальных компьютерных систем* должен быть выделен один *универсальный формальный язык* — язык-ядро, по отношению к которому все остальные используемые *формальные языки* являются *подъязыками*. *денотационная семантика* указанного *универсального формального языка* должна задаваться соответствующей *формальной онтологией* максимально высокого уровня. Иначе о какой ***конвергенции*** и *интеграции знаний*, о какой *семантической совместимости* компьютерных систем можно вести речь.

В основе предлагаемой организации *человеческой деятельности* в области *Искусственного интеллекта* лежат следующие положения:

- ***комплексная конвергенция*** — как "вертикальная" *конвергенция* между различными *видами деятельности* в области *Искусственного интеллекта*, так и "горизонтальная" *конвергенция* в рамках каждого из этих *видов деятельности*, соответствующая различным компонентам или различным классам *интеллектуальных компьютерных систем* — базам знаний, решателям задач, различным моделям решения задач, различным видам интерфейсов (зрительным, аудио, естественно-языковым), робототехническим интеллектуальным компьютерным системам, интеллектуальным обучающим системам, интеллектуальным автоматизированным системам управления, интеллектуальным системам автоматизации проектирования и так далее;
- ***"горизонтальная" конвергенция*** в рамках каждого вида *человеческой деятельности* в области *Искусственного интеллекта* включает в себя:
  - *конвергенцию* в рамках *научно-исследовательской деятельности в области Искусственного интеллекта*, означающую переход от независимого развития различных направлений *Искусственного интеллекта* к общей теории *интеллектуальных компьютерных систем*;
  - *конвергенцию* в рамках развития *технологий Искусственного интеллекта*, означающую переход от независимого развития частных технологий к созданию единого комплекса семантически совместимых частных технологий;
  - *конвергенцию* в рамках *инженерной деятельности в области Искусственного интеллекта*, означающую переход от практики независимой разработки различных прикладных *интеллектуальных компьютерных систем* к разработке комплекса (экосистемы) интероперабельных *интеллектуальных компьютерных систем*;
  - *конвергенцию* в рамках *учебной деятельности в области Искусственного интеллекта*, обозначающую переход от изучения отдельных учебных дисциплин к формированию у молодых специалистов целостной картины текущего состояния *Искусственного интеллекта* и проблемных направлений дальнейшего развития;
  - *конвергенцию* в рамках *общей организационной деятельности в области Искусственного интеллекта*, переход от отдельных вышеперечисленных видов деятельности в области *Искусственного интеллекта* к единому комплексу всех этих видов деятельности и обеспечивающую конвергенцию и интеграцию указанных видов деятельности в области *Искусственного интеллекта*, что существенно повысит их качество, поскольку каждый из этих видов деятельности находится в сильной зависимости от всех остальных;
- организация разработки и перманентного развития предлагаемой *технологии* в виде ***открытого международного проекта***, предоставляющего:
  - свободный доступ к использованию текущей версии разрабатываемой *технологии*;
  - возможность каждому желающему войти в состав коллектива разработчиков этой *технологии*;
- ***поэтапность*** процесса формирования рынка *семантически совместимых* и *активно взаимодействующих* между собой *интеллектуальных компьютерных систем нового поколения*, начальными этапами которого являются:
  - разработка *логико-семантических моделей* (баз знаний) нескольких *прикладных интеллектуальных компьютерных систем нового поколения*;
  - программная реализация на современных *ostis-платформах*;
  - установка каждой разработанной *логико-семантической модели прикладной интеллектуальной компьютерной системы* на *ostis-платформу* с последующим *тестированием* и *реинжинирингом* каждой такой модели;
  - разработка и перманентное совершенствование логико-семантической модели (базы знаний) *интеллектуальной компьютерной метасистемы*, которая содержит (1) описание *стандарта интеллектуальных компьютерных систем нового поколения*, (2) *библиотеку* многократно используемых (в различных *интеллектуальных компьютерных системах*) знаний различного вида и, в частности, различных *методов решения задач*, (3) *методы проектирования* и *средства поддержки проектирования различных видов компонентов интеллектуальных компьютерных систем* (компонентов *баз знаний, решателей задач, интерфейсов*);
  - разработка *ассоциативного семантического компьютера* в качестве аппаратной реализации *платформы интерпретации логико-семантических моделей интеллектуальных компьютерных систем нового поколения*;
  - перенос разработанных *логико-семантических моделей интеллектуальных компьютерных систем нового поколения* на новые, более эффективные варианты реализации платформы интерпретации этих моделей;
  - развитие *рынка интеллектуальных компьютерных систем нового поколения* в виде Глобальной экосистемы, состоящей из активно взаимодействующих таких систем и ориентированной на комплексную автоматизацию всех *видов человеческой деятельности*;
  - создание ***рынка знаний*** на основе *Экосистемы OSTIS*;
  - автоматизация *реинжиниринга* эксплуатируемых *интеллектуальных компьютерных систем нового поколения* в направлении приведения их в соответствие с новыми версиями *стандарта интеллектуальных*

компьютерных систем путем автоматической замены устаревших *компонентов* в этих системах на текущие версии этих компонентов.

Следует особо подчеркнуть, что **ключевым фактором решения рассматриваемых методологических проблем** в области *Искусственного интеллекта* являются различные направления ***конвергенции*** и ***интеграции***, обеспечивающие переход к ***интеллектуальным компьютерным системам нового поколения***, к соответствующей технологии комплексной поддержки их жизненного цикла и к существенному повышению уровня автоматизации всего комплекса человеческой деятельности:

- *конвергенция* и *интеграция* различных моделей представления и обработки *информации* в *интеллектуальных компьютерных системах нового поколения*
  - *конвергенция* и *интеграция* различных *видов знаний* в *базах знаний интеллектуальных компьютерных систем нового поколения*
  - *конвергенция* и *интеграция* различных *моделей решения задач*
  - *конвергенция* и *интеграция* различных *видов интерфейсов интеллектуальных компьютерных систем нового поколения*
- *конвергенция* и *интеграция* различных направлений *Искусственного интеллекта* в целях построения *Общей формальной теории интеллектуальных компьютерных систем нового поколения*
- *конвергенция* и *интеграция* технологий *проектирования* различных *компонентов интеллектуальных компьютерных систем нового поколения* в целях построения комплексной *Технологии проектирования интеллектуальных компьютерных систем нового поколения*
- *конвергенция* и *интеграция* технологий поддержки различных *этапов жизненного цикла интеллектуальных компьютерных систем нового поколения* в целях построения *Технологии комплексной поддержки всех этапов жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *конвергенция* и *интеграция* различных *видов человеческой деятельности в области Искусственного интеллекта* (*научно-исследовательской деятельности*, *развития технологического комплекса*, *прикладной инженерии*, *образовательной деятельности*) для повышения уровня согласованности и координации этих *видов деятельности*, а также для повышения уровня их комплексной автоматизации с помощью ***семантически совместимых*** *интеллектуальных компьютерных систем нового поколения*
- *конвергенция* и *интеграция* самых различных *видов и областей человеческой деятельности*, а также средств комплексной автоматизации этой деятельности с помощью *интеллектуальных компьютерных систем нового поколения*

Конечным практическим результатом человеческой деятельности в области *Искусственного интеллекта* является:

- Реорганизация и комплексная автоматизация *человеческой деятельности в области Искусственного интеллекта* с помощью *интеллектуальных компьютерных систем нового поколения*;
- Поэтапное создание глобальной сети эффективно взаимодействующих ***интеллектуальных компьютерных систем нового поколения***, обеспечивающих комплексную автоматизацию всевозможных видов и областей *человеческой деятельности*.

Переход от современных интеллектуальных компьютерных систем к *интеллектуальным компьютерным системам нового поколения* и к соответствующей комплексной технологии не требует от специалистов в области Искусственного интеллекта изменения сферы их научных интересов. От них требуется только преодолеть синдром "*Вавилонского столпотворения*", оформляя свои научные результаты как часть общего коллективного продукта.

Проблемы текущего этапа развития *Искусственного интеллекта*, направленного на создание Общей теории и технологии *интеллектуальных компьютерных систем нового поколения*, требуют фундаментального комплексного междисциплинарного подхода и принципиально новой организации научно-технической деятельности.

Пункт 7.1.1.11. Комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения

В рамках *Технологии OSTIS* поддержка жизненного цикла интеллектуальных компьютерных систем нового поколения (*ostis-систем*) осуществляется на основе *Метасистемы OSTIS*, которая относится к классу *ostis-систем* и фактически является формой реализации указанной Технологии. Автоматизация поддержки жизненного цикла *ostis-систем* осуществляется как в форме инструментального обслуживания инженерной деятельности (в частности, Метасистема OSTIS является системой автоматизации проектирования ostis-систем), так и в форме информационного обслуживания указанной деятельности. Для этого база знаний *Метасистема OSTIS* содержит:

- текущее состояние полного текста *Стандарта ostis-систем*;

- Текущее состояние Библиотеки многократно используемых компонентов ostis-систем;
- Используемые и реализуемые инженерами методики поддержки жизненного цикла ostis-систем;
- Документацию инструментальных средств, инженерами для поддержки жизненного цикла ostis-систем.

Кроме всего этого, *Метасистема OSTIS*:
- Обеспечивает автоматизацию *Поддержки жизненного цикла Стандарта ostis-систем*, то есть обеспечивает организацию взаимодействия между авторами этого Стандарта, направленного на перманентное его развитие;
- Обеспечивает автоматизацию *Поддержки жизненного цикла Технологии OSTIS*, которая сводится к поддержке жизненного цикла основной части базы знаний *Метасистемы OSTIS*, которая является полной документацией текущего состояния *Технологии OSTIS*.

Автоматизация других направлений *Человеческой деятельности в области Искусственного интеллекта* также может осуществляться с помощью *ostis-систем*, семантически совместимых и взаимодействующих с *Метасистемой OSTIS* в рамках *Экосистемы OSTIS*.

§ 7.1.2. Проблемы и перспективы комплексной автоматизации всевозможных видов и областей человеческой деятельности с помощью интеллектуальных компьютерных систем нового поколения

⇒ *подраздел**:
- *Пункт 7.1.2.1. Общие принципы систематизации человеческой деятельности и ее комплексной автоматизации с помощью интеллектуальных компьютерных систем нового поколения*
- *Пункт 7.1.2.2. Многообразие видов человеческой деятельности и связей между ними*

Выше было рассмотрено то, как осуществляется и автоматизируется с помощью интеллектуальных компьютерных систем нового поколения весь комплекс *Человеческой деятельности в области Искусственного интеллекта*. Сейчас обобщим это и рассмотрим принципы организации и комплексной автоматизации *человеческой деятельности* в целом, то есть автоматизации самых различных видов и областей человеческой деятельности.

Пункт 7.1.2.1. Общие принципы систематизации человеческой деятельности и ее комплексной автоматизации с помощью интеллектуальных компьютерных систем нового поколения

Опыт комплексной организации, структуризации и автоматизации *человеческой* деятельности в области *Искусственного интеллекта* (в области создания и сопровождения интеллектуальных компьютерных систем) можно обобщить и для других областей человеческой деятельности. Это обусловлено следующими причинами:
- Во-первых, потому, что человеческая деятельность, направленная на поддержку всего *жизненного цикла интеллектуальных компьютерных систем* нового поколения, является частной областью деятельности по отношению к виду человеческой деятельности, направленному на (обеспечивающему) поддержку всего жизненного цикла любой искусственной (искусственно создаваемой) сущности (любого артефакта). В зависимости от сложности искусственно создаваемой сущности, уровень сложности человеческой деятельности, направленной на поддержку жизненного цикла этой сущности, может быть самым различным, но общая структура этой деятельности, соответствующая различным этапам жизненного цикла искусственно создаваемых сущностей, а также необходимым направлениям обеспечения этой инженерной деятельности является одинаковой для искусственных сущностей различных классов. К указанным направлениям обеспечения поддержки жизненного цикла искусственных сущностей относятся:
  - научно-исследовательская деятельность, направленная на изучение искусственных сущностей соответствующего класса;
  - разработка стандарта искусственных сущностей указанного класса;
  - разработка технологии поддержки искусственных сущностей указанного класса;
  - подготовка кадров, способных осуществлять поддержку жизненного цикла искусственных сущностей указанного класса, то есть способных эффективно использовать указанную выше технологию;
  - подготовка кадров, способных участвовать в указанной выше научно-исследовательской деятельности;
  - подготовка кадров, способных участвовать в разработке стандарта искусственных сущностей заданного класса;
  - подготовка кадров, способных участвовать в разработке и развитии указанной выше технологии;
  - организационное обеспечение всего комплекса работ по развитию и использованию указанной технологии.

- Во-вторых, потому, что многие сложные технические системы фактически становятся *интеллектуальными компьютерными системами* (в том числе распределенными) с различными наборами сенсорных и эффекторных подсистем — интеллектуальными автомобилями с автопилотом и автоштурманом, интеллектуальными заводами-автоматами, умными домами, умными городами и тому подобными.
- В-третьих, потому, что характер деятельности *интеллектуальных компьютерных систем нового поколения* и характер деятельности каждого *человека* и каждой организации по сути мало чем отличаются, поскольку *интеллектуальные компьютерные системы нового поколения* становятся равноправными партнерами (субъектами) *человеческой деятельности*, так как уровень их самостоятельности, ответственности, интероперабельности и интеллектуальности приближается к соответствующим качествам *естественных* субъектов человеческой деятельности (физических лиц, юридических лиц, подразделений крупных организаций, неформальных организаций).

Итак, структуризацию *человеческой деятельности* в области *Искусственного интеллекта* на основе понятий *вида деятельности*, *области деятельности*, *продукта деятельности* (объекта деятельности) можно легко обобщить для всех *научно-технических дисциплин*, что дает возможность рассматривать автоматизацию деятельности в рамках всех *научно-технических дисциплин* с общих позиций, так как автоматизация различных *видов деятельности* в рамках различных *научно-технических дисциплин* может выглядеть аналогичным образом, а иногда может быть реализована с помощью одной и той же *интеллектуальной компьютерной системы*. Так например, любая *интеллектуальная компьютерная система автоматизации проектирования* технических систем заданного вида может быть построена на основе *интеллектуальной компьютерной системы автоматизации проектирования и реинжиниринга баз знаний*, поскольку результатом проектирования любой *технической системы* является формальная модель (описание, спецификация, документация) этой *технической системы*, обладающая достаточно полнотой для воспроизводства (реализации) этой системы.

На текущем этапе развития *Искусственного интеллекта* необходимо переходить от автоматизации отдельных *видов человеческой деятельности* к интегрированной автоматизации всего комплекса *человеческой деятельности*, к созданию и постоянной эволюции всей ***Глобальной экосистемы интеллектуальных компьютерных систем***, самостоятельно взаимодействующих как между собой, так и с людьми, автоматизацию деятельности которых они осуществляют, а также с современными компьютерными системами, не являющимися интеллектуальными системами. При этом надо помнить, что основные "накладные" расходы, основные проблемы, возникают на "стыках" при интеграции различных технических решений. Разработчик каждой подсистемы должен гарантировать отсутствие указанных "накладных" расходов. При этом необходимо подчеркнуть, что следует ориентироваться не столько на создание эффективной *Глобальной экосистемы интеллектуальных компьютерных систем*, сколько на создание эффективных методик и средств, направленных на *перманентную эволюцию* такой *экосистемы*.

Методика комплексной автоматизации *человеческой деятельности* включает в себя следующие этапы:
- Построение общей ***структуры человеческой деятельности***, в основе которой лежит иерархия *человеческой деятельности* по видам деятельности и продуктам деятельности с четкой фиксацией различного вида связей между различными компонентами этой структуры.
- Формализация различных *видов человеческой деятельности*.
- Разработка ***технологии***, обеспечивающей максимально возможную автоматизацию этой деятельности с помощью *интеллектуальных компьютерных систем нового поколения*.
- Обеспечение максимально возможной ***конвергенции*** различных *видов деятельности*, что позволит сократить многообразие средств автоматизации (то есть соответствующих *интеллектуальных компьютерных систем нового поколения*).
- Обеспечение максимально возможной ***конвергенции технологий*** выполнения одного и того же *вида деятельности* для разных объектов деятельности (конвергенции технологий проектирования объектов различных классов, конвергенции технологий мониторинга, профилактики и диагностики для агентов различных классов и так далее) и, тем самым, обеспечить ***конвергенцию*** соответствующих средств автоматизации, построенных на основе *интеллектуальных компьютерных систем нового поколения*.

Пункт 7.1.2.2. Многообразие видов человеческой деятельности и связей между ними

Базовым видом человеческой деятельности можно считать ***поддержку жизненного цикла*** различных сущностей.

Классом объектов деятельности для этого *вида деятельности* является класс всевозможных социально значимых объектов, на которые имеет смысл воздействовать, поддержку жизненного цикла которых целесообразно осуществлять.

поддержка жизненного цикла
:= [поддержка жизненного цикла социально значимых сущностей]
$\in$ *вид деятельности*

⇒ *частный вид деятельности, выполняемой на некотором этапе**:
- *проектирование*
- *производство*
- *начальное обучение*
  ≔ [настройка]
- *мониторинг качества*
  ≔ [плановое обследование и диагностика]
- *восстановление требуемого уровня качества*
  ≔ [ремонт, лечение]
- *реинжиниринг*
  ≔ [обновление, совершенствование]
- *обеспечение безопасности*
- *использование*
  ≔ [эксплуатация, употребление]

⇒ *частный вид деятельности над подклассом объектов деятельности**:
- *научно-исследовательская деятельность*
  ≔ [поддержка жизненного цикла научных теорий]
  ⇒ *класс объектов деятельности**:
  научная теория
- *стандартизация*
  ≔ [поддержка жизненного цикла стандартов]
  ⇒ *класс объектов деятельности**:
  стандарт
- *поддержка жизненного цикла технологий*
  ⇒ *класс объектов деятельности**:
  технология
- *образовательная деятельность*
  ≔ [учебная деятельность]
  ≔ [поддержка жизненного цикла кадровых ресурсов]
  ⇒ *класс объектов деятельности**:
  кадровый ресурс
- *поддержка жизненного цикла метасистем комплексного управления поддержкой и обеспечение поддержки жизненного цикла сущностей соответствующих классов*
  ⇒ *класс объектов деятельности**:
  метасистема комплексного управления поддержкой и обеспечением поддержки жизненного цикла сущностей соответствующих классов

Когда выше рассматривалась общая структура *человеческой деятельности* путем обобщения структуры ***Человеческой деятельности в области Искусственного интеллекта***, мы:
- ввели понятие *вида деятельности*
- в качестве "исходной точки" обобщения выбрали такой *вид деятельности*, как *поддержка жизненного цикла интеллектуальных компьютерных систем*
- далее расширяли класс *объектов деятельности* выбранного *вида деятельности*,
  - переходя от класса *интеллектуальных компьютерных систем* к классу всевозможных *искусственных материальных сущностей*
  - объединяя класс *искусственных материальных сущностей* с классом *естественных материальных сущностей* (материальных сущностей естественного происхождения), а также с классом *естественно-искусственных материальных сущностей* (либо *естественных искусственно модифицированных материальных сущностей*, либо *гибридных естественно-искусственных материальных сущностей*, имеющих компоненты как естественного так и искусственного происхождения);
  - объединяя класс *материальных сущностей* с классом *информационных ресурсов*, то есть социально значимых информационных конструкций (документов), являющихся продуктами соответствующих действий или деятельностей (*научными теориями*, *стандартами*, *базами знаний*, *методами*, *проектными документациями* соответствующих создаваемых объектов)
  - объединяя класс *материальных сущностей информационных ресурсов* с классом *материально-информационных объектов*, к которым, в частности, относятся различные технологии.

Таким образом, *поддержка жизненного цикла* различных социально значимых объектов является особым видом *человеческой деятельности*. Во-первых, эффективность *Человеческой деятельности* в целом зависит (1) от длительности социально полезной (активной) фазы жизненного цикла используемых объектов и (2) от объема затрат общества на поддержание необходимых социально полезных свойств используемых объектов. Во-вторых, характер и *технология* поддержки жизненного цикла разных видов социально значимых объектов могут суще-

ственно отличаться друг от друга. Так, например, существенно отличается организация поддержки жизненного цикла автомобилей, традиционных компьютерных систем различного назначения, современных интеллектуальных компьютерных систем, интероперабельных интеллектуальных компьютерных систем, людей, предприятий, домов, различных юридических лиц, населенных пунктов и других. При этом типология социально значимых объектов, жизненный цикл которых должен поддерживаться, включает в себя самые разнообразные классы объектов - искусственно создаваемые материальные информационные продукты человеческой деятельности, всех людей, всевозможные социальные сообщества и предприятия. Многообразие типов социально значимых объектов порождает многообразие соответствующих им технологий, что усложняет комплексную автоматизацию человеческой деятельности в целом.

Тем не менее, заметим, что *видов человеческой деятельности* значительно меньше, чем *областей человеческой деятельности*. Это в определенной степени обусловлено тем, что видов связей между сущностями (относительных понятий) значительно меньше, чем классов различных сущностей. Данное обстоятельство указывает на то, что в основе движения в направление глобальной автоматизации деятельности *общества* должна лежать ориентация на грамотную систематизацию *видов человеческой деятельности*, и на их максимально глубокую *конвергенцию* (как внутри каждого вида деятельности, так и между различными видами). Благодаря этому искусственно привносимое многообразие средств автоматизации *человеческой деятельности* может быть сведено к минимуму.

следует отличать

∋ {• *научно-исследовательская деятельность*
  ≔ [поддержка жизненного цикла научных теорий]
  • *стандартизация*
  ≔ [разработка и развитие стандартов]
  ≔ [поддержка жизненного цикла стандартов]
  • *поддержка жизненного цикла технологий*
}

Научно-исследовательская деятельность направлена на ***изучение сущностей заданного класса***, на изучение принципов, лежащих в основе их структуры и функционирования. В рамках этого вида деятельности важна новизна и конкуренция идей и подходов, важно соотношение между структурой (архитектурой) организацией функционирования исследуемых *сущностей* и общими характеристиками (параметрами) качества этих сущностей, общими предъявляемыми к ним требованиями. Продуктом рассматриваемой деятельности является *Общая теория сущностей заданного класса*, которая отражает множественность и даже противоречивость разных точек зрения и важнейшим направлением развития (эволюции) которой является сближение (*конвергенция*) различных точек зрения и обеспечение совместимости и непротиворечивости между ними. В основе *научно-исследовательской деятельности* лежит конкуренция точек зрения, принципиальная новизна идей и верифицированных результатов, направленных на выявление и обоснование неочевидных свойств и закономерностей соответствующей *предметной области*, на разработку методов решения различных *классов задач*, решаемых в рамках этой *предметной области*. Цель *научно-исследовательской деятельности* и требуемая детализация вырабатываемых знаний об объектах исследований соответствующих *предметных областей*.

В отличие от *научно-исследовательской деятельности* в основе разработки *стандарта* создаваемых сущностей и разработки соответствующей *технологии* поддержки их жизненного цикла лежит согласование различных точек зрения (поиск консенсуса) и максимально возможное их упрощение (соблюдение *Принципа Бритвы Оккама*). Необходимость такой методологической установки обусловлена массовым характером *человеческой деятельности* по созданию и *поддержке жизненного цикла* соответствующего класса сущностей и необходимостью вовлечения в эту деятельность людей с *разной* (в том числе и достаточно низкой) квалификацией. В процессе *разработки стандарта сущностей заданного класса* важна не конкуренция различных точек зрения, а их *конвергенция*, *семантическая совместимость* и глубокая интеграция. Каждый *стандарт искусственных сущностей заданного класса* — это согласованная на текущий момент точка зрения (консенсус) о структуре, функционировании, свойствах и закономерностях искусственных сущностей заданного класса, согласованная (общепризнанная) часть *Общей теория искусственных сущностей заданного класса*, доступная для понимания широкому контингенту практиков (инженеров), которые проектируют, производят и поддерживают весь жизненный цикл конкретных *искусственных сущностей заданного класса*.

При создании и *поддержке жизненного цикла технологий* должны учитываться ряд требований, предъявляемых к любым *технологиям*:

- комплексность — максимально возможное покрытие всех задач, которые должны решаться с помощью *технологии* (как минимум всех этапов жизненного цикла)
- максимально возможная простота в использовании *технологии* (необходимая полнота документации, интеллектуальная help-поддержка, отсутствие лишней информации, которая не является необходимой для использования *технологии*, наличие богатой и систематизированной библиотеки типовых многократно используемых решений)

общество — это иерархическая система взаимодействующих индивидуальных и коллективных субъектов, каждый из которых:

- Производит либо часть социально значимой продукции, производимой коллективным субъектом, в состав которого входит данный субъект, либо целостный социально-значимый продукт (производимый товар), потребляемый другими внешними субъектами или оказывает некоторую услугу другому субъекту, направленную на обеспечение жизнедеятельности и совершенствование этого другого субъекта.
- Потребляет продукцию, произведенную другими субъектами, необходимую для производства собственной продукции (сырье и оборудование), а также необходимую для обеспечения своей жизнедеятельности.
- Потребляет услуги, оказываемые другими субъектами необходимые для производства собственной продукции или услуг, а также необходимые для совершенствования своей деятельности.

Основными направлениями автоматизации всего комплекса *человеческой деятельности* являются:

- автоматизация социально полезной профессиональной деятельности всех субъектов деятельности (как индивидуальных субъектов — всех физических лиц, так и всевозможных коллективных — корпоративных субъектов, в том числе юридических лиц)
- автоматизация обеспечения (создания) комфортных условий для всех субъектов деятельности общества на основе мониторинга деятельности и конкретного (адаптированного) содействия эволюции каждого субъекта с учетом его непосредственных потребностей и проблем.

Организация взаимодействий каждого субъекта с внешней средой должна осуществляться как со стороны этого субъекта, так и со стороны указанной внешней среды (то есть со стороны общества). *общество* должно повернуться "лицом" к каждому субъекту и не бросать его на произвол судьбы. В настоящее время создание (обеспечение) условий субъектов деятельности общества отдано на откуп каждого такого субъекта. Общество в лице специально предназначенных для этого других субъектов оказывает услуги и снабжает товарами по заказу (по инициативе) нуждающегося в этом субъекта. Таким образом, ответственность за развитие каждого субъекта деятельности ложится исключительно на "плечи" этого субъекта. Поддержка общества носит общий характер и никак не учитывает особенности текущего положения каждого субъекта.

Важнейшей причиной, препятствующей дальнейшему повышению общего уровня автоматизации человеческой деятельности является то, что автоматизация различных областей человеческой деятельности осуществляется локально.

На современном этапе применения интеллектуальных компьютерных систем основной проблемой является не автоматизация локальных видов и областей человеческой деятельности, а автоматизация комплексных процессов человеческой деятельности, требующая *интеграции* в априори непредсказуемых комбинациях самых различных информационных ресурсов и самых различных автоматизированных сервисов, реализуемых в виде специализированных интеллектуальных компьютерных систем.

Локальность автоматизации человеческой деятельности приводит к тому, что вся человеческая деятельность приобретает облик "архипелага", состоящего из хорошо автоматизированных "островов", но соединяемых между собой "вручную". Это "ручное" не автоматизируемое соединение указанных "островов" полностью зависит от человеческого фактора и квалификации соответствующих исполнителей.

Указанное "ручное" соединение некоторого множества семантически близких автоматизированных областей человеческой деятельности можно автоматизировать, но делать это надо очень грамотно на высоком уровне системной культуры и на фундаментальной основе общей теории человеческой деятельности.

Еще одна важная причина, препятствующая дальнейшему повышению общего уровня автоматизации общества заключается в том, что автоматизация различных областей человеческой деятельности осуществляется без выявления и глубокого анализа сходства некоторых видов деятельности в разных областях и соответственно без сближения, ***конвергенции*** и ***унификации*** этих *видов деятельности*.

Важнейшим направлением повышения уровня автоматизации человеческой деятельности является переход к автоматизации все более и более комплексных (крупных) видов и областей человеческой деятельности например, от автоматизации деятельности различных предприятий, организаций, хозяйственных служб к автоматизации деятельности города в целом).

Автоматизации комплексных видов человеческой деятельности требует создания комплекса активно взаимодействующих компьютерных систем, каждая из которых обеспечивает автоматизацию соответствующего частного вида человеческой деятельности, входящего в состав автоматизируемого комплексного вида деятельности. При этом число уровней иерархии автоматизируемых видов человеческой деятельности ничем не ограничивается. Очевидно, что уровень автоматизации комплексных видов человеческой деятельности определяется:

- уровнем конвергенции (сближения, совместимости) соответствующих частных видов деятельности;
- качеством интеграции этих частных видов деятельности;
- уровнем конвергенции компьютерных систем, обеспечивающих автоматизацию указанных частных видов деятельности;
- качеством взаимодействия этих компьютерных систем то есть уровнем интероперабельности этих систем).

[Уровень эволюции общества во многом зависит от уровня автоматизации человеческой деятельности, от уровня развития соответствующих технологий такой автоматизации. Но эта зависимость выглядит значительно сложнее чем, кажется на первый взгляд, особенно, если речь идет об автоматизации не физической, интеллектуальной человеческой деятельности (как индивидуальной, так и коллективной). Безграмотная, а тем более социально безответственная или злонамеренная автоматизация информационной деятельности общества способны нанести огромный ущерб его развитию. Такая безграмотность и безответственность, например, приводит к таким побочным факторам, как компьютерная зависимость, виртуализация окружающей среды, поверхностный характер мышления, снижение познавательной мотивации и активности и многое другое.]

⇒ *следовательно**:

- [необходимо существенно повысить уровень социальной ответственности у разработчиков компьютерных систем и соответствующих технологий.]
- [Опасность от безграмотного, социально безответственного и тем более злонамеренного внедрения интеллектуальных компьютерных систем нового поколения может иметь для человечества летальный характер.]

человеческое общество

⇒ *направления развития**:

[Если рассматривать *общество* как *многоагентную систему*, состоящую из самостоятельных интеллектуальных агентов, то, очевидно, что важнейшими факторами, определяющими повышение качества (уровня развития) *общества* являются:

- повышение эффективности использования опыта, накопленного *обществом*, эффективности использования человечеством *знаний* и *навыков*
- повышение темпов приобретения, накопления и систематизации эффективно используемым человечеством *знаний* и *навыков*.

Решение указанных проблем становится вполне возможным, если для этого использовать *интеллектуальные компьютерные системы нового поколения*, с помощью которых накапливаемые человечеством *знания* и *навыки* будут организованы как систематизированная распределенная библиотека многократно используемых информационных ресурсов (*знаний* и *навыков*).]

⇒ *следовательно**:

[Систематизация и автоматизация многократного использования накапливаемых человечеством информационных ресурсов требует их конвергенции, глубокой интеграции и формализации. Особое место в этом процессе занимает математика, как основа систематизации и формализации знаний и навыков на уровне формальных *онтологий верхнего уровня*.]

Заключение к Главе 7.1.

Благодаря тому, что *интеллектуальные компьютерные системы нового поколения* становятся самостоятельными и активными субъектами *человеческой деятельности* в достаточной степени равноправными людям (естественным индивидуальным субъектам человеческой деятельности), характер и, соответственно, уровень автоматизации *человеческой деятельности* существенно меняется — снимается необходимость управлять средствами автоматизации, поскольку такое "ручное" управление заменяется распределением обязанностей и ответственности между людьми и *интеллектуальными компьютерными системами нового поколения*.

Если автоматизация любого вида в любой области *человеческой деятельности* будет осуществляться с помощью *интеллектуальных компьютерных систем нового поколения* и если *интеллектуальные компьютерные системы нового поколения*, обеспечивающие автоматизацию разных видов и областей *человеческой деятельности*, будут содержательно взаимодействовать между собой, то общий уровень автоматизации *человеческой деятельности* существенно возрастет благодаря тому, что отпадет необходимость вручную координировать использование различных средств автоматизации.

Эффективность и трудоемкость автоматизации различных видов и областей *человеческой деятельности* будет существенно определяться степенью ***конвергенции*** между различными видами и областями *человеческой деятельности*. Необходимо построить иерархическую модель *человеческой деятельности,* в рамках которой должна быть проведена грамотная систематизация и стратификация всех видов и областей *человеческой деятельности*, направленная против излишнего эклектического многообразия. Таким образом, прежде, чем осуществлять комплексную автоматизации *человеческой деятельности* с помощью *интеллектуальных компьютерных систем нового поколения*, необходимо с позиций общей теории систем переосмыслить организацию этой деятельности. В противном случае автоматизация беспорядка приведет к еще большему беспорядку.

Особо подчеркнем то, что многие из рассмотренных нами проблем текущего состояния и направлений дальнейшего развития *Человеческой деятельности в области Искусственного интеллекта* аналогичны проблемам и тенденциям развития многих других научно-технических дисциплин. Следовательно, подходы к решению этих проблем могут носить междисциплинарный характер.

Время каждого человека является главным невосполнимым ресурсом общества и тратить его надо не на рутинную поддержку жизненного цикла всевозможных социально значимых объектов, а на комплексное развитие соответствующих *технологий*. Автоматизация человеческой деятельности с помощью глобальной системы интероперабельных семантически совместимых и активно взаимодействующих *интеллектуальных компьютерных систем* в самых разных областях *человеческой деятельности* позволит существенно сократить время каждого человека на выполнение рутинной, легко автоматизируемой деятельности. Человеческая деятельность должна стать ориентированной на максимально возможную самореализацию, раскрытие творческого потенциала каждого человека, направленного на ускорение темпов повышения уровня интеллекта всего общества.

Создание *Глобальной экосистемы интеллектуальных компьютерных систем нового поколения*, предполагает:

- Построение формальной модели *человеческой деятельности*;
- Переход от эклектичного построения сложных *интеллектуальных компьютерных систем*, использующих различные виды *знаний* и различные виды *моделей решения задач*, к их глубокой *интеграции* и *унификации*, когда одинаковые модели представления и модели обработки знаний реализуется в разных системах и подсистемах одинаково;
- Сокращение дистанции между современным уровнем *теории интеллектуальных компьютерных систем* и практики их разработки;
- Разработку грамотной тактики и стратегии переходного периода, в рамках которого современные *интеллектуальные компьютерные системы* должны постепенно заменяться на *интеллектуальные компьютерные системы нового поколения*, которые должны эффективно взаимодействовать не только между собой, но и с хорошо зарекомендовавшими себя современными информационными ресурсами и сервисами.

Глава 7.2.
Метасистема OSTIS

⇒ *авторы**:
- *Голенков В. В.*
- *Шункевич Д. В.*
- *Банцевич К. А.*
- *Загорский А. Г*

⇒ *аннотация**:
[Данная глава посвящена рассмотрению подхода к автоматизации процессов создания, развития и применения стандартов на основе *Технологии OSTIS*. Также в главе сформулированы основные принципы стандартизации *интеллектуальных компьютерных систем*, методов и средств их проектирования в рамках предлагаемого подхода.]

⇒ *подраздел**:
- *§ 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS*
- *§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS*

⇒ *ключевое понятие**:
- *Метасистема OSTIS*
- *Стандарт Технологии OSTIS*

⇒ *библиографическая ссылка**:
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Серенков П.С..КонцеИСкБЗ-2004ст*
- *Углев В.А.АктуаССПС-2012ст*
- *МетасOSTIS-2022эл*
- *Golenkov V.V..tStand oICSaaK-2018art*
- *Bantsevich K.A.Metas otOSTIS-2022art*

Введение в Главу 7.2.

В основе каждой развитой сферы человеческой деятельности лежит ряд стандартов, формально описывающих различные ее аспекты — систему понятий (включая терминологию), типологию и последовательность действий, выполняемых в процессе применения соответствующих методов и средств (см. *Golenkov V.V..Metho aTfECoC-2019art*).

Стандарты в самых различных областях являются важнейшим видом знаний, главной целью которых является обеспечение совместимости различных видов деятельности. Несмотря на развитие информационных технологий, в настоящее время подавляющее большинство стандартов представлено либо в виде традиционных линейных документов, либо в виде web-ресурсов содержащих набор статических страниц, связанных гиперссылками. Для того чтобы стандарты выполняли свою главную функцию, они должны постоянно совершенствоваться.

Текущее оформление стандартов имеет ряд недостатков, которые мешают эффективному и грамотному использованию стандартов в различных областях (см. *Серенков П.С..КонцеИСкБЗ-2004ст* и *Углев В.А.АктуаССПС-2012ст*):
- дублирование информации в рамках документа, описывающего стандарт;
- трудоемкость сопровождения самого стандарта, обусловленная в том числе дублированием информации, в частности, трудоемкость изменения терминологии;
- проблема интернационализации стандарта – фактически перевод стандарта на несколько языков приводит к необходимости поддержки и согласования независимых версий стандарта на разных языках;
- неудобство применения стандарта, в частности, трудоемкость поиска необходимой информации. Как следствие — трудоемкость изучения стандарта;
- несогласованность формы различных стандартов между собой, как следствие — трудоемкость автоматизации процессов развития и применения стандартов;

- трудоемкость автоматизации проверки соответствия объектов или процессов требования того или иного стандарта;
- и другие.

Перечисленные проблемы связаны в основном с формой представления стандартов.

Задачей любого стандарта в общем случае является описание согласованной системы понятий (и соответствующих терминов), бизнес-процессов, правил и других закономерностей, способов решения определенных классов задач и так далее. Для формального описания информации такого рода с успехом применяются онтологии. Более того, в настоящее время в ряде областей вместо разработки стандарта в виде традиционного документа разрабатывается соответствующая онтология. Такой подход дает очевидные преимущества в плане автоматизации процессов согласования и использования стандартов.

Однако, актуальной остается проблема, связанная не с формой, а с сутью (семантикой) стандартов — проблема несогласованности системы понятий и терминов между различными стандартами, которая актуальна даже для стандартов в рамках одной и той же сферы деятельности.

В настоящее время *Информатика* преодолевает важнейший этап своего развития — переход от информатики данных (data science) к информатике знаний (knowledge science), где акцентируется внимание на семантических аспектах представления и обработки *знаний*.

Без фундаментального анализа такого перехода невозможно решить многие проблемы, связанные с управлением *знаниями*, экономикой *знаний*, с *семантической совместимостью интеллектуальных компьютерных систем*.

С семантической точки зрения каждый стандарт есть иерархическая *онтология*, уточняющих структуру и систем понятий соответствующих им *предметных областей*, которая описывает структуру и функционирование либо некоторого класса технических или иных искусственных систем, либо некоторого класса организаций, либо некоторого вида деятельности.

Наиболее перспективным подходом к решению перечисленных проблем является преобразование каждого конкретного стандарта в *базу знаний*, в основе которой лежит набор *онтологий*, соответствующих данному стандарту. Такой подход позволяет в значительной мере автоматизировать процессы развития стандарта и его применения.

В рамках *Технологии OSTIS* данный подход используется при построении *Стандарта OSTIS*.

Предлагаемый *Стандарт OSTIS* оформлен в виде *семейства разделов базы знаний* специальной интеллектуальной компьютерной *Метасистемы OSTIS* (Intelligent MetaSystem for ostis-systems) (см. *МетасOSTIS-2022эл*), которая построена по *Технологии OSTIS* и представляет собой постоянно совершенствуемый интеллектуальный *портал научно-технических знаний*, который поддерживает перманентную эволюцию *Стандарта OSTIS*, а также разработку различных *ostis-систем* (интеллектуальных компьютерных систем, построенных по *Технологии OSTIS*).

§ 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS

⇒ *подраздел**:
- *Пункт 7.2.1.1. Структура Метасистемы OSTIS*
- *Пункт 7.2.1.2. Назначение Метасистемы OSTIS*
- *Пункт 7.2.1.3. Особенности Метасистемы OSTIS*
- *Пункт 7.2.1.4. Достоинства Метасистемы OSTIS*

⇒ *ключевое понятие**:
- *Метасистема OSTIS*

⇒ *библиографическая ссылка**:
- *МетасOSTIS-2022эл*

Эффективность любой технологии, в том числе и ***Технологии OSTIS*** определяется не только сроками создания искусственных систем соответствующего класса, но и темпами совершенствования самой технологии (темпами совершенствования средств автоматизации и темпами совершенствования системы стандартов, лежащих в основе технологии).

Для фиксации текущего состояния *Технологии OSTIS*, а также для организации ее эффективного использования и ее перманентного совершенствования с участием ученых, работающих в области искусственного интеллекта, и инженеров, разрабатывающих семантические компьютерные системы различного назначения, в состав *Экосистемы OSTIS* вводится *Метасистема OSTIS*, назначение которой делает ее ключевой *ostis-системой* в рамках *Экосистемы OSTIS*.

Метасистема OSTIS
:= [Intelligent MetaSystem for ostis-systems design]
:= [ostis-система автоматизации проектирования ostis-систем]
⇒ *примечание**:
[При разработке *Технологии OSTIS* средством автоматизации этой деятельности является не вся *Метасистема OSTIS*, а только ее часть — входящая в состав *Метасистемы OSTIS*, *Встраиваемая ostis-система поддержки реинжиниринга ostis-систем*, которая поддерживает деятельность разработчиков базы знаний *Метасистемы OSTIS*. Это обусловлено тем, что вся деятельность по разработке *Технологии OSTIS* сводится к разработке (инженирингу) и обновлению (совершенствованию, реинжинирингу) *Базы знаний Метасистемы OSTIS*).]

Метасистема OSTIS — интеллектуальная компьютерная система, обеспечивающая
- комплексную информационную поддержку всех этапов *жизненного цикла интеллектуальных компьютерных систем нового поколения*;
- автоматизацию проектирования всех компонентов *интеллектуальных компьютерных систем нового поколения*;
- комплексную автоматизацию всех этапов жизненного цикла *интеллектуальных компьютерных систем нового поколения*.

Метасистема OSTIS — метасистема, являющаяся:
- *корпоративной ostis-системой*, обеспечивающей организацию (координацию) деятельности *Консорциума OSTIS*;
- формой представления реализации и фиксации текущего состояния *Ядра Технологии OSTIS*;
- корпоративной *ostis-системой*, взаимодействующей со всеми корпоративными ostis-системами, каждая из которых координирует развитие соответствующей *специализированной ostis-технологии*.

Формой реализации представленной *Метасистемы OSTIS* является *Технология OSTIS*.

Метасистема OSTIS
:= [Универсальная базовая (предметно-независимая) ostis-система автоматизации проектирования ostis-систем (любых ostis-систем)]
∈ *ostis-система*
:= [Интеллектуальная метасистема, построенная по стандартам *Технологии OSTIS* и предназначенная (1) для инженеров *ostis-систем* – для поддержки проектирования. Реализации и обновления (реинжиниринга) *ostis-систем* и для разработчиков *Технологии OSTIS* – для поддержки коллективной деятельности по развитию стандартов и библиотек *Технологии OSTIS*.]
⇐ *форма реализации**:
Технология OSTIS
:= [Интеллектуальная Метасистема, являющаяся формой (вариантом) реализации (представления, оформления) *Технологии OSTIS* в виде *ostis-системы*]
⇒ *примечание**:
[Тот факт, что *Технология OSTIS* реализуется в виде *ostis-системы*, является весьма важным для эволюции *Технологии OSTIS*, поскольку методы и средства эволюции (перманентного совершенствования) *Технологии OSTIS* становятся фактически совпадающими с методами и средствами разработки любой (!) *ostis-системы* на всех этапах их жизненного цикла. Другими словами, эволюция *Технологии OSTIS* осуществляется методами и средствами самой этой технологии.]
:= [Система комплексной автоматизации (информационной и инструментальной поддержки) проектирования и реализации ostis-систем, которая сама реализована также в виде ostis-системы.]
:= [*Портал знаний по Технологии OSTIS*, интегрированный с с.а.п.р. ostis-систем и реализованный в виде ostis-системы.]
∈ *портал научно-технических знаний*

Пункт 7.2.1.1. Структура Метасистемы OSTIS

Принципы технической реализации *Метасистема OSTIS* полностью совпадают с принципами технической реализации прикладных интеллектуальных систем, разрабатываемых с помощью этой метасистемы.

Метасистема OSTIS
⇒ *декомпозиция**:
{• *полное описание самой Технологии OSTIS*
• *история эволюции Технологии OSTIS*

- *описание правил использования Технологии OSTIS*
- *описание организационной инфраструктуры, направленной на развитие Технологии OSTIS*
- *библиотека многократно используемых компонентов ostis-систем*
- *методы и инструментальные средства проектирования различного вида компонентов ostis-систем*
- *технические средства координации деятельности участников проекта, направленные на постоянное совершенствование Технологии OSTIS*

}

Метасистема OSTIS

⇒ *декомпозиция ostis-системы**:

{• *SC-модель Метасистемы OSTIS*

⇒ *декомпозиция sc-модели ostis-системы**:

{• *База знаний Метасистемы OSTIS*
• *Решатель задач Метасистемы OSTIS*
• *Пользовательский интерфейс Метасистемы OSTIS*
}

• *Программный вариант реализации ostis-платформы*
}

⇒ *подсистема**:

• *Библиотека Метасистемы OSTIS*
• *Средства разработки компонентов ostis-систем*

⇒ *декомпозиция**:

{• *Средства поддержки проектирования баз знаний ostis-систем*
• *Средства поддержки проектирования решателей задач ostis-систем*
• *Средства поддержки проектирования пользовательских интерфейсов ostis-систем*
}

База знаний Метасистемы OSTIS

⇒ *декомпозиция**:

{• *Стандарт OSTIS*
• *Раздел Проект OSTIS. История, текущее состояние и перспективы эволюции и применения Технологии OSTIS*
• *Документация Метасистемы OSTIS*
• *История и текущие процессы эксплуатации Метасистемы OSTIS*
• *Раздел Проект OSTIS. История, текущие процессы и план развития Метасистемы OSTIS*
}

Решатель задач Метасистемы OSTIS

⇒ *примечание**:

[Решатель задач собственно *Метасистемы OSTIS*, без учета подсистем, включает в себя набор sc-агентов информационного поиска, реализующих базовые механизмы навигации по базе знаний.]

⇒ *декомпозиция**:

{• *Абстрактный sc-агент поиска всех входящих константных позитивных стационарных sc-дуг принадлежности*
• *Абстрактный sc-агент поиска всех идентификаторов заданного sc-элемента*
• *Абстрактный sc-агент поиска полной семантической окрестности заданного элемента*
• *Абстрактный sc-агент поиска связок декомпозиции для заданного sc-элемента*
• *Абстрактный sc-агент поиска всех известных сущностей, являющихся общими по отношению к заданной*
• *Абстрактный sc-агент поиска определения или пояснения для заданного объекта*
• *Абстрактный sc-агент поиска всех известных сущностей, являющихся частными по отношению к заданной*
• *Абстрактный sc-агент поиска всех выходящих константных позитивных стационарных sc-дуг принадлежности с их ролевыми отношениями*
• *Абстрактный sc-агент поиска всех выходящих константных позитивных стационарных sc-дуг принадлежности*
• *Абстрактный sc-агент поиска всех входящих константных позитивных стационарных sc-дуг принадлежности с их ролевыми отношениями*
}

Пункт 7.2.1.2. Назначение Метасистемы OSTIS

Метасистема OSTIS является в *Экосистеме OSTIS* ключевой интеллектуальной системой, которая поддерживает не только проектирование новых интеллектуальных систем и не только замену устаревших компонентов в интеллектуальных системах, входящих в состав *Экосистемы OSTIS*, но и включение (интеграцию) в состав *Экосистемы OSTIS* новых создаваемых интеллектуальных систем.

Метасистема OSTIS ориентирована на разработку и практическое внедрение методов и средств компонентного проектирования семантически совместимых интеллектуальных систем, которая предоставляет возможность быстрого создания интеллектуальных приложений различного назначения. Подчеркнем при этом, что сферы практического применения методики компонентного проектирования семантически совместимых интеллектуальных систем ничем не ограничены.

Описываемая *Метасистема OSTIS* является:
- системой информационной и инструментальной поддержки всех этапов жизненного цикла и.к.с. нового поколения (*ostis-систем*) самого различного назначения;
- порталом знаний по *Технологии OSTIS*, обеспечивающим:
  - координацию работ по развитию *Технологии OSTIS*;
  - автоматизацию анализа качества *Стандарта OSTIS*.

  То есть *Метасистема OSTIS* является системой управления *Проектом создания и развития Стандарта OSTIS*.

Важнейшим направлением *Метасистемы OSTIS* и, соответственно, важнейшим направлением применения *Стандарта OSTIS*, подробно описанного в *§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS*, является использование их в качестве комплексного интегрированного компьютерного учебного пособия по специальности “Искуственный интеллект”. Для этого устанавливается связь между разделами *Стандарта OSTIS* и программами различных *учебных дисциплин* указанной специальности. Важно подчеркнуть при этом: *Стандарт OSTIS* содержит достаточно полный сравнительный анализ с различными альтернативными подходами, то есть ни в коем случае не ограничивается рассмотрением только *Технологией OSTIS*.

Пункт 7.2.1.3. Особенности Метасистемы OSTIS

Особенность *Метасистемы OSTIS* заключается в унификации представления различного вида информации в памяти компьютерных систем на основе смыслового (семантического) представления этой информации, что обеспечивает:
- устранение дублирования одной и той же информации в разных интеллектуальных системах и в разных компонентах одной и той же системы;
- семантическую совместимость различных компонентов интеллектуальных систем и различных *интеллектуальных систем* в целом;
- существенное расширение *библиотек многократно используемых компонентов ostis-систем* за счет "крупных" компонентов и, в частности, *встраиваемых ostis-систем*.

Пункт 7.2.1.4. Достоинства Метасистемы OSTIS

Метасистема OSTIS взаимодействует не только со своими разработчиками и конечными пользователями, но и с другими ostis-системами, которые созданы с помощью *Технологии OSTIS* и представляют собой ее дочерние системы*.

Метасистема OSTIS для своих дочерних систем может:
- осуществлять автоматическую сборку *дочерних ostis-систем* стартовых версий по инструкциям. Таким образом, генерировать новые *дочерние ostis-системы*;
- включать в *дочерние ostis-системы* новые многократно используемые компоненты из постоянно пополняемой *библиотеки многократно используемых семантически совместимых компонентов ostis-систем*;
- заменять в *дочерних ostis-системах* устаревшие версии многократно используемых компонентов на новые версии из *Библиотеки Метасистемы OSTIS*;
- включать в *дочерние ostis-системы* подсистему совершенствования своей расширенной базы знаний и, при необходимости, подсистему улучшения ее интегрированной машины обработки знаний и пользовательского интерфейса.

Таким образом, после появления *дочерней ostis-системы* ее связь с *Метасистемой OSTIS* не прерывается и она становится постоянным участником процесса совершенствования всех *дочерних ostis-систем*.

Метасистема OSTIS является одновременно и системой автоматизации проектирования *ostis-систем*, и интеллектуальной системой, обучающей методам и средствам проектирования *ostis-систем*. Этот факт существенно повышает качество проектирования прикладных *ostis-систем*, расширяет контингент разработчиков *ostis-систем* и интегрирует проектную (инженерную) деятельность в области искусственного интеллекта с образовательной деятельностью в этой области.

Стоит отметить также, что все опубликованные материалы о *Технологии OSTIS* в формализованном виде входят в базу знаний *Метасистемы OSTIS*.

§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS

⇒ *подраздел**:
- *Пункт 7.2.2.1. Оглавление Стандарта OSTIS*
- *Пункт 7.2.2.2. Ключевые знаки Стандарта OSTIS*
- *Пункт 7.2.2.3. Назначение Стандарта OSTIS*
- *Пункт 7.2.2.4. Аналоги Стандарта OSTIS*
- *Пункт 7.2.2.5. Особенности Стандарта OSTIS*
- *Пункт 7.2.2.6. Пользователи Стандарта OSTIS*
- *Пункт 7.2.2.7. Авторский коллектив Стандарта OSTIS*
- *Пункт 7.2.2.8. Требования, предъявляемые к Стандарту OSTIS*
- *Пункт 7.2.2.9. Правила построения Стандарта OSTIS*
- *Пункт 7.2.2.10. Направления развития Стандарта OSTIS*
- *Пункт 7.2.2.11. Достоинства Стандарта OSTIS*

⇒ *ключевое понятие**:
- *Стандарт Технологии OSTIS*

⇒ *библиографическая ссылка**:
- *Голенков В.В..СтандОТОП-2021кн*
- *Taberko V..OntolAfSDwI-2020art*

Стандарт Технологии OSTIS представляет собой Документацию *Технологии OSTIS*, которая представлена в виде *основной части базы знаний* специальной *интеллектуальной компьютерной системы*, предназначенной для комплексной поддержки жизненного цикла семантически совместимых *интеллектуальных компьютерных систем нового поколения* (*Метасистемы OSTIS*).

Стандарт OSTIS
:= [Документация *Технологии OSTIS*]
:= [Документация Открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных *интеллектуальных компьютерных систем*]
:= [Описание *Технологии OSTIS* (Open Semantic Technology for Intelligent Systems), представленная в виде семейства разделов *базы знаний специальной ostis-системы* (системы, построенной по *Технологии OSTIS*) на внутреннем языке *ostis-систем* и обладающее достаточной полнотой для использования этой *технологии* разработчиками *интеллектуальных компьютерных систем*]
:= [Полное описание текущего состояния *Технологии OSTIS*, представленное в виде семейства разделов *базы знаний*, построенной по *Технологии OSTIS*]
:= [Семейство разделов *базы знаний Метасистемы OSTIS*, которое предназначено для комплексной поддержки онтологического проектирования семантически совместимых *гибридных интеллектуальных компьютерных систем*]
∈ *семейство разделов базы знаний*
  := [семейство разделов внутреннего представления *базы знаний ostis-системы – интеллектуальной компьютерной системы*, построенной по *Технологии OSTIS*]
:= [Достаточно полная формальная Документация текущей версии *Технологии OSTIS*, представленная либо в виде основной части базы знаний *Метасистемы OSTIS*, либо в виде внешнего формального представления этой *базы знаний*]
:= [Основная часть базы знаний *Метасистемы OSTIS*, описывающая текущую версию *Технологии OSTIS*]
:= [Формальный текст, объектом описания которого является *Технология OSTIS*, то есть текст, являющий достаточно полным описанием текущего состояния *Технологии OSTIS*]
:=

[Документация *Технологии OSTIS*, полностью отражающая текущее состояние *Технологии OSTIS* и представленная соответствующим *семейством разделов базы знаний* специальной *ostis-системы*, которая ориентирована на поддержку проектирования, производства, эксплуатации и эволюции (реинжиниринга) *ostis-систем*, а также на поддержку эволюции самой *Технологии OSTIS* и которая названа нами *Метасистемой OSTIS*]

:= [Семейство разделов, в состав которого входят все *разделы Стандарта OSTIS*]

⇒ *основной sc-идентификатор**:

[Стандарт OSTIS]

⇐ *сокращение**:

[Стандарт *Технологии OSTIS*]

⇐ *сокращение**:

[Стандарт Открытой *технологии* комплексной поддержки *жизненного цикла* семантически совместимых *интеллектуальных компьютерных систем нового поколения*]

Следует подчеркнуть, что *Стандарт OSTIS* — это не описание некоторого состояния *Технологии OSTIS*, а динамическая информационная модель процесса эволюции этой *технологии* (см. *Главу 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*).

В рамках *Стандарта OSTIS* вводится оглавление, система ключевых знаков. Также строго регламентированы:

- требования, предъявляемые к *Стандарту OSTIS*;
- правила построения *Стандарта OSTIS*;
- направления развития *Стандарта OSTIS*.

Это позволяет воспринимать пользователю *Стандарт OSTIS*, как целостный понятный текст. Также избежать возможных противоречий.

Пункт 7.2.2.1. Оглавление Стандарта OSTIS

Одним из составляющих *Стандарта OSTIS* является *Оглавление Стандарта OSTIS*.

Оглавление Стандарта OSTIS

:= *пояснение**:

[Иерархический перечень разделов, входящих в состав *Стандарта OSTIS*, с дополнительной спецификацией некоторых разделов]

⇒ *примечание**:

[Существенно подчеркнуть, что иерархия разделов *Стандарта OSTIS* не означает то, что *разделы* более низкого уровня иерархии входят в состав (являются частями) соответствующих разделов более высокого уровня. Связь между *разделами* разных уровней иерархии означает то, что *раздел* более низкого уровня иерархии является *дочерним* разделом по отношению к соответствующему *разделу* более высокого уровня, то есть *разделом*, который наследует свойства указанного *раздела* более высокого уровня.

В отличие от этого каждая *часть Стандарта OSTIS*, а также сам *Стандарт OSTIS* является *семейством разделов* (совокупность разделов), входящих в ее состав.]

⇒ *примечание**:

[Описание логико-семантических связей каждого раздела *Стандарта OSTIS* с другими разделами *Стандарта OSTIS* приводится в рамках *титульной спецификации* каждого *раздела*.]

Основной текст *Стандарта OSTIS* состоит из следующих частей:

Стандарт OSTIS

⇒ *декомпозиция**:

Структура Стандарта OSTIS верхнего уровня

= ⟨• *Часть 1 Стандарта OSTIS.*

Введение в интеллектуальные компьютерные системы нового поколения

:= [Анализ текущего состояния *технологий Искусственного интеллекта* и постановка задачи на создание комплекса совместимых *технологий искусственного интеллекта*, обеспечивающего поддержку всего *жизненного цикла интеллектуальных компьютерных систем нового поколения* и названного нами *Технологией OSTIS*]

• *Собственно документация Технологии OSTIS*

⇒ *декомпозиция**:

⟨• *Стандарт ostis-систем*

:=

[Стандарт интеллектуальных компьютерных систем нового поколения, построенных по Технологии OSTIS]

:= [Формальная теория ostis-систем]

:= [Формальные структурно-функциональные и логико-семантические модели ostis-систем]

⇒ *декомпозиция**:

⟨• *Часть 2 Стандарта OSTIS.*

Смысловое представление и онтологическая систематизация знаний в и.к.с. нового поколения.

:= [Стандарт представления информации в ostis-системах]

:= [Модели представления знаний и баз знаний в ostis-системах]

• *Часть 3 Стандарта OSTIS.*

Многоагентные решатели задач и.к.с. нового поколения

:= [Стандарт процессов и методов обработки информации в ostis-системах]

:= [Модели обработки знаний в ostis-системах (логические, продукционные, функциональные, нейросетевые, процедурные и непроцедурные, четкие и нечеткие)]

• *Часть 4 Стандарта OSTIS.*

Онтологические модели интерфейсов и.к.с. нового поколения

:= [Стандарт информационных ресурсов и моделей решения интерфейсных задач в ostis-системах]

⟩

• *Стандарт методов и средств поддержки жизненного цикла ostis-систем*

:= [Стандарт бизнес-процессов и методик, автоматически реализуемых процессов и методов, информационных средств и инструментальных средств, используемых для поддержки жизненного цикла ostis-систем]

⇒ *декомпозиция**:

⟨• *Часть 5 Стандарта OSTIS.*

Методы и средства проектирования и.к.с. нового поколения

:= [Методики, методы и средства проектирования баз знаний, решателей задач и интерфейсов ostis-систем]

• *Часть 6 Стандарта OSTIS.*

Платформы реализации и.к.с. нового поколения

:= [Методы и средства реализации ostis-систем (на основе программных ostis-платформ и специально созданных для этого компьютеров)]

• *Часть 7 Стандарта OSTIS.*

Методы и средства реинжиниринга и эксплуатации и.к.с. нового поколения

:= [Методы и средства эксплуатации ostis-систем конечными пользователями, а также их сопровождения (поддержки работоспособности) и реинжиниринга (обновления, модернизации)]

⟩

⟩

• *Часть 8 Стандарта OSTIS.*

Экосистема и.к.с. нового поколения и их пользователей

:= [Описание продуктов, создаваемых с помощью Технологии OSTIS, основными их которых является *Экосистема OSTIS*, семантически совместимых и активно взаимодействующих ostis-систем и их пользователей]

:= [Теория Экосистемы OSTIS и ее эволюции]

• *Библиография OSTIS*

:= [Спецификация *библиографических источников*, семантически близких *Технологии OSTIS*, в контексте их сравнительного анализа со *Стандартом OSTIS*]

⟩

Пункт 7.2.2.2. Ключевые знаки Стандарта OSTIS

Система ключевых знаков *Стандарта OSTIS* упорядочена в точном соответствии с *Оглавлением Стандарта OSTIS* и является уточнением указанного Оглавления путем перечисления и пояснения ключевых сущностей, описываемых в разделах Стандарта, и, в первую очередь тех сущностей, которые указываются в идентификаторах (названиях) разделов *Стандарта OSTIS*.

Система ключевых знаков Стандарта OSTIS является целостным дополнением к *Оглавлению Стандарта OSTIS*, поскольку:

- иерархия и последовательность ключевых знаков четко соответствуют иерархии и последовательности разделов стандарта;
- система ключевых знаков *Стандарта OSTIS*, как и его Оглавление, воспринимается (читаться) как целостный понятный текст.

Пункт 7.2.2.3. Назначение Стандарта OSTIS

Поскольку *Стандарт OSTIS* является неотъемлемой частью *Метасистемы OSTIS* (основной частью ее *базы знаний*), основным назначением *Стандарта OSTIS* является обеспечение максимально эффективной реализации того, для чего предназначена *Метасистема OSTIS*.

Стандарт OSTIS рассматривается как результат конвергенции и интеграции всевозможных направлений *Искусственного интеллекта*, что позволяет студентам и магистрантам сформировать целостное представление о тематике *Искусственного интеллекта*, а не мозаичное представление в виде множества дисциплин (направлений), связи между которыми подробно и тем более формально не рассматриваются.

Стандарт OSTIS перманентно и достаточно быстро эволюционирует. За время обучения студентов и магистрантов происходит весьма существенные изменения текущей версии *Стандарта OSTIS*.

Студенты и магистранты активно вовлекаются в процесс эволюции *Стандарта OSTIS*, это обеспечивает:

- формирование необходимого уровня их квалификации в условиях быстрого морального старения того, чему их уже научили;
- формирование необходимых навыков, позволяющих им в процессе реальной профессиональной деятельности быстро адаптироваться к новым условиям этой деятельности и, в частности, к новым версиям соответствующих технологий.

Пункт 7.2.2.4. Аналоги Стандарта OSTIS

Аналогами *Стандарта OSTIS* можно считать:

- любую серьезную попытку систематизации результатов, полученных в области Искусственного интеллекта к текущему моменту:
  - учебник, достаточно полно отражающий текущее состояние *Искусственного интеллекта*;
  - справочник, содержащий достаточно полную информацию о текущем состоянии *Искусственного интеллекта*.
- любую попытку перехода от частных формальных моделей различных многократно используемых компонентов ostis-систем к общей (объединенной, интегрированной) формальной модели и.к.с в целом — теории и.к.с.
- любую унификацию технических решений, устранения многообразия форм технических решений при разработке и.к.с.
- первые попытки разработки стандартов *интеллектуальных компьютерных систем*, а также *технологий Искусственного интеллекта*, которые, чаще, всего, ограничиваются построением систем соответствующих понятий.

Пункт 7.2.2.5. Особенности Стандарта OSTIS

Стандарт OSTIS — это не просто систематизация современного состояния результатов в области *Искусственного интеллекта*, это систематизация, представленная в виде общей комплексной формальной модели *интеллектуальных компьютерных систем* и комплексной формальной модели поддержки их жизненного цикла. Более того, текст *Стандарта OSTIS* представляет собой *основную часть базы знаний* специальной *интеллектуальной метасистемы*, которая ориентирована:

- на поддержку разработки *интеллектуальных компьютерных систем* различного назначения;
- на поддержку эволюции *Стандарта OSTIS*;
- на поддержку *подготовки специалистов в области Искусственного интеллекта*.

Стандарт OSTIS — это динамический текст, перманентно отражающий новые научно-технические результаты, получаемые в области *Искусственного интеллекта* в рамках *Общей теории интеллектуальных компьютерных*

систем и *Общей комплексной технологии разработки интеллектуальных компьютерных систем*. Здесь важной является оперативность фиксации новых научно-технических результатов, то есть минимизация отрезка времени между моментом получения новых результатов и моментом интеграции описания этих результатов в состав *Стандарта OSTIS*. В перспективе авторы новых научно-технических результатов в области *Искусственного интеллекта* будут заинтересованы лично публиковать (интегрировать) свои результаты в состав *Стандарта OSTIS*, то есть становиться соавторами *Стандарта OSTIS*, чтобы обеспечить необходимую оперативность такой публикации и отсутствие искажений своих результатов. Динамичность *Стандарта OSTIS* и достаточная оперативность интеграции в его состав новых научно-технических результатов в области *Искусственного интеллекта* делает *Стандарт OSTIS* всегда актуальным и никогда морально устаревшим.

В рамках *Стандарта OSTIS* нет противопоставления между научно-технической информацией, добываемой в области *Искусственного интеллекта*, и учебно-методической информацией, используемой для подготовки и самоподготовки специалистов в области Искусственного интеллекта. информация о том, чему учить, должна быть "переплетена", интегрирована с информацией о том, как учить.

Важно заметить, что *Стандарт OSTIS* в отличие от остальных стандартов является структурированным формальным текстом, который может быть непосредственно использован не только разработчиками *интеллектуальных компьютерных систем*, но также и интеллектуальными компьютерными системами, осуществляющими автоматизацию проектирования разрабатываемых *интеллектуальных компьютерных систем* и поддержку последующих этапов их жизненного цикла. Таким образом, разработка *Стандарта OSTIS* является неотъемлемой частью разработки комплекса средств информационной и инструментальной поддержки всего жизненного цикла *интеллектуальных компьютерных систем*, а указанные средства поддержки *жизненного цикла интеллектуальных компьютерных систем* становятся равноправными партнерами в процессе создания, эксплуатации и сопровождения *интеллектуальных компьютерных систем* благодаря своему осознанию (пониманию) того, что такое *интеллектуальные компьютерные системы* и их *жизненный цикл*.

Содержательно *Стандарт OSTIS* охватывает не только описание моделей разрабатываемых *интеллектуальных компьютерных систем*, но также и описание методик, автоматизируемых методов и инструментальных средств поддержки (автоматизации) всех этапов жизненного цикла разрабатываемых интеллектуальных компьютерных систем.

Проект развития Стандарта OSTIS ориентирован на высокие темпы эволюции *Стандарта OSTIS* благодаря автоматизации управления этим проектом с помощью *Метасистемы OSTIS*, которая является полноправным участником этого проекта.

Построение и структуризация текста *Стандарта OSTIS* ориентированы на максимально возможное снижение языкового и когнитивного барьера для начинающих его пользователей. Для этой цели используются (1) различного рода естественно-языковые примечания и комментарии, имеющие соответствующие семантические связи с поясняемыми сущностями, а также (2) различного рода дидактические знания, указывающие на различные аналогии, различия, примеры, принципы, лежащие в основе описываемых сущностей и тому подобные.

Пункт 7.2.2.6. Пользователи Стандарта OSTIS

Рассмотрим целевую аудиторию *Стандарта OSTIS*.

Стандарт OSTIS
⇒ *класс пользователей**:
пользователь Стандарта OSTIS
≔ [целевая аудитория Стандарта OSTIS]
⇒ *разбиение**:
{• *разработчик ostis-системы*
⊃ *разработчик Метасистемы OSTIS*
⇒ *разбиение**:
{• *разработчик Стандарта OSTIS*
• *разработчик решателя задач Метасистемы OSTIS*
• *разработчик пользовательского интерфейса Метасистемы OSTIS*
}
⊃ *разработчик базы знаний ostis-системы*
⊃ *разработчик Стандарта OSTIS*
⊃ *разработчик решателя задач ostis-системы*
⊃ *разработчик решателя задач Метасистемы OSTIS*
⊃ *разработчик интерфейса ostis-системы*
⊃ *разработчик пользовательского интерфейса Метасистемы OSTIS*

⊃ *разработчик ostis-платформы*
- *потенциальный разработчик ostis-системы*
- *специалист в области Искусственного интеллекта, желающий интегрировать свои результаты в состав общей теории и.к.с. нового поколения и соответствующей комплексной технологии*
- *студент или магистрант специальности "Искусственный интеллект" либо другой смежной специальности, желающий приобрести практический опыт в разработке прикладных и.к.с. нового поколения или в разработке соответствующей комплексной технологии*

}

Пункт 7.2.2.7. Авторский коллектив Стандарта OSTIS

Для обеспечения перманентной эволюции существует ряд требований, ориентированный на авторов *Стандарта OSTIS*.

Авторы *Стандарта OSTIS* должны:
- Отслеживать и изучать новые публикации по тематике, рассматриваемой в Стандарте OSTIS. Близкими источниками для этого являются:
  - выпуски журналов;
  - материалы конференций:
    - организуемых Консорциумом 3WC;
    - по интеграции различных направлений *Искусственного интеллекта*;
  - стандарты в области *Искусственного интеллекта*;
  - публикации, рассматривающие:
    - формальные онтологии;
    - онтологии верхнего уровня;
    - семантические сети;
    - графы знаний;
    - графовые базы данных и графовые с.у.б.д.;
    - смысловое представление знаний;
    - конвергенцию различных направлений *Искусственного интеллекта*.
- Фиксировать результаты изучения новых публикаций по тематике, близкой *Стандарту OSTIS*, в *Библиографии OSTIS*, а также в основном тексте *Стандарта OSTIS* в виде соответствующих ссылок, цитат, сравнительного анализа.
- Отслеживать текущее состояние всего текста *Стандарта OSTIS*, формировать предложения, направленные на развитие *Стандарта OSTIS* и на повышение темпов этого развития. Активно участвовать в обсуждении проблем развития *Технологии OSTIS*.
- Максимально возможным образом увязывать персональную работу над *Стандартом OSTIS* с другими формами деятельности – научной, учебной, прикладной.
- Указывать авторство своих предложений по дополнению и/или корректировке текущего текста *Стандарта OSTIS*.
- Участвовать в рецензировании и согласовании предложений, представленных другими авторами *Стандарта OSTIS*.

Большой объем работ по созданию и развитию *Стандарта OSTIS* и, соответственно, *Технологии OSTIS*, комплексный характер этих работ, в которых необходима глубокая *конвергенции* и *интеграции* различных направлений *Искусственного интеллекта* предъявляют к *Авторскому коллективу Стандарта OSTIS* высокие требования по уровню мотивации, по уровню качества творческой атмосферы, по уровню *интероперабельности* всех членов коллектива, то есть по уровню способности быстро и качественно согласовывать персональные точки зрения.

Поскольку *Проект создания и развития Стандарта OSTIS* является открытым, членом *Авторского коллектива Стандарта OSTIS* может стать любой желающий, соблюдающий Правила организации взаимодействия членов *Авторского коллектива Стандарта OSTIS*, разделяющий цели и задачи разработки такого стандарта.

Выделяются следующие ключевые пункты *Правил организации взаимодействия членов Авторского коллектива Стандарта OSTIS*:
- Коллегиально формировать тактические и стратегические направления развития Стандарта OSTIS и, соответственно, Технологии OSTIS;
- Коллегиально распределять задачи по реализации утвержденных направлений развития Стандарта OSTIS с учетом (1) научных интересов, квалификации и возможности каждого члена Авторского коллектива, (2) приоритета задач и при достаточно полном охвате всех приоритетных задач.

В рамках *Авторского коллектива Стандарта OSTIS* выделяется также ***Редакционная коллегия Стандарта OSTIS***.

Редакционная коллегия Стандарта OSTIS — часть *Авторского коллектива Стандарта OSTIS*, являющаяся центром коллегиального принятия решений по основным направлениям развития Стандарта OSTIS и, соответственно, Технологии OSTIS, по уточнению соответствующих приоритетов и сроков. *Редакционная коллегия Стандарта OSTIS* также несет ответственность за формирование и реализацию стратегических направлений развития Стандарта OSTIS и, в частности, за подбор и назначение *ответственных исполнителей разделов Стандарта OSTIS*.

Основными направлениями деятельности *Редакционной коллегии Стандарта OSTIS* являются:
- Обеспечение целостности и повышения качества постоянно развиваемой (совершенствуемой) Технологии OSTIS, а также достаточно точное описание (документирование) каждой текущей версии этой технологии.
- Обеспечение четкого контроля совместимости версий Технологии OSTIS в целом, а также версий различных компонентов этой технологии.
- Постоянное уточнение степени важности различных направлений развития Технологии OSTIS для каждого текущего момента.
- Формирование и постоянное уточнение плана тактического и стратегического развития самой *Технологии OSTIS*, а также полной документации этой Технологии в виде *Стандарта OSTIS*. Подчеркнем при этом, что указанная документация является неотъемлемой частью *Технологии OSTIS*.

Стандарт OSTIS
⇒ *редакционная коллегия**:
Редакционная коллегия Стандарта OSTIS
≔ [Редколлегия Стандарта OSTIS]
≔ [Рабочий орган, обеспечивающий организацию коллективного творческого процесса по развитию *Стандарта OSTIS*, совмещенного с учебно-методическим обеспечением подготовки соответствующих специалистов.]
≔ [Редакционная коллегия, несущая ответственность за качество перманентно эволюционируемого *Стандарта OSTIS*.]
⇒ *обязанности**:
- [Обеспечение развития *Стандарта OSTIS*, совмещенного (интегрированного) с комплексным учебно-методическим обеспечением *подготовки специалистов в области Искусственного интеллекта*]
- [Обеспечение корректности (непротиворечивости), системности, целостности, полноты всех разрабатываемых материалов *Стандарта OSTIS* и, в том числе, *учебно-методического обеспечения подготовки специалистов в области Искусственного интеллекта*.]
- [Кординация деятельности авторов разработки очередной (следующей) версии *Стандарта OSTIS*.]
- [В частности, *Редколлегия Стандарта OSTIS* может осуществлять распределение работ по построению следующей версии *Стандарта OSTIS* с четкой привязкой ответственных лиц *Стандарта OSTIS* к соответствующим разделам *Стандарта OSTIS*.]

⇒ *примечание**:
[Очень важная структура, определяющая научно-технический уровень, авторитет и репутацию всей *Технологии OSTIS* в глазах международной научно-технической общественности.]
⇒ *примечание**:
[Каждый член *Редакционной коллегии Стандарта OSTIS* должен быть активным членом *Авторского коллектива Стандарта OSTIS*.]

Пункт 7.2.2.8. Требования, предъявляемые к Стандарту OSTIS

Стандарт OSTIS должен соответствовать следующим требованиям:
- вводимые понятия (в т.ч. дидактические отношения) должны быть четко пояснены и/или определены в соответствующем разделе Стандарта OSTIS;
- доступность понимания текста *Стандарта OSTIS* для читателей;
- обеспечение возможности поэтапной формализации информации, начиная от ея-текстов, которые могут быть в последствии записаны на формальном языке;
- независимость от естественных языков (только на уровне внешних идентификаторов (имен) sc-элементов);
- четкая логико-семантическая спецификация каждой предметной области, рассматриваемой в Стандарте OSTIS. Названная спецификация должна отражать как внутреннюю структуру предметной области (роли ее ключевых элементов), так и связи с другими предметными областями;

- конвергенция, ("бесшовная") интеграция различных видов *знаний*, описывающих самые различные сущности, к числу которых, в частности относятся и сами знания всевозможного вида;
- целостность, полнота, связность:
  - отсутствие информационных дыр;
  - достаточно полная спецификация всех сущностей;
  - согласованность основных идентификаторов (терминов), отсутствие синонимов и омонимов.
- отсутствие информационных излишеств и информационного мусора;
- четкая семантическая стратификация – каждый фрагмент базы знаний должен иметь свою семантическую "полочку" (никакого дублирования);
- строгая логическая последовательность текста (все используемые сущности должны быть введены либо в заданной предметной области, либо в предметной области более высокого уровня);
- унификация стилистики – текст не должен вызывать трудностей для его понимания;
- богатая библиография и сравнительный анализ;
- четкое соблюдение и совершенствование правил идентификации и спецификации описываемых сущностей;
- достаточно подробная спецификация каждого вводимого понятия в соответствующей предметной области.

Пункт 7.2.2.9. Правила построения Стандарта OSTIS

В рамках разработки *Стандарта OSTIS* выделяются *Общие правила построения Стандарта OSTIS* и *Частные правила построения Стандарта OSTIS*.

Общие правила построения Стандарта OSTIS
:= [принципы, лежащие в основе структуризации и оформления Стандарта OSTIS]

Рассмотрим основные положения:
- Основной формой представления *Стандарта OSTIS* как полной документации текущего состояния *Технологии OSTIS* является *внутреннее представление* основной части *базы знаний* специальной интеллектуальной компьютерной *Метасистемы OSTIS*, обеспечивающей использование и эволюцию (перманентное совершенствование) *Технологии OSTIS*. Такое представление *Стандарта OSTIS* обеспечивает эффективную семантическую навигацию по содержанию *Стандарта OSTIS* и возможность задавать *Метасистеме OSTIS* широкий спектр нетривиальных вопросов о самых различных деталях и тонкостях *Технологии OSTIS*;
- Кроме представления *Стандарта OSTIS* на внутреннем *языке представления знаний* используется также внешняя форма представления *Стандарта OSTIS* на *внешнем языке представления знаний*. При этом указанное внешнее представление *Стандарта OSTIS* должно быть структурировано и оформлено так,чтобы читатель мог достаточно легко "вручную" найти в этом тексте практически любую интересующую его *информацию*. В качестве *формального языка* внешнего представления *Стандарта OSTIS* используется *SCn-код*;
- *Стандарт OSTIS* имеет онтологическую структуризацию, то есть представляет собой иерархическую систему связанных между собой *формальных предметных областей* и соответствующих им *формальных онтологий*. Благодаря этому обеспечивается высокий уровень стратифицированности *Стандарта OSTIS*;
- Каждому *понятию*, используемому в *Стандарте OSTIS*, соответствует свое место в рамках этого Стандарта, своя *предметная область* и соответствующая ей *онтология*, где это *понятие* подробно рассматривается (исследуется), где концентрируется вся основная информация об этом *понятии*, о различных его свойствах.
- В состав *Стандарта OSTIS* входят также файлы информационных конструкций, не являющихся конструкциями *SC-кода* (в том числе и sc-текстов, принадлежащих различным естественным языкам). Такие файлы позволяют формально описывать в базе знаний синтаксис и семантику различных внешних языков, а также позволяют включать в состав базы знаний различного рода пояснения, примечания, адресуемые непосредственно пользователям и помогающие им в понимании формального текста базы знаний;
- С семантической точки зрения *Стандарт OSTIS* представляет собой иерархическую систему формальных моделей *предметных областей* и соответствующих им *формальных онтологий*;
- С семантической точки зрения *Стандарт OSTIS* представляет собой большую *рафинированную семантическую сеть*, которая, соответственно, имеет нелинейный характер и которая включает в себя знаки любых видов описываемых сущностей(материальных сущностей, абстрактных сущностей, понятий, связей, структур) и, соответственно этому, содержит связи между всеми этими видами сущностей(в частности, связи между связями, связи между структурами);
- *Стандарт OSTIS* представляет собой иерархическую систему *предметных областей* и соответствующих им *онтологий*, специфицирующих эти *предметные области*. Каждая из *предметных областей* описывает соответствующие *классы объектов исследования* с максимально возможной степенью детализации, определяемой набором *отношений* и *параметров*, заданных на *классах объектов исследования*. На множестве *предметных*

областей, задано отношение *дочерняя предметная область**, которое указывает направление наследования свойств объектов исследования, рассматриваемых в разных *предметных областях*;
- Каждый *раздел Стандарта OSTIS* может содержать те *знания*, которые входят в состав той *предметной области и онтологии*, которая либо полностью представлена указанным *разделом*, либо представлена частично в виде спецификации одного или нескольких конкретных объектов исследования;
- недопустима синонимия и омонимия основных sc-идентификаторов в рамках каждого семейства;
- Спецификация каждой предметной области и каждого раздела должна иметь достаточную степень полноты. Как минимум, для каждой предметной области должна быть указана роль каждого используемого в ней понятия;
- В состав *Стандарта OSTIS* входят также файлы информационных конструкций, не являющихся конструкциями *SC-кода* (в том числе и sc-текстов, принадлежащих различным естественным языкам). Такие файлы позволяют формально описывать в базе знаний синтаксис и семантику различных внешних языков, а также позволяют включать в состав базы знаний различного рода пояснения, примечания, адресуемые непосредственно пользователям и помогающие им в понимании формального текста базы знаний;
- Непосредственно сам *Стандарт OSTIS* представляет собой внутреннее *смысловое представление* основной части базы знаний *Метасистемы OSTIS* на внутреннем смысловом языке *ostis-систем* (этот язык назван нами *SC-кодом* - Semantic Computer Code).

Кроме *Общих правил построения Стандарта OSTIS* в *Стандарте OSTIS* приводятся описания различных частных (специализированных) правил построения (оформления) различных видов фрагментов *Стандарта OSTIS*.

К таким видам фрагментов относятся следующие:
- *sc-идентификатор*
  - := [внешний идентификатор внутреннего знака (*sc-элемента*) входящего в состав *базы знаний ostis-системы*]
  - := [*информационная конструкция* (чаще всего это строка символов), обеспечивающая однозначную идентификацию соответствующей сущности, описываемой в *базах знаний ostis-систем*, и являющаяся, чаще всего, именем (термином), соответствующим описываемой сущности, именем, обозначающим эту сущность во внешних текстах *ostis-систем*]
- *sc-спецификация*
  - := [семантическая окрестность]
  - := [семантическая окрестность соответствующего *sc-элемента* (внутреннего знака, хранимого в памяти *ostis-системы* в составе ее *базы знаний*, представленной на внутреннем языке *ostis-систем.*)]
  - := [семантическая окрестность некоторого *sc-элемента*, хранимого в *sc-памяти*, в рамках текущего состояния этой *sc-памяти*]
- *sc-конструкция* \ *sc-спецификация*
  - := [*sc-конструкция* (конструкция *SC-кода* – внутреннего языка *ostis-систем*), не являющаяся *sc-спецификацией*]
- *файл ostis-системы* \ *sc-идентификатор*
  - := [*файл ostis-системы*, не являющийся sc-идентификатором]

Важно также отметить, что к числу частных правил построения *sc-конструкций* относятся *Правила построения баз знаний ostis-систем*. Эти правила направлены на обеспечение целостности баз знаний ostis-систем, на обеспечение (1) востребованности (нужности) знаний, входящих в состав каждой базы знаний, и (2) целостности самой базы знаний, то есть достаточности знаний, входящих в состав каждой базы знаний для эффективного функционирования соответствующей ostis-системы.

Пункт 7.2.2.10. Направления развития Стандарта OSTIS

Стандарт OSTIS
- ⇒ *общие направления развития**:
  - {• [Включить в Стандарт OSTIS достаточно подробные правила построения (оформления) sc-идентификаторов и sc-спецификаций различного вида сущностей, а также различного вида файлов ostis-систем]
  - [На каждом этапе четко распределять работу по развитию различных разделов Стандарта OSTIS]
  - [Все инструментальные средства, входящие в состав Технологии OSTIS, должны быть достаточно детально описаны (специфицированы) в виде соответствующих онтологических моделей, имеющих четкую семантическую связь с соответствующими онтологиями и смежными предметными областями, входящими в состав Стандарта OSTIS]
  - [Доработать структуру и содержание титульной спецификации Стандарта OSTIS]
  - [Доработать титульные спецификации всех разделов Стандарта OSTIS]

- [Обеспечить достаточную полноту sc-спецификации всех рассматриваемых (описываемых, обозначаемых sc-элементами) сущностей]
- [Существенно расширить Библиографию OSTIS]
- [Постоянно отслеживать синонимию/омонимию sc-идентификаторов]
- [Постоянно совершенствовать контроль качества выполнения работ по развитию Стандарта OSTIS]
- [Необходимо постоянно проводить анализ публикаций других авторов по вопросам, близким тематике Стандарта OSTIS, и фиксировать в Стандарте OSTIS результаты сравнительного анализа точки зрения, представленной в Стандарте OSTIS с точками зрения иных авторов путем включения в Стандарт OSTIS спецификаций соответствующих библиографических источников с полезными цитатами, используемых в них терминов по сравнению с терминологией Стандарта OSTIS и так далее.]
- [Все sc-идентификаторы, входящие в состав sc.g-текстов, sc.n-текстов и различных иллюстраций должны иметь одинаковй шрифт и размер. За исключением, возможно размера sc-идентификаторов в sc.g-текстах]

}

Пункт 7.2.2.11. Достоинства Стандарта OSTIS

Стандарт OSTIS является примером перехода к принципиально новой форме представления и публикации *научно-технической информации*, результатов исследовательской деятельности — не просто к форме электронного документа, а к форме семантически структурированного электронного документа, являющегося частью *базы знаний* по соответствующей *научно-технической дисциплине*. Это существенно повышает эффективность использования накапливаемой человеком *научно-технической информации*, поскольку пользователь этой информации может не только ее просматривать (читать), но и взаимодействовать с *и.к.с.*, которая становится *партнером* в использовании нужной ему информации.

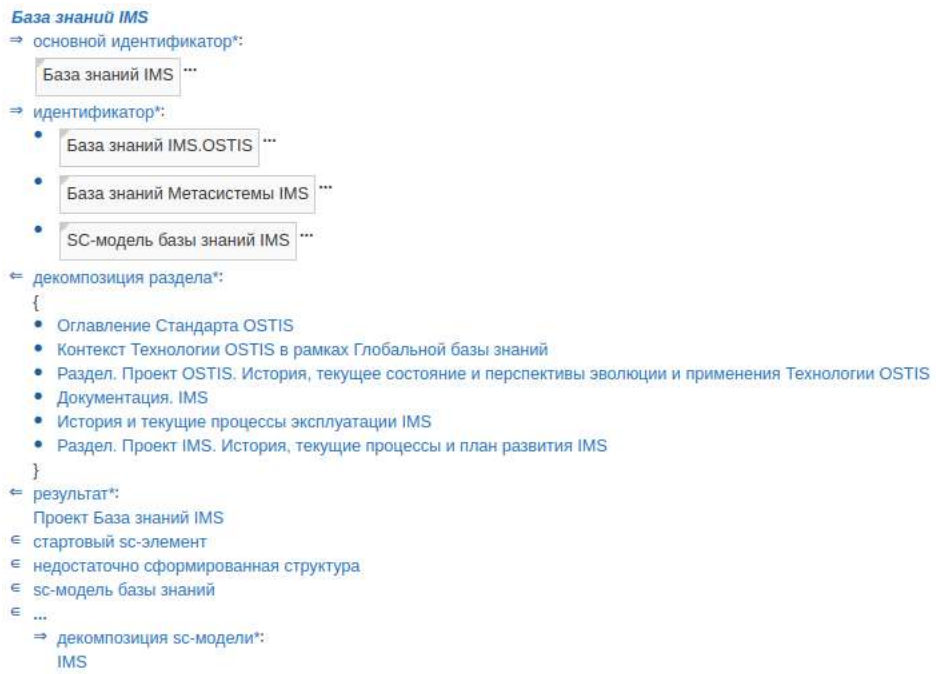
Проект создания и развития Стандарта OSTIS является прообразом принципиально нового подхода к организации *научно-технической деятельности* в рамках каждой *научной дисциплины*. Эта деятельность реализуется в форме открытого проекта, направленного на развитие *базы знаний* интеллектуального портала знаний по соответствующей научно-технической дисциплине. Такой уровень формализации *научно-технической информации*, который понятен не только специалистам, но и *интеллектуальным компьютерным систем* существенно повышает эффективность и расширяет области использования этой информации в *интеллектуальных компьютерных системах*. Так, например, интеллектуальный портал знаний по какой-либо технической дисциплине естественным образом становится *интеллектуальной системой автоматизации проектирования* технических систем соответствующего класса.

Также важным достоинством является факт того, что *Стандарт OSTIS* является прототипом учебных пособий нового поколения, имеющих четкую логико-семантическую структуризацию и стратификацию учебного материала, а также теоретико-множественную и логическую классификацию всех используемых понятий. Следовательно, использование *Стандарта OSTIS* не только как основы интеллектуальной системы автоматизации комплексной поддержки *жизненного цикла и.к.с. нового поколения*, но и в качестве комплексного учебного пособия по подготовке молодых специалистов в области *Искусственного интеллекта*, является прообразом широкого использования различных интеллектуальных порталов научно-технических знаний в качестве комплексных учебных пособий для подготовки молодых специалистов по соответствующим специальностям. Это существенно повысит качество образования, которое не должно отставать от развития соответствующих научно-технических направлений, а должно становиться неотъемлемой составляющей этого развития.

Стандарт OSTIS является основной частью базы знаний *Метасистемы OSTIS*, описывающей текущую версию *Технологии OSTIS* (см. рисунок *Рисунок. Стартовая страница базы знаний Метасистемы OSTIS*).

Рисунок. Стартовая страница базы знаний Метасистемы OSTIS

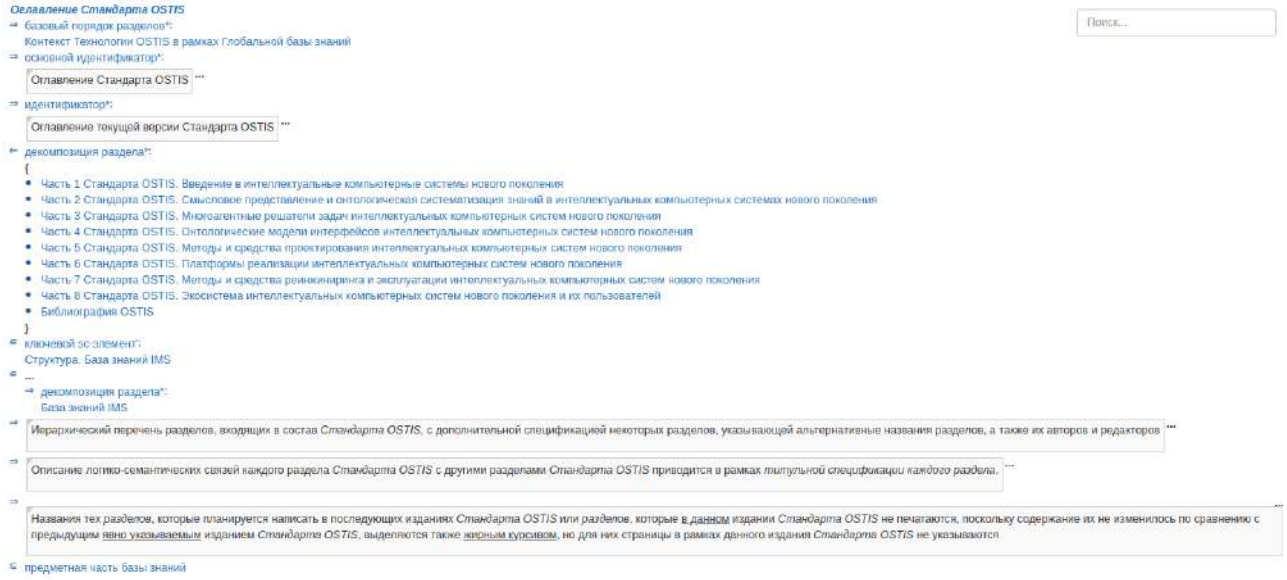
=



Предлагаемое представление *Стандарта OSTIS* обеспечивает эффективную семантическую навигацию по содержанию *Стандарта OSTIS*, а именно, обеспечивает возможность перейти к любой интересующей главе, позволяет задавать *Метасистеме OSTIS* широкий спектр нетривиальных вопросов о самых различных деталях и тонкостях *Технологии OSTIS*, и получать ответы на заданные вопросы (см. рисунок *Рисунок. Оглавление Стандарта OSTIS*).

Рисунок. Оглавление Стандарта OSTIS

=



По умолчанию ответы системы пользователю отображаются в *SCn-коде*, который является гипертекстовым вариантом внешнего отображения текстов *SC-кода* и может читаться как линейный текст.

Заключение к Главе 7.2.

Для реализации кооперативного целенаправленного и адаптивного взаимодействия *интеллектуальных компьютерных систем* в рамках автоматически формируемых коллективов *интеллектуальных компьютерных систем* необходима их семантическая совместимость, а это, в свою очередь, требует унификации интеллектуальных компьютерных систем. Унификация *интеллектуальной компьютерной системы* возможна только на основе общей формальной теории *интеллектуальных компьютерных систем* и соответствующего ей **стандарта интеллектуальных компьютерных систем**, а для этого необходима глубокая конвергенция различных направлений исследований в области искусственного интеллекта.

Поскольку результатом развития искусственного интеллекта как научной дисциплины является перманентная эволюция общей теории интеллектуальных компьютерных систем и соответствующего стандарта интеллектуальных компьютерных систем, для повышения темпов развития искусственного интеллекта и, соответственно, технологии разработки интеллектуальных компьютерных систем необходимо создание портала научно-технических знаний по искусственному интеллекту, обеспечивающего координацию деятельности специалистов, а также согласование и интеграцию результатов этой деятельности.

В главе рассмотрен подход к автоматизации процессов создания, развития и применения стандартов на основе *Технологии OSTIS*. На основе *Стандарта Технологии OSTIS* рассмотрены основные принципы, лежащие в основе предлагаемого подхода к стандартизации.

Предложенный в данной главе подход позволяет обеспечить не только возможность автоматизации процессов создания, согласования и развития стандартов, но и позволяет значительно повысить эффективность процессов применения стандарта, как в ручном, так и в автоматическом режиме.

Глава 7.3.
Структура Экосистемы OSTIS

⇒ *автор**:
- *Загорский А. С.*
- *Голенков В. В.*
- *Шункевич Д. В.*

⇒ *аннотация**:
[Рассмотрена структура *цифровой экосистемы интеллектуальных компьютерных систем* на основе *Технологии OSTIS*. Уточнена формальная трактовка таких понятий как *ostis-система*, *ostis-сообщество*, выделена типология *ostis-систем*, что в совокупности позволяет определить структуру *Экосистемы OSTIS*.]

⇒ *подраздел**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*
- *§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний*
- *§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения*
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

⇒ *ключевой знак**:
- *Экосистема OSTIS*

⇒ *ключевое понятие**:
- *распределенная система*
- *цифровая экосистема*

⇒ *библиографическая ссылка**:
- *Zagorskiy A..Princ fItEoNGICS-2022art*
- *Van B..KnowlSiaENoP-2005art*
- *Mack R..KnowlPatEDKW-2001art*
- *Ameri F..ProduLMCtKL-2005art*
- *Gerhard D.ProduLMCoC-2017art*
- *Meurisch C..ReferMoNGDP-2017art*
- *Meurisch C..ExploUEoPAI-2020art*
- *Jeni P..CanDPAP-2022art*
- *Awais A.Proac iIPAaSB-2022art*
- *Briscoe B..DigitESOoEA-2008art*
- *Boley H..DigitEPaS-2007art*
- *Masaharu T..aRevie otECTCE-2018art*
- *Mohseni M..aModelDAfSW-2021art*
- *Berners-Lee T..SemanW-2001art*
- *Kirrane S..PrivaSaPaRoP-2018art*
- *McCooI R.Rethi tSWP2-2006art*
- *Nacer H..SemanWSSAC-2014art*
- *Burstrom T..SoftwENaitFaDS-2022art*

Введение в Главу 7.3.

Понятие *цифровой экосистемы* представляет собой сложную и динамичную систему, которая состоит из множества компонентов, включая технологии, процессы, пользователей, предприятия и многое другое. В контексте цифровых технологий переход к *цифровой экосистеме* является ключевым аспектом для достижения целей бизнеса и общества.

Понятие *цифровой экосистемы* можно определить как совокупность цифровых продуктов и сервисов, которые взаимодействуют друг с другом и с внешней средой, образуя единую среду обитания. Реализация *цифровой экосистемы* сильно связано с формированием *распределенной системы*. Такой принцип реализации имеет как

преимущества (высокий уровень адаптивности, устойчивости, связности), так и недостатки (неоптимальность, неуправляемость, непредсказуемость поведения) (см. *Briscoe B..DigitESOoEA-2008art*, *Boley H..DigitEPaS-2007art* , *Burstrom T..SoftwENaitFaDS-2022art*). В отличие от полностью иерархически контролируемых систем, *цифровая экосистема* представляет собой децентрализованную структуру, которая обеспечивает более гибкое и устойчивое управление (см. *Masaharu T..aRevie otECTCE-2018art*).

При традиционных подходах к решению проблемы формирования *цифровой экосистемы* возникают проблемы, связанные с низким уровнем *интероперабельности* таких систем (см. *Li W..DigitECaP-2012art*). Традиционные подходы к решению данной проблемы зачастую неэффективны, поскольку каждая из систем имеет свой специализированный программный интерфейс и формат данных для взаимодействия. Это приводит к дополнительным расходам на устранение недостатков таких проблем. Поддержка жизненного цикла и модификация уже существующих систем может также потребовать дополнительных временных и ресурсных затрат.

Использование современных подходов к формированию *цифровой экосистемы*, таких как открытые стандарты и протоколы взаимодействия, может значительно упростить задачу обеспечения *интероперабельности* между различными системами. Это позволяет повысить эффективность и экономическую целесообразность проектов цифровой трансформации, снизить временные и финансовые затраты на разработку и поддержку *цифровой экосистемы* (см. *Mohseni M..aModelDAfSW-2021art*). Однако стоит отметить, что даже при принятии идей семантического веба (см. *Berners-Lee T..SemanW-2001art*) могут возникнуть некоторые проблемы или ограничения, которые необходимо учитывать (см. *Kirrane S..PrivaSaPaRoP-2018art*, *McCooI R.Rethi tSWP2-2006art*, *Nacer H..SemanWSSAC-2014art*).

Технология OSTIS предоставляет возможности для создания *цифровых экосистем*. Она обеспечивает эффективное управление данными и знаниями, обеспечивает автоматическую обработку информации и позволяет создавать интеллектуальные системы, способные обмениваться данными и знаниями между собой.

В данной главе монографии рассмотрена архитектура *Экосистемы OSTIS* и ее основных компонентов, методы разработки коллективов *ostis-систем*, а также типология *ostis-систем*, входящих в состав *Экосистемы OSTIS*. Основная цель - предоставить полное представление о возможностях создания *цифровых экосистем* на примере *Экосистемы OSTIS*.

§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ

⇒ *ключевое понятие**:
- *ostis-система*
- *самостоятельная ostis-система*
- *встроенная ostis-система*
- *коллектив ostis-систем*

⇒ *подраздел**:
- *Пункт 7.3.1.1. Поддержка совместимости между ostis-системами, входящими в состав Экосистемы OSTIS*
- *Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS*

⇒ *библиографическая ссылка**:
- *Zagorskiy A..Princ fItEoNGICS-2022art*

Введение в § 7.3.1.

Для создания успешной *цифровой экосистемы* необходимо решать множество проблем, связанных с обеспечением высокого уровня интероперабельности между самостоятельно действующими системами. Одним из возможных решений является переход к универсальным сообществам индивидуальных *интеллектуальных кибернетических систем*, которые объединяются в *многоагентные системы*.

Реализация такого универсального сообщества интероперабельных *интеллектуальных кибернетических систем* может осуществляться в виде глобальной *Экосистемы OSTIS* (см. *Zagorskiy A..Princ fItEoNGICS-2022art*). *Экосистема OSTIS* — социотехническая экосистема, представляющая собой коллектив взаимодействующих *семантических компьютерных систем* и осуществляющая перманентную поддержку эволюции и семантической совместимости всех входящих в нее систем, на протяжении всего их жизненного цикла.

Система, построенная в соответствии с требованиями и стандартами *Технологии OSTIS*, определяется как *ostis-система*.

Экосистема OSTIS представляет собой коллектив взаимодействующих:
- самих *ostis-систем*;
- пользователей указанных *ostis-систем* (как конечных пользователей, так и разработчиков);
- некоторых *компьютерных систем*, не являющихся *ostis-системами*, но рассматриваемых ими в качестве дополнительных *информационных ресурсов* или *сервисов*.

В рамках *Экосистемы OSTIS ostis-системы* способны коммуницировать друг с другом и формировать специализированные коллективы для коллективного решения сложных задач. Такой подход не только повышает уровень интеллекта каждой *индивидуальной кибернетической системы*, но и обеспечивает более эффективное взаимодействие между ними в рамках единой *цифровой экосистемы*. Это обеспечивает существенное развитие целого ряда свойств каждой *компьютерной системы*, позволяющих значительно повысить *уровень интеллекта* (и, прежде всего, их *уровень обучаемости* и *уровень социализации*).

Участники коллектива *Экосистемы OSTIS* характеризуются как:
- *семантически совместимые*;
- постоянно эволюционирующие индивидуально;
- постоянно поддерживающие свою совместимость с другими участниками в ходе своей индивидуальной эволюции;
- способные децентрализованно координировать свою деятельность.

Экосистема OSTIS — это переход от *самостоятельных ostis-систем* к *коллективам ostis-систем*, то есть к распределенным *ostis-системам*.

ostis-система
⇒ *разбиение**:
{• *самостоятельная ostis-система*
• *встроенная ostis-система*
• *коллектив ostis-систем*
}

Цель *Экосистемы OSTIS* — обеспечить постоянную поддержку совместимости компьютерных систем, входящих в *Экосистему OSTIS* как на этапе их разработки, так и в ходе их эксплуатации. Проблема заключается в том, что в ходе эксплуатации систем, входящих в *Экосистему OSTIS*, они могут изменяться из-за чего совместимость может нарушаться.

Экосистема OSTIS
⇒ *задачи**:
- оперативное внедрение всех согласованных изменений стандарта ostis-систем (в том числе, и изменений систем используемых понятий и соответствующих им терминов)
- перманентная поддержка высокого уровня взаимопонимания всех систем, входящих в *Экосистему OSTIS*, и всех их пользователей
- корпоративное решение различных комплексных задач, требующих координации деятельности нескольких ostis-систем, а также, возможно, некоторых пользователей

Пункт 7.3.1.1. Поддержка совместимости между ostis-системами, входящими в состав Экосистемы OSTIS

Каждая система, входящая в состав *Экосистемы OSTIS*, должна:
- интенсивно, активно и целенаправленно обучаться, как с помощью учителей-разработчиков, так и самостоятельно;
- сообщать всем другим системам о предлагаемых или окончательно утвержденных изменениях в онтологиях и, в частности, в наборе используемых понятий;
- принимать от других ostis-систем предложения об изменениях в онтологиях, в том числе в наборе используемых понятий, для согласования или утверждения этих предложений;
- реализовывать утвержденные изменения в онтологиях, хранимых в ее базе знаний;
- способствовать поддержанию высокого уровня семантической совместимости не только с другими ostis-системами, входящими в *Экосистему OSTIS*, но и со своими пользователями (обучать их, информировать их об изменениях в онтологиях).

К *самостоятельной ostis-системе*, входящей в состав *Экосистемы OSTIS*, предъявляются особые требования:

- она должны обладать всеми необходимыми знаниями и навыками для обмена сообщениями и целенаправленной организации взаимодействия с другими *ostis-системами*, входящими в *Экосистему OSTIS*;
- в условиях постоянного изменения и эволюции *ostis-систем*, входящих в *Экосистему OSTIS*, она должна сама следить за состоянием своей совместимости (согласованности) со всеми остальными *ostis-системами*, то есть должна самостоятельно поддерживать эту совместимость, согласовывая с другими *ostis-системами* все требующие согласования изменения, происходящие у себя и в других системах.

Для обеспечения высокой эффективности эксплуатации и высоких темпов эволюции *Экосистемы OSTIS* необходимо постоянно повышать уровень информационной совместимости (уровень взаимопонимания) не только между компьютерными системами, входящими в состав *Экосистемы OSTIS*, но также между этими системами и их пользователями.

Одним из направлений обеспечения такой совместимости является стремление к тому, чтобы база знаний, картина мира каждого пользователя стала частью объединенной базы знаний *Экосистемы OSTIS*. Это значит, что каждый пользователь должен знать, как устроена структура каждой научно-технической дисциплины (объекты исследования, предметы исследования, определения, закономерности и так далее), как могут быть связаны между собой различные дисциплины.

Поддержка совместимости *Экосистемы OSTIS* с ее пользователями осуществляется следующим образом:
- в каждую *ostis-систему* включаются встроенные *ostis-системы*, ориентированные
  - на перманентный мониторинг деятельности конечных пользователей и разработчиков этой *ostis-системы*,
  - на анализ качества и, в первую очередь, корректности этой деятельности,
  - на повышение квалификации пользователей (персонифицированное обучение);
- в состав *Экосистемы OSTIS* включаются *ostis-системы*, специально предназначенные для обучения пользователей *Экосистемы OSTIS* базовым общепризнанным знаниям и навыкам решения соответствующих классов задач.

Экосистеме OSTIS ставится в соответствие ее объединенная база знаний, которая представляет собой виртуальное объединение баз знаний всех ostis-систем, входящих в состав *Экосистемы OSTIS*. Качество этой базы знаний (полнота, непротиворечивость, чистота) является постоянной заботой всех *самостоятельных ostis-систем*, входящих в состав *Экосистемы OSTIS*.

Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS

⇒ *ключевой знак**:
- *Корпоративная система Экосистемы OSTIS*

⇒ *ключевое понятие**:
- *агент Экосистемы OSTIS*
- *пользователь Экосистемы OSTIS*
- *ostis-сообщество*

Субъект, входящий в состав *Экосистемы OSTIS*, является *агентом Экосистемы OSTIS*.

агент Экосистемы OSTIS

⇒ *разбиение**:
- {• *индивидуальная ostis-система Экосистемы OSTIS*
  - ⇒ *разбиение**:
    - {• *самостоятельная ostis-система Экосистемы OSTIS*
      • *встроенная ostis-система Экосистемы OSTIS*
    - }
- • *пользователь Экосистемы OSTIS*
- • *ostis-сообщество*
  - ⇒ *разбиение**:
    - {• *простое ostis-сообщество*
      • *иерархическое ostis-сообщество*
    - }
- }

Понятие *ostis-сообщества* представляет собой не только *коллектив ostis-систем*, но также определенный коллектив людей (пользователей и разработчиков соответствующих *ostis-систем*). *ostis-сообщество* является устойчивым фрагментом *Экосистемы OSTIS*, обеспечивающим комплексную автоматизацию определенной части коллективной человеческой деятельности и перманентное повышение ее эффективности. *иерархическим ostis-сообществом*

называется такое *ostis-сообщество*, по крайней мере одним из членов которого является некоторое другое *ostis-сообщество*.

Правила поведения *агентов Экосистемы OSTIS*:
- согласовывать денотационную семантику всех используемых знаков (в первую очередь понятий);
- согласовывать терминологию, устранять противоречия и информационные дыры;
- постоянно бороться с синонимией и омонимией как на уровне sc-элементов (внутренних знаков), так и на уровне соответствующих им терминов и прочих внешних идентификаторов (внешних обозначений);
- каждый *агент Экосистемы OSTIS* по своей инициативе может стать членом любого ostis-сообщества Экосистемы OSTIS после соответствующей регистрации;

Все правила поведения *агентов Экосистемы OSTIS* должны соблюдаться не только *ostis-системами*, которые являются агентами *Экосистемы OSTIS*, но и людьми, являющиеся ими. Корректное поведение *ostis-систем* как *агентов Экосистемы OSTIS* значительно проще обеспечить, чем корректное поведение людей в качестве таких агентов. Поведение пользователей (естественных агентов) *Экосистемы OSTIS* необходимо внимательно мониторить и контролировать, постоянно способствуя повышению уровня их квалификации как *агентов Экосистемы OSTIS*, а также повышению уровня их мотивации, целенаправленности и самореализации.

Экосистема OSTIS является максимальным *иерархическим ostis-сообществом*, обеспечивающим комплексную автоматизацию всех видов человеческой деятельности. Оно не может входить в состав какого-либо другого *ostis-сообщества*. Принципы, лежащие в основе *Экосистемы OSTIS*:
- *Экосистема OSTIS* представляет собой сеть *ostis-сообществ*;
- Каждому *ostis-сообществу* взаимно однозначно соответствует *корпоративная ostis-система* этого *ostis-сообщества*;
- Каждое *ostis-сообщество* может входить в состав любого другого *ostis-сообщества* по своей инициативе. Формально это означает, что *корпоративная ostis-система* первого *ostis-сообщества* является членом другого *ostis-сообщества*;
- Каждому специалисту, входящему в состав *Экосистемы OSTIS* ставится во взаимнооднозначное соответствие его *персональный ostis-ассистент*, который трактуется как *корпоративная ostis-система* вырожденного *ostis-сообщества*, состоящего из одного человека.

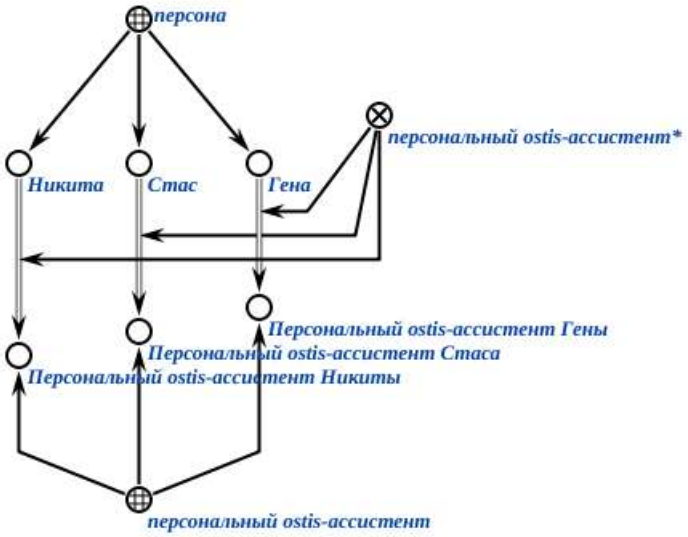
В *Экосистеме OSTIS* можно выделить следующие уровни иерархии:
- индивидуальные *кибернетические системы* (индивидуальные *ostis-системы* и люди, являющиеся конечными пользователями *ostis-систем*);
- иерархическая система *ostis-сообществ*, членами каждого из которых могут быть *индивидуальные ostis-системы*, люди, а также другие *ostis-сообщества*;
- *Максимальное ostis-сообщество Экосистемы OSTIS*, не являющееся членом никакого другого *ostis-сообщества*, входящего в состав *Экосистемы OSTIS*.

Качество *Экосистемы OSTIS* во многом определяется эффективностью взаимодействия каждой *ostis-системы* (в том числе каждого *ostis-сообщества*), каждого человека со своей внешней средой, а также качеством и чистотой самой внешней среды. Потому основной целью *Экосистемы OSTIS* является повышение качества информационной внешней среды для всех субъектов, входящих в состав *Экосистемы OSTIS*. Иными словами, *Экосистема OSTIS* обеспечивает поддержку информационной экологии человеческого общества.

ostis-сообщество
⇒ *разбиение**:
{• *минимальное ostis-сообщество*
• *коллектив ostis-систем*
}

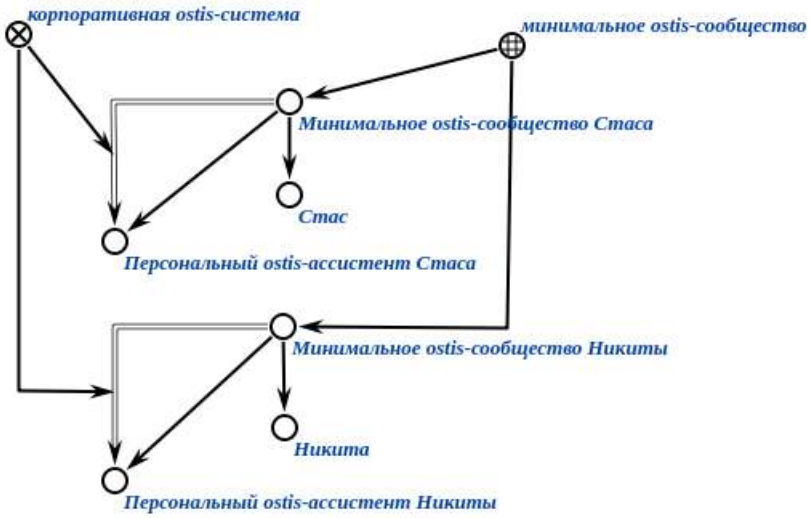
Каждой персоне, входящей в состав *Экосистемы OSTIS* взаимно однозначно соответствует его личный (персональный) ассистент в виде *персонального ostis-ассистента*. Таким образом, количество *персональных ostis-ассистентов*, входящих в состав *Экосистемы OSTIS*, совпадает с числом персон, входящих в состав *Экосистемы OSTIS* (см. *SCg-текст. Пример персоны и соответствующего ему персонального ostis-ассистента*).

SCg-текст. Пример персоны и соответствующего ему персонального ostis-ассистента
=



Коллектив, состоящий из персоны и соответствующего ей *персонального ostis-ассистента*, фактически является *минимальным ostis-сообществом*(см. *SCg-текст. Пример минимального ostis-сообщества*).

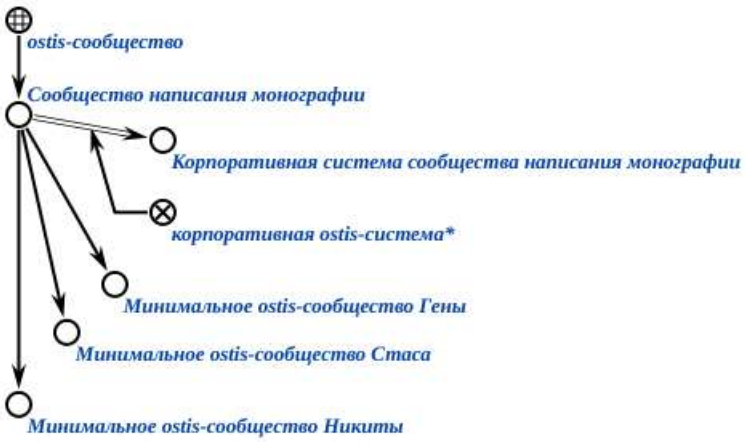
SCg-текст. Пример минимального ostis-сообщества
=



Поскольку формально в не *минимальные ostis-сообщества* входят не персоны, а соответствующие им *персональные ostis-ассистенты*, все *ostis-сообщества*, кроме *минимальных ostis-сообществ*, являются *коллективами ostis-систем*.

корпоративная ostis-система есть центральная *ostis-система*, осуществляющая координацию, организацию, а также поддержку эволюции деятельности членов соответствующего *ostis-сообщества*. *корпоративная ostis-система* является представителем соответствующего *ostis-сообщества* в других *ostis-сообществах*, членом которых оно является(см. *SCg-текст. Пример корпоративной ostis-системы ostis-сообщества*).

SCg-текст. Пример корпоративной ostis-системы ostis-сообщества
=



Основным назначением *Корпоративной системы Экосистемы OSTIS* является организация общего взаимодействия при выполнении самых различных видов и областей человеческой деятельности, которые могут быть либо полностью автоматизированными, либо частично автоматизированными, либо вообще неавтоматизированными. Из этого следует, что база знаний *Корпоративной системы Экосистемы OSTIS* должна содержать *Общую формальную теорию человеческой деятельности*, включающей в себя типологию видов и областей человеческой деятельности, а также общую методологию этой деятельности.

Деятельность в области Искусственного интеллекта, осуществляемая на основе Технологии OSTIS
⇒ *основной продукт**:
Экосистема OSTIS
⇒ *подпроект**:
- *Проект Метасистемы OSTIS*
- *Проект программной реализации абстрактной sc-машины*
- *Проект разработки универсального sc-компьютера*

Продуктом человеческой деятельности в области Искусственного интеллекта, осуществляемой на основе *Технологии OSTIS*, является не просто множество *ostis-систем* различного назначения, а Экосистема, состоящая из взаимодействующих *ostis-систем* и их пользователей.

По назначению *ostis-системы*, входящие в *Экосистему OSTIS*, могут быть:
- ассистентами конкретных пользователей или конкретных пользовательских коллективов;
- типовыми встраиваемыми подсистемами *ostis-систем*;
- системами информационной и инструментальной поддержки проектирования различных компонентов и различных классов *ostis-систем*;
- системами информационной и инструментальной поддержки проектирования или производства различных классов технических и других искусственно создаваемых систем;
- порталами знаний по самым различным научным дисциплинам;
- системами автоматизации управления различными сложными объектами (производственными предприятиями, учебными заведениями, кафедрами вузов, конкретными обучаемыми);
- интеллектуальными справочными и help-системами;
- интеллектуальными робототехническими системами.

Типология *ostis-систем*, являющихся агентом Экосистемы OSTIS, представлена ниже.

ostis-система, являющаяся агентом Экосистемы OSTIS
⊃ *персональный ostis-ассистент*
⊃ *корпоративная ostis-система*
⊃ *ostis-портал знаний*
⊃ *ostis-система автоматизации проектирования*
⊃ *ostis-система автоматизации производства*
⊃ *ostis-система автоматизации образовательной деятельности*
  ⊃ *обучающаяся ostis-система*
  ⊃ *корпоративная ostis-система виртуальной кафедры*
⊃ *ostis-система автоматизации бизнес-деятельности*

⊃ *ostis-система автоматизации управления*
- ⊃ *ostis-система управления проектами соответствующего вида*
- ⊃ *ostis-система сенсомоторной координации при выполнении определенного вида сложных действий во внешней среде*
  - ⊃ *ostis-система управления самостоятельным перемещением*
  - ⊃ *робота по пересеченной местности*

§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний

⇒ *ключевое понятие**:
- *портал знаний*
- *ostis-портал знаний*

⇒ *библиографическая ссылка**:
- *Van B..KnowlSiaENoP-2005art*
- *Mack R..KnowlPatEDKW-2001art*

Понятие *портала знаний* представляет собой один из способов создания централизованного доступа к информации, которая может быть необходима для решения задач, связанных с работой в организации. Такие порталы могут содержать информацию, связанную с процессами, документацией, процедурами, обучающими материалами, а также ответы на часто задаваемые вопросы (см. *Van B..KnowlSiaENoP-2005art*, *Mack R..KnowlPatEDKW-2001art*).

Одним из ключевых преимуществ *порталов знаний* является их способность к сбору и хранению информации из различных источников, таких как базы данных, системы управления документами, системы управления проектами и так далее. Это позволяет пользователям получать полную и актуальную информацию в одном месте.

На основе *портала знаний* обеспечивается возможность взаимодействия между пользователями путем создания форумов, обсуждений и коллективного редактирования документов. Это может способствовать обмену знаниями и опытом между сотрудниками организации и повысить эффективность их работы.

При создании *порталов знаний* возникают проблемы, связанные с организацией и управлением информацией. Например, необходимо обеспечить корректное и структурированное хранение информации, ее поиск и обновление. Также необходимо учитывать потребности пользователей и обеспечить удобный и интуитивно понятный интерфейс.

Целями интеллектуального портала знаний являются:
- ускорение погружения каждого человека в новые для него области при постоянном сохранении общей целостной картины Мира (образовательная цель);
- фиксация в систематизированном виде новых результатов так, чтобы все основные связи новых результатов с известными были четко обозначены;
- автоматизация координации работ по рецензированию новых результатов;
- автоматизация анализа текущего состояния базы знаний.

Создание интеллектуальных *порталов знаний*, обеспечивающих повышение темпов интеграции и согласования различных точек зрения, — это способ существенного повышения темпов эволюции научно-технической деятельности. Совместимые *порталы знаний*, реализованные в виде *ostis-систем*, входящих в *Экосистему OSTIS*, являются основой новых принципов организации научной деятельности, в которой результатами этой деятельности являются не статьи, монографии, отчеты и другие научно-технические документы, а фрагменты глобальной *базы знаний*, разработчиками которых являются свободно формируемые научные коллективы, состоящие из специалистов в соответствующих научных дисциплинах. С помощью *ostis-порталов знаний* осуществляется как координация процесса рецензирования новой научно-технической информации, поступающей от научных работников в *базы знаний* этих порталов, так и процесс согласования различных точек зрения ученых (в частности, введению и семантической корректировке понятий, а также введению и корректировке терминов, соответствующих различным сущностям).

Реализация семейства семантически совместимых *ostis-порталов знаний* в виде совместимых *ostis-систем*, входящих в состав *Экосистемы OSTIS*, предполагает разработку иерархической системы семантически согласованных формальных онтологий, соответствующих различным дисциплинам, с четко заданным наследованием свойств описываемых сущностей и с четко заданными междисциплинарными связями, которые описываются связями между соответствующими формальными онтологиями и специфицируемыми ими предметными областями.

Реализация *ostis-порталов знаний* на основе *Технологии OSTIS* предоставляет ряд преимуществ по сравнению с другими подходами. Ниже приведены некоторые из них:
- использование методов семантической обработки информации, что позволяет более точно и эффективно организовывать и искать информацию на портале знаний;

- высокий уровень гибкости и расширяемости, что позволяет адаптировать *ostis-порталы знаний* под различные нужды и требования пользователей;
- автоматическая интеграция *ostis-порталов знаний* с другими *ostis-системами* в рамках *Экосистемы OSTIS*, что позволяет создать централизованный доступ к информации из различных источников.
- возможность создания персонализированного *ostis-портала знаний*, который учитывает интересы и потребности каждого пользователя, что позволяет более эффективно использовать знания *ostis-систем*;
- возможность производить *ostis-порталы знаний* быстро и с минимальными затратами благодаря использованию существующих компонентов и инструментов.

Примером *портала знаний*, построенного в виде *ostis-системы* является *Метасистема OSTIS*, содержащая все известные на текущий момент знания и навыки, входящие в состав *Технологии OSTIS*.

Таким образом, реализация *порталов знаний* на основе *Технологии OSTIS* позволяет создать более эффективную и гибкую систему для хранения, организации и поиска знаний, которая может быть адаптирована под различные требования пользователей и организаций.

§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения

⇒ *ключевое понятие**:
- *корпоративная система*
- *корпоративная ostis-систем*

⇒ *библиографическая ссылка**:
- *Ameri F..ProduLMCtKL-2005art*
- *Gerhard D.ProduLMCoC-2017art*

корпоративные системы представляют собой программные решения, предназначенные для автоматизации бизнес-процессов и управления ресурсами и данными внутри организации. Они могут включать в себя различные подсистемы, такие как управление отношениями с клиентами, управление контентом, управление проектами, управление ресурсами предприятия, управление документами и многое другое.

Роль *корпоративных систем* в современных организациях заключается в обеспечении эффективного управления бизнес-процессами и ресурсами, повышении производительности и качества работы, а также обеспечении прозрачности и оперативности принятия решений на основе актуальных данных (см. *Ameri F..ProduLMCtKL-2005art*, *Gerhard D.ProduLMCoC-2017art*).

корпоративные системы могут использоваться для следующих целей:
- автоматизация многих рутинных задач, таких как обработка заказов, управление складом, учет финансовых операций и так далее. Это позволяет сократить время на выполнение задач и уменьшить количество ошибок.
- сбор, хранение и обработка данных о бизнес-процессах и ресурсах организации. Это позволяет увеличить точность и оперативность принятия решений, а также обеспечить прозрачность в управлении организацией.
- эффективное управление ресурсами организации, такими как финансы, трудовые ресурсы, материальные и технические ресурсы и так далее. Это позволяет сократить затраты на управление ресурсами и повысить эффективность их использования.
- управление отношениями с клиентами, автоматизация процессов продаж и обслуживания, а также анализ данных о клиентах. Это позволяет повысить удовлетворенность клиентов и увеличить объемы продаж.
- управление проектами, планирование и отслеживание выполнения работ, управление ресурсами и расписание проектов. Это позволяет повысить эффективность выполнения проектов, уменьшить сроки выполнения работ и снизить затраты на проекты.
- управление документами, контроль версиями, автоматизация процессов редактирования и утверждения документов. Это позволяет повысить эффективность работы с документами и обеспечить безопасность их хранения и передачи.

Хотя *корпоративные системы* могут принести значительные выгоды для организаций, они также могут столкнуться с рядом проблем, связанных с их внедрением и эксплуатацией. Ниже перечислены некоторые из этих проблем:
- Внедрение корпоративных систем может быть дорогостоящим и трудоемким процессом, который требует значительных ресурсов и экспертизы. Кроме того, многие системы могут потребовать изменения бизнес-процессов и требовать адаптации культуры организации.
- Корпоративные системы могут столкнуться с проблемами совместимости с другими системами, используемыми в организации. Это может привести к проблемам с обменом данными и снижению эффективности работы.

- Корпоративные системы могут стать мишенью для кибератак, поэтому важно обеспечить безопасность хранения и передачи данных, используемых в системах.
- Корпоративные системы могут потребовать значительных затрат на обслуживание и поддержку, включая установку обновлений, устранение ошибок и техническую поддержку.
- Внедрение новых корпоративных систем может потребовать обучения персонала, что может быть трудоемким и затратным процессом.
- Внедрение корпоративных систем может потребовать изменения бизнес-процессов, что может быть сложным и вызвать сопротивление со стороны сотрудников.

Для создания семантически совместимых интеллектуальных *корпоративных систем* необходимо обеспечить высокую степень гибкости, масштабируемости, автоматизации и интеграции. Это позволит организациям более эффективно управлять ресурсами и данными и повысить их конкурентоспособность на рынке. Для достижения этих целей необходимо использовать современные технологии, такие как Аналитика данных, Машинное обучение, Искусственный интеллект и технологии распределенных вычислений. Кроме того, необходимо учитывать особенности организации и ее бизнес-процессов, чтобы обеспечить максимальную эффективность использования системы.

Для решения задачи формирования корпоративной системы целесообразно применение *Технологии OSTIS. корпоративная ostis-система* позволяет отслеживать, анализировать и постепенно автоматизировать все процессы обработки данных в рамках ostis-сообщества. Такая система действует по следующим принципам:

- интеллектуальные подсистемы (агенты) упорядочивают структуру данных таким образом, что актуальная информация всегда доступна, а устаревшая информация автоматически архивируется или удаляется в соответствии с законами о хранении и защите данных в режиме реального времени;
- запросы к системе выполняются в виде простых инструкций, система помогает менеджерам вводить необходимую информацию, осуществляет частичную или полную автоматизацию обновления информации из баз данных, доступных через Интернет;
- интеллектуальные подсистемы выполняют структуризацию и классификацию документов и информации для принятия быстрых и правильных решений, автоматически обрабатывает документы и доступные базы данных для отбора ключевой информации, необходимой в данный момент и в будущем;
- существующее системное окружение на предприятии может быть легко подключено к системе через открытые интерфейсы, вся информация остается доступной;
- все ключевые системы данных синхронизируются с основной системой, данные постоянно сравниваются друг с другом, чтобы избежать потерь;
- вся информация доступна в базе знаний, которая является источником данных для рабочих процессов, отчетности и комплексных проверок;

Достоинствами внедрения предложенной системы являются:

- помощь сбора и оценки информации без преднамеренных искажений или ошибок, связанных с человеческим фактором;
- предоставление возможности полного контроля своих данных;
- система предоставляет только высококачественные, достоверные и актуальные данные;
- цифровое представление всех процессов сообщества обеспечивает интегрированную обработку информации внутри сообщества, что дает полную прозрачность управления, облегчает доступ ко всей информации и ее анализ;
- благодаря поддержке интеллектуальных подсистем все необходимые данные из документов, процессов и внешних источников могут быть извлечены, структурированы и грамотно оценены.

С точки зрения структуры *Экосистемы OSTIS*, *корпоративная ostis-система* осуществляет координацию и эволюцию деятельности некоторых групп *ostis-систем* и их пользователей. Основная цель *корпоративных ostis-систем* — локализовать *базы знаний* указанных групп *ostis-систем*, перевести их из статуса виртуальных в статус реальных и автоматизировать их эволюцию.

корпоративные ostis-системы могут быть применены в различных областях: медицина и здравоохранение, образовательная деятельность широкого профиля, страховой бизнес, промышленная деятельность, административная деятельность, недвижимость, транспорт и так далее.

§ 7.3.4. Персональные ostis-ассистенты пользователей

⇒ *ключевое понятие**:

- *персональный ассистент*
- *персональный ostis-ассистент*

⇒ *библиографическая ссылка**:
- *Meurisch C..ReferMoNGDP-2017art*
- *Meurisch C..ExploUEoPAI-2020art*
- *Jeni P..CanDPAP-2022art*
- *Awais A.Proac iIPAaSB-2022art*

Общество должно обеспечивать персональную поддержку каждому человеку, учитывая его индивидуальные особенности, с целью достижения следующих целей:
- максимального уровня физического здоровья, активности и долголетия;
- максимального уровня физического комфорта, личного пространства и материального благосостояния;
- максимального уровня социального комфорта и защиты прав и свобод.

Для этого должен осуществляться:
- персональный мониторинг каждой личности по всем направлениям;
- диагностика и устранение нежелательных отклонений;
- оказание своевременной персональной помощи в уточнении направлений дальнейшей эволюции каждой личности.

Необходимо перейти от оказания услуг в решении различных проблем по инициативе самих лиц, столкнувшихся с этими проблемами, к своевременному обнаружению возможности возникновения этих проблем и к соответствующей профилактике. Это возможно только при наличии четкой системной организации персонального мониторинга.

Цифровые *персональные ассистенты* — это программы, основанные на технологиях искусственного интеллекта и машинного обучения, которые помогают пользователям в выполнении повседневных задач, таких как составление расписания, управление контактами, поиск информации, напоминание о важных событиях и так далее (см. *Meurisch C..ReferMoNGDP-2017art*, *Meurisch C..ExploUEoPAI-2020art*, *Jeni P..CanDPAP-2022art*, *Awais A.Proac iIPAaSB-2022art*).

Персональный ассистент должен учитывать, что роли пользователя в обществе могут меняться, расширяться, также как и его интересы и цели. При этом, все *персональные ассистенты* должны быть семантически совместимыми с целью понимания друг друга, а также обладать способностью самостоятельно взаимодействовать в рамках различных *корпоративных систем*, представляя интересы своих пользователей.

Одной из основных проблем, связанных с реализацией цифровых *персональных ассистентов*, является необходимость точного понимания запросов и задач, поступающих от пользователя. Это может быть вызвано различными факторами, такими как нечеткость и неоднозначность формулировок, использование аббревиатур и сленга, а также многозначность некоторых слов.

Пользователь не обязан знать множество сервисов, из которых он должен выбирать подходящий ему функционал. Комплекс семантически совместимых сервисов должен располагаться "за кадром". Следовательно, все используемые информационные ресурсы и сервисы должны быть семантически совместимы. Выбор подходящего для пользователя ресурса или сервиса должен производить его *персональный ассистент*.

Таким образом, при реализации цифровых *персональных ассистентов* необходимо обеспечить их масштабируемость и адаптивность к потребностям пользователей. Это означает, что система должна быть способна автоматически адаптироваться к изменениям в поведении пользователя, учитывая его предпочтения, особенности работы и другие факторы.

Технология OSTIS позволяет создавать семантически совместимые системы, которые способны обрабатывать запросы и задачи пользователей, учитывая их контекст и смысл. Это достигается за счет использования семантических сетей, которые позволяют описывать знания и связи между ними. Кроме того, *технология OSTIS* обеспечивает масштабируемость и гибкость системы, что позволяет ей адаптироваться к изменениям в поведении пользователей и изменениям в их потребностях.

Персональный ostis-ассистент есть *ostis-система*, являющаяся *персональным ассистентом* пользователя в рамках *Экосистемы OSTIS*. Такая система предоставляет возможности:
- анализа деятельности пользователя и формирования рекомендаций по ее оптимизации;
- адаптации под настроение пользователя, его личностные качества, общую окружающую обстановку, задачи, которые чаще всего решает пользователь;
- перманентного обучения самого ассистента в процессе решения новых задач, при этом обучаемость потенциально не ограничена;
- вести диалог с пользователем на естественном языке, в том числе в речевой форме;
- отвечать на вопросы различных классов, при этом если системе что-то не понятно, то она сама может задавать встречные вопросы;
- автономного получения информации от всей окружающей среды, а не только от пользователя (в текстовой или речевой форме).

При этом система может как анализировать доступные информационные источники (например, в интернете), так и анализировать окружающий ее физический мир, например, окружающие предметы или внешний вид пользователя.

Достоинства *персонального ostis-ассистента*:
- пользователю нет необходимости хранить разную информацию в разной форме в разных местах, вся информация хранится в единой базе знаний компактно и без дублирований;
- благодаря неограниченной обучаемости ассистенты могут потенциально автоматизировать практически любую деятельность, а не только самую рутинную;
- благодаря базе знаний, ее структуризации и средствам поиска информации в базе знаний пользователь может получить более точную информацию более быстро.

Персональные ассистенты имеют самое различное назначение и могут быть использованы для самых различных категорий пользователей (пациент, юридическое обслуживание, административное обслуживание, покупатель, потребитель различных услуг). *Персональный ostis-ассистент* может использовать знания и данные, хранящиеся в других *ostis-системах*, таких как *корпоративные ostis-системы*, чтобы предоставлять пользователю более полную и актуальную информацию. Это может быть особенно полезно для пользователей, которые работают с большим количеством данных и информации. *Персональный ostis-ассистент* автоматически интегрируется с другими *ostis-системами*, что позволяет ему более эффективно работать с данными и информацией. Он может использовать технологии машинного обучения и искусственного интеллекта для адаптации к поведению пользователя и улучшения его производительности и эффективности. *Персональный ostis-ассистент* может быть создан и настроен с учетом конкретных потребностей организации и ее процессов, что может привести к значительным экономическим и производственным преимуществам.

Таким образом, *персональные ostis-ассистенты* обладают рядом преимуществ по сравнению с другими реализациями цифровых *персональных ассистентов*, таких как более точное понимание запросов и задач пользователей, доступ к актуальным данным и информации, автоматическая интеграция с другими *ostis-системами* в рамках *Экосистемы OSTIS* и адаптация к потребностям организации и ее процессов.

Заключение к Главе 7.3.

Экосистема OSTIS представляет собой саморазвивающуюся сеть *ostis-систем*, которая обеспечивает комплексную автоматизацию всевозможных видов и областей человеческой деятельности.

Экосистема OSTIS является следующим этапом развития человеческого общества, обеспечивающий существенное повышение уровня общественного, коллективного интеллекта путем преобразования человеческого общества в экосистему, состоящую из людей и семантически совместимых интеллектуальных систем. *Экосистема OSTIS* — предлагаемый подход к реализации *smart-общества* или Общества 5.0, построенного на основе *Технологии OSTIS*.

Сверхзадачей *Экосистемы OSTIS* является не просто комплексная автоматизация всех видов человеческой деятельности (только тех видов деятельности, автоматизация которых целесообразна), но и существенное повышение уровня интеллекта различных человеко-машинных сообществ и всего человеческого общества в целом.

Глава 7.4.

Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами

⇒ *автор**:
- *Загорский А. С.*
- *Таранчук В. Б.*
- *Шункевич Д. В.*
- *Соловьев А. М.*
- *Коршунов Р. А.*
- *Савенок В. А.*

⇒ *аннотация**:
[В главе описываются общие принципы *интеграции*, а также принципы *интеграции Экосистемы OSTIS* с разнородными *сервисами* и структурированными *информационными ресурсами*. Также описывается *интеграция* инструментов компьютерной алгебры в *ostis-системы*. Эта глава будет полезна для специалистов в области программной инженерии, искусственного интеллекта и системного анализа, которые заинтересованы в *интеграции* различных *сервисов* и *ресурсов* в *интеллектуальные системы*.]

⇒ *подраздел**:
- *§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

⇒ *ключевой знак**:
- *Экосистема OSTIS*

⇒ *ключевое понятие**:
- *интеграция*
- *информационный ресурс*
- *сервис*

⇒ *библиографическая ссылка**:
- *Valdez O.How tDaDEaPF-2019art*
- *Li W..DigitECaP-2012art*
- *Caldarola E..Appro tOIfORiK-2015art*
- *Bork D..aOpenPfMMCtO-2019art*
- *Kroshchanka A..aNeuraSAtCV-2022art*
- *RDFCAS-2023el*
- *RDB tRML-2012el*
- *EasilGHKG-2022el*
- *ЧтоТОД-2023эл*
- *Дьяконов В.ЭнцикКА-2022кн*
- *Аладьев В.З..Введе вСПМ22-1999кн*
- *Аладьев В.З..МодулПМvMaVV-2011кн*
- *List oCAS-2023el*
- *СистеКАОС-эл*
- *СистеКАМ-эл*
- *Стахин Н.А.ОсновРсСАС-2008кн*
- *AXIOM.tSCS-эл*
- *Дьяконов В.MatlaПС-2022кн*
- *Особе иПВМ-эл*
- *MathWMtLoT-el*
- *MapleB-el*
- *Дьяконов В.Maple вМР-2022кн*
- *Аладьев В.З..СистеКАМИП-2006кн*
- *MaplePH-el*
- *Дьяконов В.MatheПР-2022кн*
- *Stephen-2023el*
- *tSemanRoPM-2023el*

- *Wolfram-2023эл*
- *Wolfram-2018el*
- *MatheИВ-эл*
- *MatheQRH-el*
- *Таранчук В.Б.Метод иТРПС-2019ст*
- *Таранчук В.Б.ИнтелВАВБ-2019ст*

Введение в Главу 7.4.

Важным является не только сама *Экосистема OSTIS* как форма реализации Общества 5.0, но и процесс поэтапного перехода от современной глобальной сети *компьютерных систем* к глобальной сети *ostis-систем*, то есть к *Экосистеме OSTIS*.

интеграция современных *сервисов* и *информационных ресурсов* с *Экосистемой OSTIS* является важнейшим элементом для продвижения технологических инноваций в современном мире. Поскольку спрос на эффективные, надежные и доступные системы продолжает расти, очень важно иметь комплексную и интегрированную платформу, способную удовлетворить различные потребности.

Технология OSTIS обеспечивает основу для разработки сложных систем и их беспрепятственной *интеграции* с существующими услугами и *ресурсами*. Системы, разработанные в раммках комплексной экосистемы в совокупности предоставляют возможности использовать самый разнообразный функционал: управление информацией, анализ данных, принятие решений, автоматизацию и так далее.

В данной главе рассматривается значение *интеграции Экосистемы OSTIS* с современными *сервисами* и *информационными ресурсами*, а также потенциальные преимущества такой *интеграции*. Цель данной главы — дать всесторонний анализ и представить аргументы в пользу внедрения современных систем и *сервисов* в *Экосистему OSTIS*.

§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами

⇒ *подраздел**:
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*
- *Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами*

Введение в § 7.4.1.

Процесс *интеграции Экосистемы OSTIS* с *сервисами* и *информационными ресурсами* требует глубокого понимания базовых принципов *Технологии OSTIS*. В этом параграфе рассмотрены общие принципы *интеграции Экосистемы OSTIS* с современными *сервисами* и *информационными ресурсами*, как она может быть адаптирована для интеграции с различными *сервисами* и *ресурсами*, включая *базы данных*, *веб-сервисы* и *облачные системы*.

Общие принципы *интеграции цифровой экосистемы* с современными *сервисами* и *информационными ресурсами* выглядят следующим образом (см. *Valdez O.How tDaDEaPF-2019art*, *Li W..DigitECaP-2012art*):
- *стандартизация* и *совместимость*, что достигается путем использования стандартизованных протоколов и форматов обмена данными;
- *открытость* и доступность *цифровой экосистемы* для различных участников, что обеспеченивается открытыми и удобными *интерфейсами*;
- *безопасность* и конфиденциальность, что достигается путем использования криптографических методов защиты данных и контроля доступа к ресурсам;
- *автоматизация* и *масштабируемость*, что позволит обеспечить эффективность и производительность *цифровой экосистемы* при работе с большим количеством *сервисов* и *ресурсов*;
- анализ и управление данными, что поможет определить эффективность *интеграции* и улучшить ее в дальнейшем.

Каждый из этих принципов играет важную роль в *интеграции цифровой экосистемы* с современными *сервисами* и *информационными ресурсами* и должен быть учтен при разработке и реализации интеграционных решений.

Важно отметить, что для успешной *интеграции* необходимо учитывать специфику каждой конкретной системы и ее потребностей. Это требует анализа и понимания всех особенностей и требований, которые могут повлиять на процесс *интеграции*.

В целом, для успешной *интеграции цифровой экосистемы* с современными *сервисами* и *информационными ресурсами* необходимо учитывать все указанные принципы и использовать современные технологии, такие как *Технология OSTIS*, которые позволяют реализовать эти принципы в практической работе.

Технология OSTIS представляет собой мощный инструмент для разработки и реализации *цифровых экосистем*, которые могут интегрироваться с различными *сервисами* и *информационными ресурсами*. Она обладает рядом преимуществ, таких как использование открытых стандартов и моделей, возможность масштабирования и автоматизации процессов, высокий уровень безопасности и конфиденциальности, а также удобные *интерфейсы* для работы с данными.

ostis-системы могут выполнять роль системных интеграторов различных *ресурсов* и *сервисов*, реализованных современными *компьютерными системами*, поскольку *уровень интеллекта ostis-систем* позволяет им с любой степенью детализации специфицировать интегрируемые *компьютерные системы* и, следовательно, достаточно адекватно понимать, что знает и/или умеет каждая из них. Также *ostis-системы* способны достаточно качественно координировать деятельность стороннего *ресурса* и *сервиса* и обеспечивать релевантный поиск нужного *ресурса* или *сервиса*.

Кроме того *ostis-системы* могут выполнять роль интеллектуальных help-систем — помощников и консультантов по вопросам эффективной эксплуатации различных *компьютерных систем* со сложными функциональными возможностями, имеющими *пользовательский интерфейс* с нетривиальной семантикой и использующимися в сложных предметных областях. Такие интеллектуальные help-системы можно сделать интеллектуальными посредниками между соответствующими *компьютерными системами* их пользователями.

На первых этапах перехода к Обществу 5.0 нет необходимости преобразовывать в *ostis-системы* все современные системы автоматизации некоторых видов и областей человеческой деятельности. Однако, *ostis-системы* должны взять на себя координационно-связующую роль благодаря высокому уровню своей *интероперабельности*. *ostis-системы* должны научиться либо выполнять миссию активной интероперабельной надстройки над различными современными средствами автоматизации, либо ставить перед современными средствами автоматизации выполнимые для них задачи, обеспечивая их непосредственное участие в решении сложных комплексных задач и организуя управление взаимодействием различных средств автоматизации в процессе коллективного решения сложных комплексных задач.

Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами

В контексте интеграции *Экосистемы OSTIS* с разнородными *сервисами*, под *сервисами* понимаются приложения, программы, *веб-сервисы* и другие *информационные системы*, которые предоставляют определенный функционал, механизм преобразования информации в соответствии с заданной функцией. Зачастую такое приложение может предоставить *программный интерфейс*, который можно использовать с определенным форматом входов, которым будут соответствовать определенные форматы выходов.

Под *интеграцией Экосистемы OSTIS* с *сервисом* следует понимать возможность использовать функционал *сервиса* для изменения внутреннего состояния *базы знаний Экосистемы OSTIS*.

При *интеграции сервисов* в *цифровых экосистемах* возникают ряд проблем, которые могут затруднять процесс *интеграции* и уменьшать эффективность экосистемы (см. *Valdez O.How tDaDEaPF-2019art* , *Li W..DigitECaP-2012art*). Некоторые из этих проблем могут включать в себя:

- различные форматы данных и протоколы обмена, которые могут привести к ошибкам при *обмене информацией*, что затрудняет взаимодействие между *сервисами*;
- *несовместимость* версий приложений, что может привести к конфликтам при *обмене информацией*;
- разные уровни безопасности, что может стать причиной утечки конфиденциальной информации;
- отсутствие единой точки управления, что затрудняет мониторинг и управление процессами *интеграции*;
- отсутствие механизмов для анализа и управления информацией, что затрудняет контроль над процессами *обмена информации*.

Перечисленные проблемы значительно усложняют разработку самих *сервисов* и ведет к значительному увеличению временных и материальных затрат. Для решения этих проблем при интеграции *цифровых экосистем* с различными *сервисами* и *ресурсами* используются различные подходы и технологии (см *Caldarola E..Appro tOIfORiK-2015art*, *Bork D..aOpenPfMMCtO-2019art*). Некоторые из них могут включать в себя:

- использование стандартных протоколов и форматов обмена данных, таких как XML, JSON и другие, что позволяет сделать *обмен информацией* более надежным и универсальным;

- разработка единой схемы данных и правил доступа, что позволяет сделать *интеграцию* более простой и управляемой;
- реализация механизмов для автоматической обработки ошибок и конфликтов, что позволяет снизить количество ошибок и улучшить надежность *цифровой экосистемы*;
- использование инструментов и технологий для анализа и управления информацией, таких как системы бизнес-аналитики и управления информацией, что позволяет контролировать процессы *обмена информации* и оптимизировать их работу.

В рамках *Экосистемы OSTIS* выделено несколько уровней *интеграции*, которые позволяют взаимодействовать с различными *информационными ресурсами* и *сервисами*.

Полная *интеграция* предполагает исполнение функции *сервиса* на платформонезависимом уровне, где вся программа исполняется в самой *базе знаний Экосистемы OSTIS*. То есть, задача интеграции такого *сервиса* сводится к выделению алогиртма обработки графовой конструкции и его реализации в рамках *базы знаний*. В результате такой полной интеграции надобность использовать сторонний *сервис* отпадает.

Частичная *интеграция* предполагает реализацию взаимодействия и изменения состояния *базы знаний Экосистемы OSTIS* на этапах исполнения функции *сервиса*. Степень глубины *интеграции* может различаться, в некоторых случаях *сервис* может обращаться к *базе знаний* для получения дополнительной информации или для записи промежуточных результатов.

В простейшем случае изменение *базы знаний Экосистемы OSTIS* может происходить единожды, после получения результата отработки функции *сервиса*. На основе такого принципа можно выделить специальные *sc-агенты*, использующие сторонний *сервис*.

Для обеспечения *интеграции* функционального *сервиса* необходимо выполнение следующих минимальных требований:
- спецификация входной конструкции в *базе знаний* системы: определение структуры в *базе знаний* системы, которая будет преобразовываться в формат данных, совместимый с *сервисом*;
- спецификация выходной конструкции в *базе знаний* системы: определение структуры в *базе знаний* системы, которая будет формироваться из исходной структуры впоследствии преобразования данных *сервиса* в знания;
- реализация *sc-агента*, который преобразует конструкцию *базы знаний* в формат, который может быть использован в *сервисе*, а также погружать результаты работы *сервиса* обратно в *базу знаний* системы в соответствии со спецификацией.

Удовлетворение данных требований позволит обеспечить эффективную *интеграцию* функционального *сервиса*, что, в свою очередь, позволит использовать данные и функциональность *сервиса* в различных *ostis-системах*. Задача формирования спецификации рассмотрена в *§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем*.

Обобщенный алгоритм *sc-агента*, использующего сторонний *сервис* для *интеграции* функциональных возможностей, может быть описан следующим образом:
- извлечение из *базы знаний* необходимых структур знаний, соответствующих требованиям функционального *сервиса*;
- преобразование извлеченных знаний в формат, необходимый для подачи на вход функциональному *сервису*;
- отправка запроса на функциональный *сервис* и ожидание его ответа;
- формирование структур знаний на основе полученных данных от функционального *сервиса*;
- погружение новых структур знаний в *базу знаний интеллектуальной системы*, с целью обеспечения их дальнейшей использования.

Следует учесть, что при *интеграции* функциональных *сервисов*, возможно потребуется проводить дополнительную обработку и преобразование данных, например, для обеспечения их *совместимости* с форматами данных, или для обеспечения безопасности передачи и хранения данных.

Таким образом, внедрение возможности использования стороннего *сервиса* в *Экосистеме OSTIS* предполагает выполнение следующих шагов:
- анализ требований к интегрируемому *сервису*, определение необходимого функционала, форматов входных и выходных данных, и других характеристик сервиса.
- разработка спецификации *интеграции*, которая будет определять форматы данных и правила взаимодействия между *базой знаний Экосистемы OSTIS* и сторонним *сервисом*;
- разработка *sc-агента*, который будет обеспечивать взаимодействие между *базой знаний* и сторонним *сервисом* в соответствии со спецификацией *интеграции*;
- тестирование и отладка;
- внедрение в *Экосистему OSTIS*, что позволит использовать возможности интегрированного стороннего *сервиса* в различных *ostis-системах*.

Использование описанного подхода описано в работе *Kroshchanka A..aNeuraSAtCV-2022art* на примере интеграции сервисов компьютерного зрения с ostis-системой.

Для подключения *sc-агента* используются различные подходы, один из которых заключается в подключении *sc-агента* в рамках уже существующей, основной *ostis-системы*. С точки зрения масштабируемости реализации такого подхода следует отметить *монолитную архитектуру* получаемой *ostis-системы*, что позволяет упростить процесс внедрения новых *сервисов* и *sc-агентов* в *ostis-систему*.

Преимуществом такого подхода является более простое и удобное внедрение новых *сервисов* и *sc-агентов* в *ostis-систему*, а также упрощение процесса управления зависимостями. Использование реализации ostis-системы с *монолитной архитектурой* может быть применено в случаях, когда функциональный *сервис* не нуждается в частой модификации и обладает достаточно простой структурой. Кроме того, *монолитная архитектура* может быть более удобна в случаях, когда обращение к *сервису* происходит через внутреннюю сеть и требует низкой задержки и высокой производительности. Так могут быть интегрированы и *сервисы* получения знаний из внешних источников (получение прогноза погоды, обработка статистической информации, и так далее), и функциональные *сервисы* (обработка аудиоинформации и погружение результатов обработки в *базу знаний*, получение синтаксического анализа предложения).

Альтернативным вариантом внедрения является реализация отдельной *ostis-системы*, в рамках которой будет интегрирована функция *сервиса*. Это позволяет перейти к использованию *микросервисной архитектуры*, что характеризуется распределенным взаимодействием *ostis-систем*.

Преимуществами такого подхода является большая *гибкость* и возможность *масштабирования*. Подход характеризуется высокой степенью распределенности, децентрализованности и доступности нового функционала. Система выходит за рамки технических ограничений, функционал может быть распределен на различном аппаратном обеспечении. Полученный функционал может быть использован различными *ostis-системами* в рамках *Экосистемы OSTIS* для достижения своих целей. Минусами такой системы является сложность разработки, а также увеличение временных затрат на общение систем друг с другом по сетевым протоколам.

микросервисную архитектуру ostis-систем предпочтительно использовать в случаях, когда функциональный *сервис* обладает сложной структурой, а также в случаях, когда требуется масштабирование и гибкость всей системы. В качестве примера можно привести *сервис*, который взаимодействует с внешними источниками данных и может быть подвержен частым изменениям. Примерами *интеграции* на основе *микросервисной архитектуры* могут служить *сервисы* считывания эмоционального состояния пользователя, естественно-языковая обработка, задачи классификации и идентификации, и так далее.

Таким образом, выбор подхода к *интеграции* функциональных *сервисов* зависит от конкретных требований и условий проекта. Использование *Технологии OSTIS* позволяет создавать гибкие и эффективные системы, которые могут быть адаптированы под различные потребности и требования пользователей.

Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами

⇒ *библиографическая ссылка**:
- *RDFCAS-2023el*
- *RDB tRML-2012el*
- *EasilGHKG-2022el*
- *ЧтоТОД-2023эл*

Существует несколько причин, по которым следует интегрировать интеллектуальную систему с информационными источниками:
- Обеспечение полноты и точности данных: Интеллектуальная система создается для обработки больших объемов данных и принятия решений на их основе. Информационные источники являются основой для этих данных, и интеграция системы с ними гарантирует полноту и точность данных.
- Уменьшение времени и усиление эффективности работы: Интеграция информационных источников в интеллектуальную систему обеспечивает быстрый и удобный доступ к нужной информации. Это уменьшает время на поиск и обработку данных, что повышает эффективность работы системы и уменьшает количество ошибок.
- Расширение возможностей системы: Информационные источники обладают большим количеством данных, которые могут быть полезны для работы интеллектуальной системы. Интеграция системы с различными источниками расширяет возможности системы и позволяет ей повысить качество работы.
- Повышение надежности: Разнообразные источники данных обеспечивают резервирование и возможность сравнения, что позволяет интеллектуальной системе работать более надежно и безопасно в случае сбоя одного или нескольких источников.

- Улучшение качества прогнозирования: Интеграция информационных источников с интеллектуальной системой способствует улучшению качества прогнозирования, так как позволяет объединять данные из различных источников и анализировать их вместе для получения более точных результатов.

Существует большое количество информационных источников, из которых можно получать информацию. Основными можно назвать следующие:
- Интернет — сайты, блоги, форумы, социальные сети, новостные порталы и другие ресурсы в сети.
- Книги и учебники — доступные в библиотеках, книжных магазинах или в электронном формате.
- СМИ — телевизионные программы, радио, газеты, журналы и другие источники новостей.
- Официальные документы и отчеты — включая законы, правительственные статистические данные, отчеты об исследованиях и другие официальные документы.

Принципы интеграции *Экосистемы OSTIS* со структурированными информационными ресурсами основаны на пополнении базы знаний системы новыми знаниями. Один из востребованных подходов в этом направлении — интеграция с ресурсами на основе RDF (Resource Description Framework, см. *RDFCAS-2023el*), который является моделью данных, предложенной консорциумом W3C.

Для успешной интеграции структурированных информационных ресурсов в Экосистему OSTIS, важно уделить должное внимание пониманию и применению принципов RDF-модели, поскольку они играют ключевую роль в организации связей между различными ресурсами. RDF используется для описания ресурсов в сети Интернет, и является основой для построения семантических веб-приложений, таких как Linked Open Data.

Основная структура абстрактного синтаксиса RDF — это тройка, состоящая из субъекта, предиката и объекта. Набор таких троек называется графом RDF. Граф RDF может быть визуализирован как диаграмма узла и направленной дуги, в которой каждая тройка представлена как связь “узел — дуга — узел”.

Графы RDF атемпоральны, то есть представляют собой статические снимки информации. Однако графы RDF могут выражать информацию о событиях и временных аспектах других сущностей, учитывая соответствующие термины из словаря. Поскольку графы RDF определены как математические наборы, добавление или удаление троек из графа RDF дает другой граф RDF.

Узел может иметь следующий тип:
- IRI. Представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс на любом языке мира. IRI представляе собой обобщение URI;
- Литерал. Представляя собой структуру, состоящую из лексической формы (UNICODE-строка) и типа данных;
- Пустой узел. Представляет собой локальный идентификатор, который используются в некоторых конкретных синтаксисах RDF или реализациях хранилища RDF.

RDF поддерживает основные типы данных, такие как строковый (string), логический (boolean), числовые (integer, double, float и другие), временные и некоторые другие.

В RDF существует такое понятие, как словарь RDF. Он представляет собой совокупность IRI, ссылающихся на другие графы с классами, литералами и так далее. Часто группа IRI может начинаться с одинакового префикса.

RDF нашел широкое применение. Так, например, RDF используется в оформлении *баз знаний* в рамках различных проектов во множестве институтов, университетов и иных организаций. Поисковые системы предлагают веб-мастерам использовать RDF и аналогичные языки разметки страниц для повышения информативности ссылок на их сайты в результатах поиска. Социальные сети, с подачи Facebook, предлагают веб-мастерам использовать RDF для описания свойств страниц, так же позволяющих красиво оформить ссылку на нее в записи пользователя социальной сети.

В ходе анализа были выявлены следующие подходы к интеграции информационных ресурсов на основе RDF с другими системами:
- R2RML (см. *RDB tRML-2012el*) — это стандарт W3C для выражения настраиваемых отображений из реляционных БД в RDF. Такие отображения предоставляют возможность просматривать существующие реляционные данные в модели данных RDF, выраженные в структуре и целевом словаре по выбору автора сопоставления;
- R2RML.io (см. *EasilGHKG-2022el*) — это open-source проект, разрабатываемый с 2013 года. Данная технология предназначена для генерации базы знаний на основе данных из полуструктурированных источников;
- “Озеро данных” (см. *ЧтоТОД-2023эл*) — это централизованное хранилище, которое позволяет хранить все структурированные и неструктурированные данные в любом масштабе. “Семантическое озеро данных” — это особая форма озер данных, в которых верхний семантический слой обогащает и связывает данные семантически. Семантический уровень преодолевает разрозненность данных и обеспечивает семантический поиск по всем данным.

Интеграция ostis-систем с внешними информационными ресурсами удобна по многим причинам. Технология OSTIS изначально предлагает инструменты для описания синтаксиса и семантики внешних языков (см. *Главу 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*). Данный инструментарий позволяет сократить время разработки в несколько раз. Также из достоинств можно выделить:

- способность осуществлять интеграцию знаний в своей памяти на высоком уровне;
- возможность интегрировать различные виды знаний;
- возможность интегрировать различные модели решения задач.

Для интеграции информационного ресурса на основе RDF в Экосистему OSTIS был реализован соответствующий *абстрактный sc-агент*. Его работу можно разбить на следующие этапы:
- интеграция с использованием готовых правил;
- интеграция с сохранением исходной схемы;
- дополнительные преобразования.

Интеграция с использованием готовых правил

На этом этапе ко всем сгенерированным тройкам применяются готовые правила интеграции, хранящиеся в *базе знаний*. Создание и применение подобных правил необходимо в ситуациях, когда способ представления конкретного знания во внешнем информационном ресурсе по какой-то причине не соответствует представлению аналогичного знания в ostis-системе.

Интеграция с сохранением исходной схемы

На данном этапе оставшиеся тройки будут преобразованы с сохранением той структуры отношения, в которой находились участвовавшие в нем сущности. Это значит, что порядок элементов в итоговой конструкции будет аналогичен порядку сущностей в исходной.

Дополнительные преобразования

На данном этапе проходят оставшиеся интеграционные преобразования, которым не нашлось места в предыдущих пунктах, но которые необходимы для завершения процесса интеграции.

Для выгрузки информации из *базы знаний* в какой-либо внешний формат можно использовать те же правила, что и для загрузки, так как в основном они представляют собой утверждения об *эквиваленции*. То есть изначально производится поиск необходимых конструкций, затем они к ним применяется соответствующее правило, и, в результате получается множество троек. В дальнейшем данные тройки преобразуются в необходимый формат.

§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS

⇒ *ключевой знак**:
- *Maxima*
- *MATLAB*
- *Mathematica*
- *Maple*
- *Wolfram*
- *Wolfram Mathematica*
- *Wolfram Alpha*
- *Wolfram Language*

⇒ *ключевое понятие**:
- *компьютерная алгебра*
- *система компьютерной математики*
- *система компьютерной алгебры*
- *системы для численных вычислений*

⇒ *ключевое знание**:
- *Способы интеграции систем компьютерной алгебры с Экосистемой OSTIS*
- *Принципы интеграции систем компьютерной алгебры с Экосистемой OSTIS*

⇒ *библиографическая ссылка**:
- *Дьяконов В.ЭнцикКА-2022кн*
- *Аладьев В.З..Введе вСПМ22-1999кн*
- *Аладьев В.З..МодулПМvMaVV-2011кн*
- *List oCAS-2023el*
- *СистеКАОС-эл*
- *СистеКАМ-эл*
- *Стахин Н.А.ОсновРсСАС-2008кн*
- *AXIOM.tSCS-эл*
- *Дьяконов В.MatlaПС-2022кн*

- *Особе иПВМ-эл*
- *MathWMtLoT-el*
- *MapleB-el*
- *Дьяконов В.Maple вМР-2022кн*
- *Аладьев В.З..СистеКАМИП-2006кн*
- *MaplePH-el*
- *Дьяконов В.МатheПР-2022кн*
- *Stephen-2023el*
- *tSemanRoPM-2023el*
- *Wolfram-2023эл*
- *Wolfram-2018el*
- *МатheИВ-эл*
- *MatheQRH-el*
- *Таранчук В.Б.Метод иТРПС-2019ст*
- *Таранчук В.Б.ИнтелВАВБ-2019ст*

⇒ *подраздел**:
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*
- *Пункт 7.4.2.3. Пример интеграции прототипа обучающей ostis-системы по дискретной математике и Wolfram Mathematica*

Введение в § 7.4.2.

В середине XX века на стыке математики и информатики возникло и интенсивно развивается фундаментальное научное направление — ***компьютерная алгебра***, наука об эффективных алгоритмах вычислений математических объектов. Синонимами термина *"компьютерная алгебра"* являются: *"символьные вычисления"*, *"аналитические вычисления"*, *"аналитические преобразования"*, а иногда и *"формальные вычисления"*. Направление "компьютерная алгебра" представлено теорией, технологиями, программными средствами. К прикладным результатам относят разработанные алгоритмы и программное обеспечение для решения с помощью компьютера задач, в которых исходные данные и результаты имеют вид математических выражений, формул. Основным продуктом компьютерной алгебры стали программные *системы компьютерной алгебры* — ***с.к.а.*** (англ. Computer Algebra System, CAS).

Круг математических задач, которые можно решить с помощью *с.к.а.*, непрерывно расширяется. Значительные усилия исследователей направлены на разработку алгоритмов вычисления топологических инвариантов многообразий, узлов, алгебраических кривых, когомологий различных математических объектов, арифметических инвариантов колец целых элементов в полях алгебраических чисел. Другое направление современных исследований — квантовые алгоритмы, имеющие иногда полиномиальную сложность, тогда как существующие классические алгоритмы имеют экспоненциальную.

Исследования и разработки теоретических основ и технологий реализации методов и программных реализаций инструментов компьютерной алгебры продолжаются. Термины, определения, названия в описаниях функций и инструментов этих систем также претерпевают изменения, некоторые формулировки, ранее приводимые в отдельных руководствах, обзорах возможностей инструментария не только уточняются, но и изменяются. Это нормально для новых научных направлений и технологий. Читатель не должен удивляться, если в других источниках встретит иные формулировки, термины.

Не следует отождествлять *системы компьютерной алгебры* и *системы компьютерной математики* (***с.к.м.***), которые в ряде изданий условно делятся на две категории: *системы компьютерной алгебры* и *системы для численных вычислений*. Обычно к *с.к.м.* относят:
- табличные процессоры, например, *Microsoft Excel*, *Lotus Symphony Spreadsheets*, *Gnumeric, OpenOffice.org Calc*;
- системы для статистических расчетов, например, *STATISTICA*, *PASW Statistics* (первоначальное название *SPSS Statistics*);
- системы для моделирования, анализа и принятия решений, например, GPSS, AnyLogic, DSS;
- системы компьютерной алгебры;
- универсальные математические системы.

Если с выделением в отдельные группы первых трех из перечисленных *с.к.м.* есть согласие большинства авторов, то отнесение в отдельную группу универсальных математических систем прослеживается относительно редко. Основные *с.к.а.* в контексте численных и аналитических вычислений большинством авторов считаются универсальными системами.

При развитии технологий на первом месте в разработке и модернизации *и.к.с.* должны быть решения, реализации интеграции таких систем и средств *систем компьютерной алгебры*. Важно превращение современного многообразия инструментальных средств (frameworks) разработки различных компонентов *и.к.с.* в единую технологию комплексного проектирования и поддержки полного жизненного цикла этих систем, гарантирующую совместимость всех разрабатываемых компонентов, а также совместимость самих *и.к.с.* как самостоятельных субъектов, взаимодействующих между собой в рамках комплексных систем автоматизации сложных видов коллективной человеческой деятельности. Необходима конвергенция и унификация интеллектуальных компьютерных систем нового поколения и их компонентов (см. *Главу 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*, *Главу 1.2. Интеллектуальные компьютерные системы нового поколения*, *Главу 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*). При этом, под конвергентными решениями, в основном, подразумеваются оптимизированные комплексы, содержащие в себе все необходимое для решения задач Искусственного интеллекта, организованные или сконфигурированные для эффективного использования информационных ресурсов, для упрощения процессов внедрения; в том числе, должны обеспечиваться возможности решения определенных задач с требованиями оптимизации и достижения максимальной производительности, и во всех реализациях — оптимизированные для простоты использования. Перечисленное в полной мере относится к *Экосистеме OSTIS*.

Подобные проблемы решаются и при разработке, совершенствовании, систематическом обновлении содержания и расширения возможностей систем компьютерной алгебры. Таким образом интеграция *с.к.а.* и *Экосистемы OSTIS* является важной и актуальной задачей.

Одним из вариантов взаимодействия *Экосистемы OSTIS* и *с.к.а.* могут быть подходы, аналогичные реализуемым в рамках интеграции в ostis-системы искусственных нейронных сетей (см. *Главу 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*). Можно рассматривать следующие методические и технические решения:

- Интеграция по принципу "черного ящика", когда в *базе знаний ostis-системы* присутствует спецификация используемой функции ядра системы компьютерной алгебры, а также спецификация способа вызова данной функции (например, указание через какой программный интерфейс осуществляется взаимодействие с данной внешней системой). Такой вариант интеграции является наиболее простым в плане реализации и в целом обладает перечисленными выше достоинствами. В то же время, данный вариант обладает и недостатком, связанным с тем, что *ostis-система* не содержит средств анализа и объяснения того, как был выполнен конкретный шаг решения задачи, реализуемый используемой функцией *с.к.а.*
- Более тесная интеграция, при которой конкретная функция по-прежнему остается частью сторонней *с.к.а.*, когда в *базу знаний ostis-системы* погружается не просто результат ее выполнения, а и всевозможная его спецификация, например, объяснение хода решения задачи, указание конкретных алгоритмов и формул, которые могут быть задействованы в решении, описание возможных альтернативных вариантов решения, оценка эффективности решения и так далее. В данном варианте интеграции *ostis-система* получает больше возможностей по анализу и объяснению хода решения задачи. (Следует отметить, что конкретно в *с.к.а. Wolfram Mathematica* уже присутствуют подробные пояснения хода решения задачи и допустим режим пошагового выполнения).
- Полная интеграция, при которой осуществляется трансляция используемых функций системы компьютерной алгебры с внутреннего языка этой системы в *ostis-систему*. Данный вариант является наиболее трудоемким и сложным с точки зрения актуализации реализации возможностей систем компьютерной алгебры в соответствующих *ostis-системах* с учетом их постоянного развития. В то же время такой вариант интеграции по сравнению в двумя предыдущими обладает важным достоинством — он обеспечивает платформенную независимость полученного решения и позволяет использовать при решении конкретной задачи все достоинства предлагаемых в рамках *Технологии OSTIS* подходов, в частности, возможность многопользовательской, параллельной обработки знаний и возможность оптимизации плана решения задачи или его фрагментов непосредственно в ходе решения.

С прикладной точки зрения на данном этапе развития и применения *Экосистемы OSTIS* представляется целесообразной интеграция в едином комплексе возможностей *систем компьютерной алгебры* (*с.к.а.*) и построенных в рамках *Экосистемы OSTIS* интеллектуальных обучающих систем (см. *Главу 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*), что обусловлено содержанием *систем компьютерной алгебры*, которые обладают несомненным преимуществом и широкими возможностями при решении актуальных для обучающих систем задач практически по всем естественно-научным и техническим дисциплинам, предполагающим использование сложного математического аппарата.

С другой стороны, несмотря на популярность тематики, связанной с автоматизацией и интеллектуализацией образовательной деятельности по естественно-научным дисциплинам и разработкой соответствующих компьютерных систем, на данный момент на рынке практически отсутствуют апробированные интеллектуальные обучающие системы, способные самостоятельно генерировать и решать различные задачи, а также проверять корректность решения задачи пользователем. В качестве прототипов можно привести отдельные системы, в которых решаются нетривиальные задачи, например, по геометрии [https://geometry.allenai.org/, https://mathpix.com/handwriting-recognition]

и теории графов [https://graphonline.ru/]. В то же время следует отметить, что в упомянутых системах нет как таковой "интеллектуальности" (по факту заложен только конкретный набор действий, в самих приложениях задачи не генерируются), нет средств проверки решений, имеющих даже незначительные отклонения правил оформления.

Подход к решению задач интеллектуализации образовательной деятельности, основанный на интеграции ostis-систем и систем компьютерной алгебры, обладает рядом преимуществ:

- При разработке *ostis-систем* исключается необходимость программировать многие функции, которые уже реализованы, оттестированы и апробированы в *с.к.а.* Это принципиально, так как системы компьютерной алгебры разрабатываются высококвалифицированными специалистами в соответствующих областях, реализация аналогичных функций в ostis-системах может потребовать значительных финансовых и временных затрат.
- Конкретная *ostis-система*, использующая отдельные функции *с.к.а.*, благодаря подходу к разработке гибридных *решателей задач* в *Технологии OSTIS* (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*) получает возможность самостоятельно планировать ход решения задач при условии, что некоторые его этапы будут реализованы при помощи присоединяемых функций. С точки зрения подхода, предлагаемого в рамках *Технологии OSTIS*, каждая функция системы компьютерной алгебры становится *методом* решения задач некоторого класса. Этот класс задач описывается в *базе знаний ostis-системы* и позволяет ей при решении конкретной задачи самостоятельно делать вывод о целесообразности применения той или иной функции *с.к.а.* Такая интеграция с *ostis-системами* позволит устранить сформулированный ранее возможный недостаток *систем компьютерной алгебры* (определяется тем, какие *с.к.а.* используются — отдельно поясняется ниже в обзоре систем компьютерной математики, условий их применения и доступа к отдельным компонентам).

Особо отметим, что обозначенные варианты интеграции не исключают друг друга и могут комбинироваться. Кроме того, углубление интеграции может осуществляться поэтапно с учетом перечисленных достоинств и недостатков, а также с учетом актуальности использования тех или иных функций систем компьютерной алгебры при решении конкретных задач в рамках *Экосистемы OSTIS* и соответствующих *ostis-систем*.

Поэтапная интеграция *с.к.а.* с *Экосистемой OSTIS* предполагает, как минимум, описание спецификации основных функций выбранной системы компьютерной алгебры средствами *Технологии OSTIS*, другими словами — разработку онтологии внешних функций. В случае с системами семейства *Wolfram Mathematica* процесс разработки такой онтологии может быть автоматизирован благодаря наличию формального языка *Wolfram Language* и хорошей документированности функций системы.

Обобщая изложенное, констатируем — интеграция обучающих систем, разрабатываемых на базе *Технологии OSTIS*, и систем компьютерной алгебры позволит создавать системы, обладающие высоким уровнем интеллекта, в более сжатые сроки, причем, с использованием тщательно отработанных (математически, алгоритмически) и многократно апробированных инструментов.

Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры

Основное назначение *с.к.а.* — работа с математическими выражениями в символьной форме. Базовые типы данных *с.к.а.*: числа и математические выражения. Числа: короткие и длинные целые (одинарной и кратной точности), рациональные, комплексные, алгебраические числа. Алгебраическое число задается своим минимальным многочленом, а иногда для его задания требуется указать интервал на прямой или область в комплексной плоскости, где содержится единственный корень данного многочлена. Математические выражения: арифметика, функции, уравнения, производные, интегралы, векторы, матрицы, тензоры. Кроме того, в компьютерной алгебре рассматриваются такие объекты, как: функциональные, дифференциальные поля, допускающие показательные, логарифмические, тригонометрические функции; матричные кольца и другие. Перечислим основные отличительные признаки систем компьютерной алгебры (см. *Дьяконов В.ЭнцикКА-2022кн* , *Аладьев В.З..Введе вСПМ22-1999кн* , *Аладьев В.З..МодулПМvMaVV-2011кн*, *List oCAS-2023el*).

с.к.а. работают следующим образом:

- математические объекты (алгебраические выражения, ряды, уравнения, векторы, матрицы и так далее) и указания, что с ними делать, задаются пользователем на входном языке системы в виде символьных выражений;
- интерпретатор анализирует и переводит символьные выражения во внутреннее представление;;
- символьный процессор системы выполняет требуемые преобразования или вычисления и выдает ответ в математической нотации.

Алгоритмы внутренних преобразований имеют алгебраическую природу, что и отражено в названии систем — системы компьютерной алгебры.

Содержание техники символьных вычислений:

- внутреннее представление математического выражения в системе символьных вычислений — синтаксическое дерево (список списков);
- суть символьных вычислений (аналитических преобразований) — переписывание терма с помощью последовательного применения правил из определенного пользователем или системой списка;
- преобразование из внешнего представления во внутреннее и обратно обеспечивается дополнительными инструментальными средствами.

Далеко не каждая математическая задача имеет определяемое существующими математическими формализмами аналитическое решение; есть алгоритмически неразрешимые задачи; в исследованиях проблемы оценки трудоемкости алгоритмов алгоритмически разрешимых задач при наличии принципиальных достижений остается значительное число вопросов. Специалисты в областях прикладной и компьютерной математики единодушны во мнении, что много практически важных задач и не могут быть формализованы настолько, чтобы решаться аналитически, в лучшем случае они могут решаться только численными методами.

Достаточно полный перечень с указанием функциональности систем символьных вычислений и платформ, на которых эксплуатируются, можно найти в (см. *List oCAS-2023el*). Классификационными признаками *с.к.а.* являются: функциональное назначение, тип архитектуры, средства реализации, области применения, интегральные оценки качества. Отметим несколько наиболее часто упоминаемых классов.

с.к.а. общего назначения и специализированные. Наиболее известные системы из первой группы (обеспечивают решение задач для большинства основных разделов символьной математики): *Derive*, *Mathematica*, *Maple*, *Macsyma* и ее потомок *Maxima*, *Scratchpad* и ее потомок *Axiom*, *Reduce*, *MuPAD*, *Sage*, *SMath Studio*, *Yacas*, *Scientific WorkPlace*, *Kalamaris*. Системы для решения задач одного или нескольких смежных разделов символьной математики — это специализированные с.к.а. Примерами таковых являются: *GAP* (теория групп), *Cadabra* (тензорная алгебра), *KANT* (алгебра и теория чисел), *Singular* (полиномиальные вычисления с акцентом на нужды коммутативной алгебры, алгебраическая геометрия), *Calc3D* (для работы с 3D матрицами, векторами, комплексными числами), *GRTensorII* (дифференциальная геометрия).

Классы с.к.а. по типу архитектуры. В *СистеКАОС-эл* предлагается следующее разделение:

- *с.к.а.* классической архитектуры: системное ядро + прикладные расширения, примеры: *Axiom*, *Maple*, *Mathematica*;
- Программный пакет для расширения базовой прикладной математической системы, примеры: ядро *Maple* для *MATLAB* и *MathCAD*;
- Встраиваемое расширение (плагин) для языка и / или системы программирования, примеры: MathEclipse / Symja — Java-библиотека;
- *Open Source*, *GNU GPL*, мультиплатформные *с.к.а.*, примеры: *Maxima* (Lisp), *PARI/GP* (C).

Типовая структура с.к.а.

Составляющие *с.к.а.*:

- ядро системы — содержит машинные коды реализаций операторов и встроенных функций *с.к.а.*, обеспечивающих выполнение аналитических (символьных) преобразований математических выражений на основе системы определенных правил;
- интерфейсная оболочка — обеспечивают поддержку всех функций, необходимых для информационных и управляющих взаимодействий между *с.к.а.* и пользователями (людьми, программами, аппаратными средствами);
- библиотеки специализированных программных модулей и функций — содержат каталогизированные (по типам обрабатываемых абстрактных объектов — числа, функции, алгебры и тому подобные и/или методам вычислений — аналитические, численные, смешанные) реализации алгоритмов решения типовых математических задач; они функционально расширяют ядро *с.к.а.*;
- пакеты расширения — обеспечивают различные формы адаптации *с.к.а.* к классам математических задач, внешнему ПО (операционным системам, графическим пакетам и тому подобных) и целям пользователей;
- справочная система — содержит описание функциональных возможностей и примеров работы в *с.к.а.*, информационные сообщения о текущем состоянии системы, а также сведения о математических основах алгоритмов *с.к.а.*

Функции ядра всегда тщательно отлажены, как правило, реализуются на машинно-ориентированном языке, так как требуется высокая производительность их выполнения. У некоторых *с.к.а.* оптимизация машинного кода обеспечивается, в том числе, с помощью частичной реализации функциональности на языке ассемблера или аппаратно. Ядро содержит реализации операторов и встроенных функций, обеспечивающих выполнение аналитических преобразований математических выражений на основе системы определенных правил. Объем ядра обычно ограничивают, но к нему добавляют библиотеки дополнительных процедур и функций. Распределение состава поддерживаемых системой алгоритмов символьных вычислений между ядром и библиотеками осуществляется по принципу баланса производительности и функциональности с учетом текущего состояния наиболее распространенного аппаратного обеспечения. У большинства коммерческих *с.к.а.* алгоритмы вычислений и программные модули ядра являются ноу-хау разработчиков и относятся к разряду тщательно скрываемых данных.

Интерфейсные оболочки обеспечивают поддержку всех функций, необходимых для информационных и управляющих взаимодействий между системой и пользователями, в том числе ввод, редактирование, сохранение, обмен программами, использование разных аппаратных средств. У большинства *с.к.а.* интерфейсные оболочки разные для разных операционных систем, при этом ведущие системы компьютерной алгебры работают без перекомпилирования исходного кода, как на различных аппаратных платформах, так и под управлением разных операционных систем; пользовательские интерфейсы обеспечивают похожие визуальные сценарии работы в *с.к.а.* на разных компьютерах, в разных операционных системах и обычно реализуются в видах: текстовые (поле ввода символьных строк, поле вывода символьных строк), графические (ячейки/секции ввода данных / вывода результатов, окна отображения графиков), командные (меню и кнопки управления *с.к.а.*, панели библиотек функций, индикаторы состояний *с.к.а.*).

Библиотеки специализированных программных модулей и функций, пакеты расширения содержат систематизированные по назначению реализации алгоритмов обработки абстрактных объектов, решения типовых математических задач. Библиотеки и пакеты функционально расширяют ядро, а также обеспечивают возможности программирования алгоритмов не только на языке самой системы, но и на языке ее реализации, а у многих *с.к.а.* и на основных языках программирования высокого уровня.

Справочная система всех *с.к.а.* содержит и обеспечивает пользователей описаниями функциональных возможностей и демонстрационными примерами работы, информационными сообщениями о текущем состоянии системы, а также сведениями о математических основах алгоритмов. Справедливо утверждение, что многие *с.к.а.*, по сути, являются не только инструментами для получения и анализа решений, но и математическими энциклопедиями. Для *с.к.а.* типичны организация и обеспечение диалога получения справок пошагово с вложенными уровнями абстракции и/или конкретизации информации. Обычно пользователю доступны: краткая контекстная справка о функциональном назначении выбранного элемента, информация о синтаксисе и семантике операторов и функций языка с поясняющими примерами, описание реализованных вариантов решения. Информативность справочной системы обеспечивается обязательным описанием всех функций ядра, инструментами поиска сведений об объекте *с.к.а.* по имени, тематическому разделу, ключевым словам. У многих в системе помощи содержатся обучающие материалы с разделением по категориям пользователей, интерактивные учебные курсы решения математических задач в среде системы, некоторые также имеют консультант-репетитора, выполняющего пошаговое решение примеров с поясняющими комментариями.

с.к.а. позволяют реализовывать с использованием компьютера аналитические и численные методы решения задач, представляя результаты в математической нотации, обеспечивают графическую визуализацию, оформление результатов и подготовку к изданию. Используя *с.к.а.* и компьютер, можно выполнять в аналитической форме:
- упрощение выражений или приведение к стандартному виду;
- подстановки символьных и численных значений в выражения;
- выделение общих множителей и делителей;
- раскрытие произведений и степеней, факторизацию;
- разложение на простые дроби;
- нахождение пределов функций и последовательностей;
- операции с рядами;
- дифференцирование в полных и частных производных;
- нахождение неопределенных и определенных интегралов;
- анализ функций на непрерывность;
- поиск экстремумов функций и их асимптот;
- операции с векторами;
- матричные операции;
- нахождение решений линейных и нелинейных уравнений;
- символьное решение задач оптимизации;
- алгебраическое решение дифференциальных уравнений;
- интегральные преобразования;
- прямое и обратное быстрое преобразование Фурье;
- интерполяция, экстраполяция и аппроксимация;
- статистические вычисления;
- машинное доказательство теорем.

Если задача имеет точное аналитическое решение, пользователь *с.к.а.* может получить это решение в явном виде (разумеется, речь идет о задачах, для которых известен алгоритм построения решения).

Также большинство *с.к.а.* обеспечивают:
- числовые операции произвольной точности;
- целочисленную арифметику для больших чисел;
- вычисление фундаментальных констант с произвольной точностью;
- поддержку функций теории чисел;

- редактирование математических выражений в двумерной форме;
- построение графиков аналитически заданных функций;
- построение графиков функций по табличным значениям;
- построение графиков функций в двух или трех измерениях;
- анимацию формируемых графиков разных типов;
- использование пакетов расширения специального назначения;
- программирование на встроенном языке;
- автоматическую формальную верификацию;
- синтез программ.

В *с.к.а.* можно производить вычисления в арифметике с плавающей точкой и указывать точность, реализована точная рациональная арифметика, то есть можно производить численные расчеты без потери точности. К особенностям *с.к.а.* относят преимущественно интерактивный характер работы — пользователь не знает заранее ни размера, ни формы результатов и поэтому должен иметь возможность корректировать ход вычислений на всех этапах, задавать режим пошагового выполнения с выводом промежуточных результатов.

Большинство *с.к.а.* в современной реализации не только применимы для исследования различных математических и научно-технических задач с использованием встроенных и дополнительных функций, но и содержат все составляющие языков программирования — де факто являются проблемно ориентированными языками программирования высокого уровня.

Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры

Лидерами *с.к.а.* являются *Mathematica* и *Maple* — мощные системы с собственными ядрами, оснащенные развитым пользовательским интерфейсом и обладающие разнообразными графическими и редакторскими возможностями. Широкое распространение в настоящее время имеют и *с.к.а.*: *Derive*, *Maxima*, *Axiom*, *Reduce*, *MuPAD*. Особое место занимают системы компьютерной математики *MATLAB*, *MathCad*.

Некоммерческие универсальные с.к.а.

Отличительной чертой современного состояния информационных технологий является то, что коммерческие программные продукты во многих случаях полностью или частично можно заменить некоммерческим программным обеспечением, аналогами с открытым исходным кодом — свободными программами. К таковым относят программные продукты, которые с изменениями или без них не имеют ограничений применения, копирования и передачи другим пользователям, за плату или безвозмездно. Ниже упоминаются программные средства, публикуемые под лицензией *GPL*. Поясним, что это предполагает. *GPL* предоставляет получателям компьютерных программ следующие права ("свободы"): свободу запуска программы с любой целью; свободу изучения того, как работает программа, а также ее модификации (предварительным условием для этого является доступ к исходному коду); свободу распространения копий исходного и исполняемого кода; свободу улучшения программы и выпуска улучшений в публичный доступ. В общем случае распространитель программы, полученной на условиях *GPL*, либо программы, основанной на таковой, обязан предоставить получателю возможность работать с соответствующим исходным кодом.

Maxima

Maxima — свободная полнофункциональная система компьютерной алгебры, потомок системы Macsyma, разрабатывавшейся в рамках проекта создания искусственного интеллекта в Массачусетском Технологическом Институте с 1968 по 1982 годы (этапы разработки и руководители групп разработчиков основных разделов перечислены в *СистеКАМ-эл*. *Macsyma* (от MAC Symbolic MAnipulation), будучи первой системой аналитических вычислений, произвела в свое время переворот в компьютерной алгебре и оказала влияние на многие другие системы, в числе которых Mathematica и Maple. Изначально *Macsyma* была закрытым коммерческим проектом, доступность которого OpenSource-сообществу стала возможной благодаря профессору Техасского университета В. Шелтеру (William Schelter), который добился от Энергетического Управления США (Department of Energy, DOE) получения кода *Macsyma* и его публикации под лицензией *GPL* с именем *Maxima*. Профессор В. Шелтер долгое время разрабатывал как саму систему, так и один из диалектов лиспа — *GCL* (GNU Common Lisp), на котором разрабатывалась Maxima. Кроме того, что Maxima является свободной полнофункциональной *с.к.а.*, она имеет и другие преимущества, основным из которых является сравнительно небольшой объем — размер дистрибутива составляет около 23 Мб, а в установленном виде системе со всеми расширениями потребуется менее 85 Мб. *Maxima* состоит из интерпретатора макроязыка, написанного на *Lisp*, и нескольких поколений пакетов расширений, написанных на макроязыке системы или непосредственно на *Lisp*.

Maxima является полноценной системой компьютерной алгебры, в которой можно выполнять:

- операции с многочленами, списками, векторами, матрицами и тензорами, множествами, рациональными функциями, обобщенными функциями Дирака и Хэвисайда;

- синтаксические, алгебраические и подстановки по шаблону;
- преобразования тригонометрических и выражений со степенями и логарифмами, выносить за скобки, а также раскрывать скобки, упрощение выражений;
- нахождение пределов в конечных точках (в том числе — поиск односторонних пределов), на бесконечности;
- вычисление сумм ряда;
- дифференцирование, интегрирование;
- нахождение разложений в ряд, вычетов;
- преобразование Лапласа;
- вычисление длины кривых, площади и объема двух-, трех- и многомерных фигур.

Используя ядро и дополнительные пакеты, в *Maxima* можно решать:

- уравнения, системы линейных алгебраических уравнений (алгоритмы численного решения задач линейной алгебры почти соответствуют популярной системе компьютерной математики *MATLAB*);
- аналитическими методами обыкновенные дифференциальные уравнения первого и второго порядка, в частности, линейные и нелинейные дифференциальные уравнения первого порядка, линейные дифференциальные уравнения второго порядка и системы линейных дифференциальных уравнений первого порядка;
- приближенными методами широкий класс обыкновенных дифференциальных уравнений (разложение в ряд Тейлора и три метода возмущений для решения, классические алгоритмы Рунге-Кутта а также алгоритмы решения жестких дифференциальных уравнений);
- интегральные уравнения с фиксированными и переменными пределами интегрирования;
- задачи теории вероятностей, математической статистики и статистической обработки данных.

Эксперты отмечают, что *Maxima*, в отличие от *Mathematica* и *Maple*, в основном ориентирована на прикладные математические расчеты. В связи с этим в системе отсутствуют или сокращены разделы, связанные с теоретическими методами, как, например, теория чисел, теория групп, алгебраические поля, математическая логика. В то же время, числа в математических выражениях в системе по умолчанию предполагаются действительными. Это позволяет получать аналитические решения для многих вычислений, встречающихся в прикладных задачах (например, таких как алгебраические преобразования и упрощения, интегрирование, решение дифференциальных уравнений), для которых в комплексной области решения не существуют.

Расчеты.

Maxima производит численные расчеты высокой точности, используя точные дроби, целые числа и числа с плавающей точкой произвольной точности.

Графика.

Maxima позволяет иллюстрировать функции и статистические данные в двух и трех измерениях. В системе реализованы возможности получения качественных иллюстраций, включая параметрические графики кривых и поверхностей, а также графики векторных полей и анимацию. Число настраиваемых атрибутов в системе большое. Например, типичный трехмерный график имеет около 200 атрибутов, которые можно менять по предпочтениям пользователя. Настройки и управление сгруппированы в простых интерфейсных диалогах, при работе с графическими объектами возможны: вращение, преобразование, увеличение, включение/выключение перспективы и осей. Оформление включает вывод и установки вида заголовка иллюстрации, других текстовых комментариев, задание цвета поверхности, толщины сетки и линий осей, наименований осей, числа точек шкалы осей и шрифта чисел; возможны уточнения внешнего вида поверхностных узлов, линий и точек; можно задать цвет наружного освещения, положение и цвета осветителей. В рабочем документе можно производить анимацию положения камеры, цветов, освещения, планов и других атрибутов. Графику можно экспортировать в основные векторные и растровые форматы.

Программирование.

Как и другие *с.к.а.*, *Maxima* имеет средства процедурного программирования и программирования по заданному правилу. Система имеет открытую архитектуру, большинство команд, хранящихся в командных файлах (с расширением .mac) могут быть прочитаны и изменены пользователем. Пользователь может программировать свои команды, пополняя библиотеку. Система генерирует коды языков *Fortran* и *C*, включая управляющие операторы (циклы, ветвления), определения subroutine и function, описания типов переменных, включая матрицы, сегментацию выражений и возможность задания оптимизации общих частей выражений. Переносимость. *Maxima* успешно работает на всех современных операционных системах: Windows (готовые сборки доступны на сайте проекта), *Linux* и *UNIX*, *Mac OS* и даже под управлением *Windows CE/Mobile*. Главную роль в переносимости *Maxima* играет язык Lisp, на котором она написана. Исторически *Lisp* имеет большое количество несовместимых друг с другом диалектов, но сейчас эпоха разнообразия закончилась, поскольку появился официальный стандарт ANSI Common Lisp. Maxima была модифицирована в соответствии с этим стандартом, и в результате может работать под управлением разных реализаций *Common Lisp*, как свободных, так и проприетарных.

Взаимодействие с системой, интерфейс. Сама по себе *Maxima* — консольная программа, все математические формулы она “отрисовывает” обычными текстовыми символами. В этом есть плюсы. Например, саму систему можно

использовать как ядро, надстраивая поверх нее разные графические специализированные интерфейсы. Соответствующих примеров на сегодняшний день существует несколько.

Традиционно все *с.к.а.* имеют интерфейсные пользовательские оболочки, способные представить данные в математической нотации и облегчающие взаимодействие с пользователем. Одной из таких оболочек для Maxima является *TeXmacs* — самостоятельная программа, которую классифицируют как научный *WYSIWYG-редактор*. *TeXmacs* разработан и развивается для визуального редактирования текстов со значительным объемом математической нотации, в котором пользователь видит на экране редактируемый текст практически в том же виде, в котором он будет распечатан. В частности, доступен так называемый математический режим ввода, удобный для работы с самыми разными формулами. *TeXmacs* поддерживает также импорт/экспорт содержания в *LaTeX* и *XML/HTML*. Именно возможностями по работе с формулами пользуется *Maxima*, вызванная из *TeXmacs'a*. Фактически, формулы отображаются в привычной математической нотации, но при этом их можно редактировать и копировать в другие документы.

Также есть несколько других оболочек, лучшей из которых считается *wxMaxima*, которая как и сама *Maxima*, помимо *Linux/*BSD* существует еще и в версии для *MS Windows*, причем реализована и в версии русского языка, но пока без встроенной справки на русском. Нужно отметить, что по функциональности графические оболочки свободных систем компьютерной алгебры пока уступают коммерческим аналогам.

Maxima хорошо документирована — имеет справочное руководство с описанием практически всех встроенных функций, оно интегрировано в систему в виде онлайнового справочника, оснащенного средствами поиска. Руководство уже переведено на несколько языков, и в настоящее время переводится на русский. Система имеет отладчик, не имеет утечек памяти, для проверки работы с ней поставляются большое число тестов.

Maxima — результат коллективного труда сотен людей. Несмотря на свой солидный возраст, система продолжают активно развиваться. Последний релиз *Maxima* 5.28.0 выпущен 27 августа 2012 года Для первичного знакомства с *с.к.а. Maxima* можно рекомендовать доступное в электронном виде учебное пособие *Стахин Н.А.ОсновРсСАС-2008кн*.

Axiom

Axiom — свободная система компьютерной алгебры (см. *AXIOM.tSCS-эл*). Она состоит из среды интерпретатора, компилятора и библиотеки, описывающей строгую, математически правильную иерархию типов. Разработка системы была начата в 1971 году группой исследователей *IBM* под руководством Ричарда Дженкса (Richard Dimick Jenks). Изначально система называлась Scratchpad. Первоначально проект рассматривался как исследовательская платформа для разработки новых идей в вычислительной математике. В 1990-х система была продана компании Numerical Algorithms Group (NAG), получила название *Axiom* и стала коммерческим продуктом, но не получила коммерческого успеха и была отозвана с рынка в октябре 2001 году NAG сделала *Axiom* свободным программным обеспечением и открыла исходные коды под модифицированной лицензией *BSD*. Разработка системы продолжается, выходят новые версии (см. *AXIOM.tSCS-эл*). В 2007 году у *Axiom* появились две ветки (форка) с открытым исходным кодом: *OpenAxiom* и *FriCAS*.

OpenAxiom (http://open-axiom.sourceforge.net) в апреле 2013 года выпустила версию 1.4.2. Основные изменения, реализованные в этой версии, относятся к работе компилятора. Упомянутая выше система подготовки и редактирования документов с математической нотацией GNU TeXmacs может использоваться как интерфейс *OpenAxiom*.

Другой активно развиваемой веткой *Axiom* является FriCAS (http://fricas.sourceforge.net), сейчас используется версия 1.3.8 (версия 22/06/2022). *FriCAS* выгодно отличается от других *с.к.а.* общего назначения развитой иерархией типов, соответствующей реальным математическим структурам. *Axiom* и названные ветки на данном этапе в темпе развития уступают *Maxima*. Начинающим лучше ориентироваться на *Maxima*.

MATLAB

Интерактивная система программирования *MATLAB* (сокращение от Matrix Laboratory) разработана компанией The MathWorks, Inc. Это одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций. В настоящее время система вышла далеко за пределы специализированной матричной и стала одной из наиболее мощных универсальных интегрированных с.к.м. MATLAB включает инструменты разработки сложных программ с развитым графическим интерфейсом, является эффективной средой для проведения исследований, создания моделей, решения естественнонаучных и инженерных задач *Дьяконов В.MatlaПС-2022кн*. Система де-факто стала одним из мировых стандартов в области современного математического и научно-технического программного обеспечения. В первую очередь *с.к.м.* ориентирована на численные расчеты, особо выделяется матричная алгебра. Эффективность системы обусловлена, прежде всего, ориентацией на работу с многомерными массивами, большими и разреженными матрицами с программной эмуляцией параллельных вычислений и упрощенными средствами задания циклов. Последние версии системы поддерживают 64-разрядные микропроцессоры и многоядерные микропроцессоры, например Intel Core 2 Duo и Quad. Функциональные возможности системы обеспечены богатой библиотекой команд и своим языком программирования. Из-за большого числа поставляемых с *MATLAB* пакетов

расширения она является и самой большой из *с.к.м.*, ориентированных на персональные компьютеры. Объем ее файлов превышает 3 Гб. *MATLAB* работает на большинстве современных операционных систем, включая *Linux*, *macOS*, *Solaris* (начиная с версии R2010b поддержка Solaris прекращена) и *Windows*. Есть много изданий с описанием системы, ее составляющих — из русскоязычных можно отметить *Дьяконов В.MatlaПС-2022кн*.

Историю версий MATLAB можно проследить по ресурсу *Особе иПВМ-эл*.

Отметим только несколько знаковых позиций в части Искусственного интеллекта, Машинного обучения, Интеллектуального анализа данных по версиям после 2012 года (код R2012* означает — версия 2012 года):

- MATLAB 7.14 R2012a — была последняя версия для поддержки 32-битного *Linux*;
- MATLAB 8.2 R2013b — добавлен тип данных таблицы, среда выполнения *Java* обновлена до версии 7;
- MATLAB 8.4 R2014b — добавлены улучшенная пользовательская панель инструментов, новые функции и пакеты, такие как ру (для использования *Python*), счетчик веб-страниц, гистограммы, TCP-клиент и другие;
- MATLAB 8.6 R2015b — для работы с графиками добавлен новый механизм исполнения (LXE) и новые классы, такие как графики и орграфы;
- MATLAB 9.1 R2016b — официальный движок *MATLAB* для *Java*, новые функции кодирования и декодирования для JSON, добавлен новый “строковый” тип данных; алгоритмы для обработки данных, не помещающихся в оперативной памяти, включая алгоритмы понижения размерности, описательной статистики, кластеризации методом к-средних, линейной регрессии, логистической регрессии и дискриминантного анализа; Байесова оптимизация для автоматической настройки параметров алгоритмов машинного обучения, анализ окрестности компонента (NCA) для выбора функций модели машинного обучения;
- MATLAB 9.5 R2018b — реализовано взаимодействие осей графиков, что обеспечивает эффективный анализ данных с панорамированием, изменением масштаба; добавлены функции: удаление выбросов в массиве, таблице или расписании; задание локального окружения о каждом элементе во входных данных;
- MATLAB 9.6 R2019a — добавлены Функци задания местоположения отсутствующего значения, обнаружения выбросов с помощью процентилей; реализованы улучшения для искусственного интеллекта и аналитики;
- MATLAB 9.7 R2019b — включает обновления по искусственному интеллекту (новые возможности позволяют пользователям обучать продвинутые сетевые архитектуры с использованием пользовательских циклов обучения, автоматического дифференцирования, общих весов и пользовательских функций потерь; пользователи могут создавать генеративные состязательные сети GAN, сиамские сети, вариационные автокодеры и сети внимания; Deep Learning Toolbox также теперь может экспортировать в сети формата ONNX, которые объединяют слои CNN и LSTM, и сети, которые включают 3D-слои CNN);
- MATLAB 9.11 R2021b — добавлены: набор инструментов для статистики и машинного обучения (анализ сигналов и изображений, предварительная обработка и извлечение параметров с помощью вейвлет-методов и интерактивных приложений для моделей искусственного интеллекта), кластеризация k-средних в реальной задачах;
- MATLAB 9.11 R2021b (2021 год);
- MATLAB 9.13 R2022b включает в себя обновления по искусственному интеллекту, набор инструментов идентификации системы — создавайте нелинейные модели пространства состояний на основе глубокого обучения, используя нейронные обыкновенные дифференциальные уравнения (ОДУ); методы машинного обучения и глубокого обучения также могут представлять нелинейную динамику в нелинейных моделях ARX и Хаммерштейна-Винера.

MATLAB — коммерческая система; существуют некоммерческие варианты программных продуктов ее типа, совместимые по базовым конструкциям языка, но не совместимые по библиотечным функциям. Например, Scilab, Maxima, Euler Math Toolbox и Octave.

В состав MATLAB входят интерпретатор команд, графическая оболочка, редактор-отладчик, профилировщик (profiler), компилятор, символьное ядро *с.к.а. Maple* для проведения аналитических вычислений, математические библиотеки и библиотеки Toolboxes, предназначенные для работы со специальными классами задач.

Язык MATLAB.

Система *MATLAB* — это одновременно и операционная среда, и язык программирования. Язык *MATLAB* является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования. Среда разработки позволяет создавать графические интерфейсы пользователя с различными элементами управления; имеется возможность создавать специальные наборы инструментов (toolbox), расширяющих функциональность и представляющих коллекции функций, написанных для решения определенного класса задач.

Программы, написанные на *MATLAB*, бывают двух типов: функции и скрипты. Функции имеют входные и выходные аргументы, собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных. Скрипты используют общее рабочее пространство. Скрипты и функции не компилируются в машинный код, они сохраняются в виде текстовых файлов. Существует возможность сохранять так называемые pre-parsed программы — функции и скрипты, обработанные в удобный для машинного исполнения вид. В общем случае такие модули

выполняются быстрее запрограммированных в других средах, особенно если функция содержит команды графики. Программы *MATLAB*, как консольные, так и с графическим интерфейсом пользователя, могут быть собраны с помощью компоненты *MATLAB Compiler* в независимые исполняемые приложения или динамические библиотеки. Программы-компоновщики *MATLAB Builder* расширяют возможности MATLAB Compiler и позволяют создавать самостоятельные компоненты *Java*, *.NET* или *Excel*.

Основные пакеты расширения.

Особенностью *с.к.м. MATLAB* является возможность создавать специальные наборы инструментов (toolbox). Компания MathWorks поставляет более 80 наборов, которые используются во многих областях. В последних релизах компании они классифицируются по трем семействам — *MATLAB*, SIMULINK и Polyspace (см. *MathWMtLoT-el*), а также партнерские продукты.

Maple

Mathematica и *Maple* являются лидерами *с.к.а.*, часто их сравнивают. Представляется, что это непродуктивно. Каждая из названных систем имеет свои особенности, у них есть свои достоинства и недостатки; постоянно конкурируя друг с другом, они развиваются и совершенствуются. Большинство пользователей *с.к.а.*, прежде чем выбрать для себя основную систему, испытывали несколько других. Обмен мнениями, анализ публикаций, выступлений на конференциях и семинарах позволяют утверждать, что у каждой системы есть свои приверженцы, а специалистов, использующих *с.к.а.* достаточно продолжительное время, бесполезно убеждать, что иная, нежели предпочитаемая ими, система в чем-то лучше других. В большинстве случаев основным фактором использования конкретной *с.к.а.* является привычка пользователя. При этом многие отмечают, что, освоив любую из систем, легко работать с другими.

По полноте функционала и интерфейсным решениям инструментарий реализации символьных и численных вычислений в системах *Mathematica* и *Maple* совершенен, вопрос не в отсутствии каких-то функций или инструментов, а в навыках пользователей. Дать полный обзор возможностей системы *Maple*, как и *Mathematica*, невозможно. Вряд ли, кто-то из авторов даже специализированных изданий с ориентацией книги на конкретный класс задач может изложить все инструменты названных *с.к.а.* из рассматриваемого ими спектра. Данный материал можно рассматривать лишь как введение в возможности системы с упоминанием классов задач по интересам студентов, магистрантов, аспирантов, исследователей, программистов. Повторим, что функционал систем *Mathematica* и *Maple* почти во всем касающемся математики, прикладной математики, информатики не только достаточен, но и избыточен. Так как основной перечень возможностей *с.к.а.* уже приведен выше, а в *Maple* они реализованы, здесь отметим то, что в ряде источников или опущено, или названо другими терминами.

Системам Maple во всем мире посвящены много книг, список которых можно найти на сайте компании разработчика в соответствующем разделе *MapleB-el*.

Издания на русском можно проследить по *Аладьев В.З..МодулПМvMaVV-2011кн; Дьяконов В.Maple вМР-2022кн*.

Несмотря на фундаментальность и направленность на самые серьезные математические вычисления, системы класса Maple необходимы довольно широкой категории пользователей: студентам и преподавателям вузов, инженерам, аспирантам, научным работникам и даже учащимся математических классов общеобразовательных и специальных школ. Maple — типичная интегрированная программная система. Она объединяет в себе следующие составляющие (см. *Дьяконов В.Maple вМР-2022кн*, *Аладьев В.З..СистеКАМИП-2006кн*):

- редактор для подготовки и изменения документов и программных модулей,
- ядро алгоритмов и правил преобразования математических выражений,
- численный и символьный процессоры,
- язык программирования,
- многооконный пользовательский интерфейс с возможностью работы в диалоговом режиме,
- справочную систему с пояснениями всех функций и опций,
- систему диагностики,
- библиотеки встроенных и дополнительных функций,
- пакеты функций сторонних производителей,
- поддержку нескольких других языков программирования.

Основной документ *Maple — Worksheet*, работа с которым подобна обычному редактированию в текстовом редакторе. Текст можно форматировать на уровне абзацев, оформляя их различными стилями, или символов. Содержимое документа можно структурировать по секциям, подсекциям и так далее вплоть до ячеек. Секция может состоять из различных объектов: текстовых комментариев, строк ввода, строк вывода, графиков и других секций (подсекций). Отличием является наличие активной строки ввода, воспринимающей команды пользователя. Введенные команды (операторы) передаются ядру системы и возвращаются, как правило, в виде текста или графического изображения.

Как и в большинстве *с.к.а.*, в интерфейсе *Maple* соединены функции текстового и командного процессоров. Начиная с версии 8, в систему добавлены средства Maplets (маплеты) для поддержки визуально-ориентированного диалога. Маплеты позволяют вводить диалоговые окна, индивидуальные палитры, средства интерфейса, привычного поль-

зователям Windows-приложений. В версиях, начиная с 11, реализована концепция "умных" документов, доступны инструменты, которые обеспечивают создание самодокументируемых контекстных меню. Например, контекстное меню для матриц позволяет вызвать матричный редактор (Matrix Browser), выполнить основные операции линейной алгебры, конвертировать матрицу в различные внешние форматы. Причем не нужно знать синтаксис команд, достаточно выбрать нужный пункт из меню, появляющегося при нажатии правой кнопкой на выбранном объекте. Эксперты отмечают, что интерфейс *Maple* организован в формате интуитивно понятного и дружественного диалога, что облегчает и ускоряет освоение системы.

Система *Maple* (как и *Mathematica*), интегрирует в себе три языка: входной или язык общения с системой, реализации, программирования. Входной язык является интерпретирующим языком сверхвысокого уровня, ориентированным на решение математических задач практически любой сложности в диалоговом режиме. Он служит для задания системе вопросов или, говоря иначе, задания входных данных для последующей их обработки. Язык имеет большое число заранее определенных математических и графических функций, а также обширную библиотеку дополнительных функций, подключаемую по мере необходимости. Встроенный язык *Maple* программирования считается одним из самых лучших и мощных языков программирования математических задач. Его классифицируют, как язык процедурного программирования.

Ядро системы *Maple* и все ее составляющие улучшаются от версии к версии. Многие встроенные в систему функции, как и функции ядра, могут использоваться без какого-либо объявления, другие нуждаются в объявлении. Имеется ряд подключаемых проблемно-ориентированных пакетов (packages), тематика которых охватывает множество разделов классической и современной математики. Общее число функций в системе *Maple*, с учетом встроенных в ядро и размещенных в пакетах, превышает 5500. Ядро (не в полном составе) используют *MATLAB* и *MathCad* (начиная с 14 версии, используется символьное ядро *MuPAD*).

Основные этапы разработки, дополнения в версиях *Maple* можно проследить по *MaplePH-el*, но следует констатировать, что вопросы Машинного обучения, *Искусственного интелекта*, *Интеллектуального анализа данных* для системы пока приоритетными не являются.

Mathematica

Mathematica — система компьютерной алгебры компании *Wolfram Research* является одним из наиболее мощных и широко применяемых интегрированных программных комплексов мультимедиа-технологии ([2, 6-9]). *Mathematica* признана фундаментальным достижением в области компьютерного проектирования. По объему программных модулей она является одним из самых больших программных комплексов; содержит много новых алгоритмов, при ее реализации применено много уникальных технических решений. В системе реализованы и доступны пользователям практически все возможности аналитических преобразований и численных расчетов, она поддерживает работу с базами данных, графикой и звуком. *Mathematica* дает пользователю возможности работать, анализировать, манипулировать, иллюстрировать графиками практически все функции чистой и прикладной математики. Система обеспечивает расчеты с любой заданной точностью; построение двух- и трехмерных графиков, их анимацию, рисование геометрических фигур; импорт, обработку, экспорт изображений и звука.

Система *Mathematica* прошла путь от программы, используемой преимущественно для математических и технических расчетов, до инструмента, широко применяемого в различных других областях ([6, 7]). Среди специалистов она отмечается, как платформа для разработки, полностью интегрирующая вычисления в рабочий процесс от начала до конца, плавно проводя пользователя от первоначальных идей до развернутых индивидуальных и промышленных решений.

Mathematica имеет встроенный язык программирования *Wolfram Language*, включающий средства создания программ и пользовательских интерфейсов, подключения внешних dll, параллельных вычислений. Язык программирования системы является типичным интерпретатором, он не предназначен для создания исполняемых файлов, но вобрал в себя лучшее из таких языков программирования, как *Бейсик*, *Фортран*, *Паскаль* и *С*. Язык программирования *Mathematica* поддерживает все известные парадигмы: функциональное, структурное, объектно-ориентированное, математическое, логическое, рекурсивное и так далее. В него включены и средства визуально-ориентированного программирования на основе применения шаблонов математических символов, таких как знаки интеграла, суммирования, произведения и так далее; по своим возможностям в выполнении математических и научно-технических вычислений этот язык превосходит обычные универсальные языки программирования.

Как и всякая система компьютерной алгебры, *Mathematica* представляет собой тип программного средства, предназначенного для манипулирования математическими формулами. Основная ее цель состоит в автоматизации зачастую утомительных и в целом ряде случаев трудных алгебраических преобразований. Пользователь работает в системе с блокнотами — NB документами, каждый из которых содержит как минимум одну секцию (cell). В русскоязычной литературе можно встретить термин не секция, а ячейка. Пояснением принятого здесь предпочтения является сопоставление с *MS Excel*, где устойчиво и всеми применяется термин ячейка. Имеющие опыт работы с *Excel* и *Mathematica*, понимают разницу и то, что в *MS Excel* именно ячейки, а в блокнотах *Mathematica* более общие объекты.

NB документы можно открывать, просматривать, редактировать, сохранять, выполнять полностью или отдельные секции. Интерфейс блокнота содержит много палитр (меню) и графических инструментов для создания, редактирования, просмотра документов, отправки и получения данных к ядру и от ядра. Блокнот включает одну или набор секций, которые при необходимости можно объединять в группы. Каждая секция содержит, по крайней мере, одну строку текста или формул, цифровой объект аудио либо видео. Блокноты можно редактировать как текст в любом редакторе или в интерфейсной оболочке *Mathematica*. Ядро выполняет вычисления, может быть запущено на том же самом компьютере, на котором выполняется интерфейс, или на другом, подключенном посредством сети. Как правило, ядро запускается в момент, начала выполнения вычислений. Секции в *Mathematica* можно условно разделить на секции ввода и результата (вывода). В секциях ввода пользователь вводит или размещает команды, комментарии, объекты мультимедиа, они могут быть исполняемыми и другими; исполняемые секции обрабатываются — система возвращает результаты.

Все версии *Mathematica* содержат мощную справочную базу данных, встроенный в систему Центр Документации (Help, Documentation Center) сам по себе является примером NB документа. Не прерывая работу с модулями, можно уточнить назначение любой функции, опции, директивы или служебного слова системы; изучить, выполняя "живые" примеры, возможности получения и оформления результатов; вставить примеры целиком или фрагменты кода из примеров в собственный код.

Фрагменты хронологии версий Mathematica.

Первый выпуск *Mathematica* — июнь 1988 года, основной концепцией стало однажды и навсегда создать одну систему для разных вычислений в последовательном и объединенном виде. Основой этому стало создание нового символьного компьютерного языка для управления при минимальном числе исходных большого числа объектов, вовлеченных в технические вычисления. С момента появления все разработки Wolfram Research Inc. регулярно занимают первые места среди достижений информационных технологий, отмечаются средствами массовой информации.

Даты выпуска и дополнения, обновления версий Mathematica полно отражены в ряде изданий и на сайтах, например, *MatheИВ-эл; MatheQRH-el*.

Здесь отметим только знаковые моменты, которые оказали принципиальное влияние и на другие *с.к.а.*, а в расширенном понимании и на развитие информационных технологий.

- Mathematica 1.0, 23 июня 1988 — первый выпуск *Mathematica*.
- Версия 1.2, август 1989 года — интерфейс под *Macintosh*, поддержка удаленных ядер, добавлены стандартные пакеты *Statistics* и *Graphics*.
- Версия 2.0, январь 1991 года — *Notebook* интерфейс, протокол *MathLink* межпроцессорного и сетевого взаимодействия, поддержка звука, поддержка наборов букв не только латинского алфавита, добавлен *ParametricPlot3D*.
- Версия 2.1, июнь 1992 года — MathLink под *Macintosh*, поддержка *Windows 3.1*, синтез звука.
- Версия 2.2, июнь 1993 года — реализация для *Linux*, трехмерное построение контурных графиков, вариационное исчисление, музыка, онлайновые руководства для *Windows* и браузер функций для *Macintosh* и *NeXT*.
- Версия 3.0, сентябрь 1996 года — интерактивная система математического набора, интервальная арифметика.
- Версия 4.0, май 1999 года — публикация документов в ряд форматов, прямой импорт и экспорт в более чем 20 форматов графических, звуковых файлов и файлов стандартных данных.
- Версия 4.1, ноябрь 2000 года — интеграция с *Java* посредством J/Link 1.1, поддержка управления в реальном времени трехмерной графики под *Linux* и *Unix*.
- Версия 4.2, ноябрь 2002 года — XML-расширения, которые позволяют сохранять файлы и выражения Mathematica как XML.
- Версия 5.0, июнь 2003 года — включена *.NET/Link*, обеспечивающая полную интеграцию с *Microsoft's .NET Framework*.
- Версия 5.1, октябрь 2004 года — встроенная подключаемость к универсальной базе данных, интегрированная поддержка веб-сервисов, интерфейс *GUIKit* и встроенный разработчик программ.
- Версия 5.2, июнь 2005 года — поддержка многоядерности на основных платформах, *MathematicaMark 5.2* — обеспечивает работу с grid и кластерами.
- Версия 6, май 2007 года — язык для интеграции данных, включая автоматическую интеграцию сотен стандартных форматов данных, объединение активных графиков и элементов управления с поточным текстом и вводом.
- Версия 7, ноябрь 2008 года — встроенная поддержка параллельных высокопроизводительных вычислений, визуализация векторных полей, полная поддержка сплайнов, включая неоднородный рациональный B-сплайн, интегрированные геодезические и GIS данные.
- Версия 8, ноябрь 2010 года — интеграция с *Wolfram|Alpha*, вейвлет-анализ, встроенная поддержка *CUDA* и *OpenCL*, автоматическое генерирование кода C, расширенная двух- и трехмерная графика, включающая отображение текстур и аппаратное ускорение 3D рендеринга, интерактивный мастер создания CDF-документов.

- Версия 9, ноябрь 2012 года — набор функций анализа социальных сетей, включая выявление сообществ; встроенная связь с *Facebook*, *LinkedIn*, *Twitter*; расширенная поддержка случайных процессов; универсальная платформа для моделирования систем, которые случайным образом изменяются во времени, включая поддержку построения реализаций, оценивание параметров (калибровку), нахождение распределений временных срезов; значительно расширенный набор функций для задач теории вероятностей и статистики, включая критерии статистической независимости, новые тесты для проверки статистических гипотез, поддержка взвешенных данных, параметрические и вторичные распределения вероятностей; поддержка графов и сетей, новые и оптимизированные распределения вероятностей на графах, функции расчета транспортных сетей; встроенная интеграция с языком R, обеспечение использования кода на языке R в процессе работы в системе *Mathematica*, обмена данными между системой *Mathematica* и средой R путем выполнения R кода непосредственно из блокнота *Mathematica*; поддержка объемных операций с 3D изображениями; поддержка полного спектра интернет доступа — доступ к интернету со стороны клиента для обмена информацией с удаленными серверами, и для работы с программными интерфейсами веб-приложений, асинхронное соединение для программирования в стиле AJAX.
- Версии 10.0 (2014/12/10), 10.1, 10.2, 10.3, 10.4 (2016/03/02), в указанных и следующих версиях можно пользоваться, как настольным, так и облачным вариантом *Mathematica Online*. Избранные составляющие: существенные обновления в разделах Математические структуры, Решение дифференциальных уравнений, Структурные и семантические данные, Расширения базового языка программирования, Вычисления, связанные со временем, Анализ случайных процессов, Визуализации и графика, Обработка изображений, Инженерные вычисления, Работа с внешними объектами; новое: Геометрические вычисления (Символьная геометрия, Именные и формульные геометрические области, Сеточные геометрические области), Машинное обучение (функции Classify и Predict, Машинное обучение с высокой степенью автоматизации, Встроенный комплект классификаторов, Автоматический анализ временных рядов), Географические вычисления (Географические визуализации, Свойства, связанные с геоположением, Георасчеты с использованием логических объектов).
- Версии 11.0 (2016/10/01), 11.1 (2017/03/18), 11.2 (2017/09/14), 11.3 (2018/03/09). Избранные составляющие, существенные обновления в разделах: Работа в интернете, Облачные данные, Географические вычисления и визуализации, Подключение к внешним сервисам (*Facebook*, *Twitter*, *Instagram*, *ArXiv*, *Reddit* и многим другим). 3D печать. В части машинного обучения — новые функции позволяют пользователям извлекать признаки моделируемых объектов, уменьшать их размер, группировать данные, оптимизировать гиперпараметры и строить интерпретируемые модели; функциональные возможности извлечения признаков могут быть использованы для визуализации данных или для создания семантических расстояний для поисковых систем; доступны: вычисление нейронных сетей, аудио интеграция и вычислительная обработка лингвистических данных. Значительно усилено машинное зрение (*ImageIdentify* может определить более 10000 объектов), реализованы возможности встроенного распознавания объектов на изображениях, обучение собственного идентификатора изображений. Сформированы базы знаний Образование, сведения о Вселенной, ряд других.
- Версии 12.0 (2019/04/16), 12.1 (2020/03/18), 12.2 (2020/12/16), 12.3 (2021/05/20). Существенно модифицированы функции, расширены возможности работы в разделах: Символьные и числовые вычисления, Визуализация и графика, Геометрия и география, Наука о данных и вычисления, Изображение и аудио (новые функции и возможности: Вычисление изображений, Аудио вычисления, Вычисление изображений для микроскопии, Машинное обучение (новые функции и возможности: Суперфункции машинного обучения, Фреймворк нейронной сети, Машинное обучение для изображений, Машинное обучение для аудио, Обработка естественного языка). Отдельно следует отметить: 25 новых типов сетей, включая популярную языковую модель рендеринга BERT и генератор преформированного преобразования 2, используемый для систем генерации текста; импорт новых реализаций нейронных сетей становится немного проще, так как версия 12.1 поддерживает ONNX, открытый формат для представления моделей машинного обучения; работающие с обработкой изображений, получают дополнительную помощь с такими дополнениями, как FindImageText, который обнаруживает текст на изображении и маркирует его, аудиофилы могут воспользоваться преимуществами SpeechInterpreter и SpeechCases. Разработаны и доступны функции работы с базами знания, в частности: открыто, наполнено хранилище Wolfram Knowledgebase — в открытом доступе в облаке, включает: Язык запросов к базе знаний, Объекты астрономии и науки о космосе, Биологические и медицинские объекты, Математические объекты, Географические объекты, Объекты питания и нутрициологии, Физические и химические объекты, Финансовые и социально-экономические объекты, Культурные и исторические объекты.
- Версия 13.0 (2021/12/13), 13.1 (2022/06/29), 13.2 (2022/12/14). Существенно модифицированы функции, расширены возможности работы в разделах: Символьные и числовые вычисления (Непрерывное и дискретное исчисление, Асимптотика), Визуализация и графика (Векторная и комплексная визуализация, Многопанельная и многоосевая визуализация, Графическое освещение, наполнители и шейдеры), Графы, деревья и геометрия (Графы и сети, Деревья, Геометрическое вычисление), Оптимизация, дифференциальное уравнение в частных производных(PDE) и системное моделирование (Математическая оптимизация, PDE моделирование, Системное моделирование и системы управления), Данные и наука о данных (Машинное обучение и нейронные сети, База знаний, Дата и время, Пространственная статистика), Видео, карты и молекулы (Видео, изображения и аудио, География, Молекулы и биомолекулярные последовательности).

Приведенный выше перечень отражает много совершенно новых достижений, которые нашли применение, развитие и в других системах, информационных технологиях. Для разработок *Wolfram Research Inc.* в основном характерны преемственность интерфейса и возможность применения исходных кодов из предыдущих версий, хотя, к сожалению, это не всегда так.

Основные возможности Mathematica.

Перечисление полного перечня возможностей системы потребовало бы в несколько раз большего объема, чем разрешенный для данного издания. Например, руководство *Дьяконов В.MatheПР-2022кн* имеет объем содержательной части более 600 страниц, но фактически в нем изложены только основные функции *с.к.а.*

Первоначально вышедшая в 1988 году и в 5-ом издании актуализированная по версии Mathematica 5 книга “The Mathematica Book by Stephen Wolfram”, Fifth Edition, 2003 года издания имеет объем 1488 страниц. Перечень книг S. Wolfram можно посмотреть на сайте *Stephen-2023el*.

Помощь и справка в Mathematica.

Центр документации (*Documentation Center*), Навигатор по функциям (Function Navigator), Виртуальный учебник (*Virtual Book*) обеспечивают помощь пользователям в изучении программного языка и функциональных возможностей системы *Mathematica*. Встроенная документация системы содержит свыше 150 тысяч представительных и наглядных примеров кодов на языке *Wolfram*. Все документы полностью интерактивны, это документы *Mathematica*, в которых можно пробовать свои коды, модифицировать готовые примеры непосредственно в справке.

Использование встроенного Центра документации.

Центр документации — панель с выпадающими меню, гипертекстовый список основных разделов справки. Открыть *Documentation Center* можно через меню Help. После открытия пользователь видит документ, в котором структурированы основные разделы справки (Wolfram Mathematica Documentation Center). Гиперссылки в справке не имеют привычного вида подчеркнутых снизу надписей. Они представлены обычными надписями, но (как и обычные гиперссылки) активизируются при наведении курсора мыши на них и щелчке левой клавишей мыши. Все справки реализованы, как набор блокнотов, написанных на языке программирования системы *Mathematica*. Это означает, что все примеры в справке могут модифицироваться пользователем, и можно немедленно запускать измененные примеры, наблюдая результаты их работы. В принципе, копирование примеров в собственные блокноты вполне возможно, но по сути для изучения примеров оно не требуется.

Использование встроенного Навигатора по функциям.

Возможность иерархического просмотра справочных материалов по категориям реализована в Навигаторе по функциям (Function Navigator) — панели с открывающимися окнами, ссылками на справочные страницы. Навигатор по функциям открывается в отдельном окне, обеспечивая просмотр функций, перечисленных в списках на справочных страницах.

Использование встроенного Виртуального учебника.

Виртуальный учебник (*Virtual Book Overview*) является упорядоченной коллекцией учебных материалов по *Mathematica*, объединенных по функциональным группам. Это энциклопедический источник информации для пользователей всех уровней подготовки, желающих получить практические навыки и более детальную информацию, знания по аспектам взаимодействия с системой и выполнения функций *Mathematica*. Учебные материалы содержат подробные пояснения, примеры и ссылки на документацию по наиболее часто используемым функциям. В нижней части страницы учебных материалов расположены ссылки на документацию, родственную или имеющую отношение к данной функции. Кроме того, веб-ссылки на часто используемые функции и связанные с ними учебные материалы можно найти в правом верхнем углу окна каждого учебника.

Изложенные выше тезисы представляются важными с позиций понимания разработчиками систем *Искусственного интеллекта* окружения в близких областях, в частности, потому что *системы компьютерной алгебры*, реализующие с помощью компьютера интеллектуальные вычисления, также являются одним из (и достаточно успешно развиваемым) направлений внедрения в жизнь *Искусственного интеллекта*.

Семантическое представление чистой математики в Wolfram Mathematica

Текущее состояние.

Основываясь на более чем тридцатилетних исследованиях, разработках и использовании во всем мире, *Mathematica* и язык *Wolfram* нацелены на долгосрочную перспективу и особенно успешны в вычислительной математике. Порядка 6000 символов, встроенных в язык *Wolfram*, позволяют представлять и манипулировать огромным разнообразием вычислительных объектов — от специальных функций до графики и геометрических областей. Кроме того, база знаний Wolfram и связанная с ней структура сущностей (см. *Wolfram-2023эл; Таранчук В.Б.Метод иТРПС-2019ст; Таранчук В.Б.ИнтелВАВБ-2019ст*) позволяют пояснять, интерпретировать, формализовывать сотни конкретных "вещей" (фактов, ситуаций, предметов). Например: люди, города, продукты питания, конструкции, планеты и так далее представляются объектами, которыми можно манипулировать, их можно обсчитывать.

Wolfram Mathematica. Ближайшие планы.

Несмотря на быстрое и постоянно растущее число областей, известных языку Wolfram, многие области знаний еще ожидают вычислительного представления. В своем блоге *"Вычислительные знания и будущее чистой математики"* Стивен Вольфрам изложил видение представления абстрактной математики, известное по-разному как Вычислимый архив математики или проект *Mathematics Heritage Project* (MHP). Конечная цель этого проекта — перевести все примерно 100 миллионов страниц рецензируемых научных работ по математике, опубликованных за последние несколько столетий, в машиночитаемую форму.

Wolfram Mathematica. О семантическом представлении абстрактной математики.

В блоге *tSemanRoPM-2023el* ведущие специалисты *Wolfram* излагают свое видение будущего семантического представления абстрактной математики на двух примерах: абстрактных математических понятий функциональных, топологических пространств; концепций и теорем общей топологии. Представляется, что такие концепции, подходы должны использоваться при решении методологических проблем современного состояния Искусственного интеллекта.

База знаний Wolfram. Примеры.

Постоянно растущая база знаний *Wolfram* (Wolfram Data Repository — *WDR*), основанная на *Wolfram|Alpha* и языке *Wolfram*, на сегодняшний день является крупнейшим в мире хранилищем вычисляемых знаний. WDR, охватывающая тысячи областей, содержит тщательно отобранные экспертные знания, полученные непосредственно из первоисточников. Она включает в себя не только триллионы элементов данных, но и огромное количество алгоритмов, инкапсулирующих методы и модели практически из каждой области. *база знаний Wolfram* основана на трех десятилетиях накопления вычисляемых знаний Wolfram. Все данные в базе знаний *Wolfram* могут быть немедленно использованы для вычислений на языке Wolfram. Каждую миллисекунду каждого дня база знаний Wolfram обновляется последними данными.

Основные предметные поля *WDR* (см. *Wolfram-2023эл*) иллюстрирует *Рисунок. Предметные поля WDR*.

Рисунок. Предметные поля WDR

=

Отметим несколько примеров. Располагая обширной статистикой по сотням тысяч учебных заведений по всему миру, *Wolfram|Alpha* может вычислять ответы на сложные вопросы об образовании. Можно запросить, какие академические степени получают студенты престижных университетов. Также можно рассчитать среднюю зарплату учителей в конкретном школьном округе, узнать больше о баллах обучаемых, сравнить соотношение учащихся и преподавателей в разных странах и многое другое.

Рисунок. Число студентов в Республике Беларусь иллюстрирует ответ на запрос о числе студентов в Республике Беларусь.

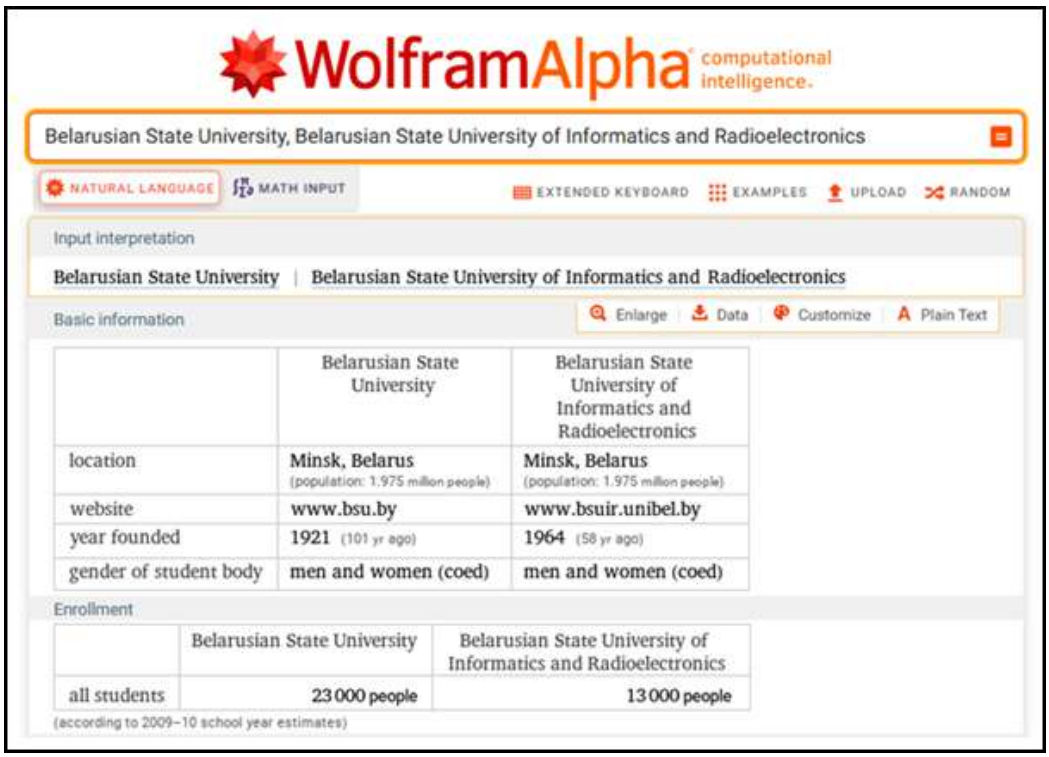
Рисунок. Число студентов в Республике Беларусь

=

Сопоставление для университетов БГУ и БГУИР иллюстрирует *Рисунок. Сравнение БГУ и БГУИР*

Рисунок. Сравнение БГУ и БГУИР

=



База знаний Wolfram. Представление знаний и доступ к ним.

Доступ к базе знаний *Wolfram* глубоко интегрирован в *Wolfram Language*. Лингвистика свободной формы позволяет легко идентифицировать многие миллионы сущностей и многие тысячи свойств и автоматически генерировать точные представления *Wolfram Language* (*WL*), подходящие для обширных дальнейших вычислений. *WL* также поддерживает пользовательские хранилища сущностей, которые позволяют выполнять те же вычисления, что и встроенная база знаний, и могут быть связаны с внешними реляционными *базами данных*.

Отметим основные группы функций *Wolfram Mathematica* для работы с *WDR*: *Entity, EntityClass, EntityValue, Transformations, Computations on Entity Classes, Standard Properties, Specific Domains, Setting Up Custom Entity Stores, Wolfram Data Repository, Wolfram Data Drop, Setting Up Custom Entity Stores, External Knowledgebases, External Database Connectivity, Web Content, Textual Question Answering, System Configuration*. В каждой из перечисленных групп более трех подгрупп. Например, в группу *Textual Question Answering* включены:

- FindTextualAnswer attempt to find answers to questions from text;
- SemanticInterpretation convert free-form linguistics to Wolfram Language for; SemanticInterpretation["string"] attempts to give the best semantic interpretation of the specified free-form string as a Wolfram Language expression;
- SemanticImport import data, converting entities etc. to Wolfram Language form,
- Interpreter interpret input of various types (e.g. “City”, “Date”, etc.); Interpreter attempt to interpret strings of a wide variety of types; Interpreter[form] represents an interpreter object that can be applied to an input to try to interpret it as an object of the specified form.

Семантический анализ естественно-языковых текстов в Mathematica.

Люди взаимодействуют друг с другом с помощью речи и текста, и это называется *естественным языком*. Компьютеры понимают естественный язык людей с помощью *обработки естественного языка* (Natural Language Processing).

обработка естественного языка — это процесс манипулирования речью текста людьми с помощью *Искусственного интеллекта*, чтобы компьютеры могли их понимать. Человеческий язык имеет много значений, выходящих за рамки буквального значения слов. Есть много слов, которые имеют разные значения, или любое предложение может иметь разные оттенки, такие как эмоциональный или саркастический. Компьютерам очень трудно интерпретировать значение этих предложений.

NLP. Основные приложения, инструменты, реализованные в системе *Wolfram Mathematica* NLP.

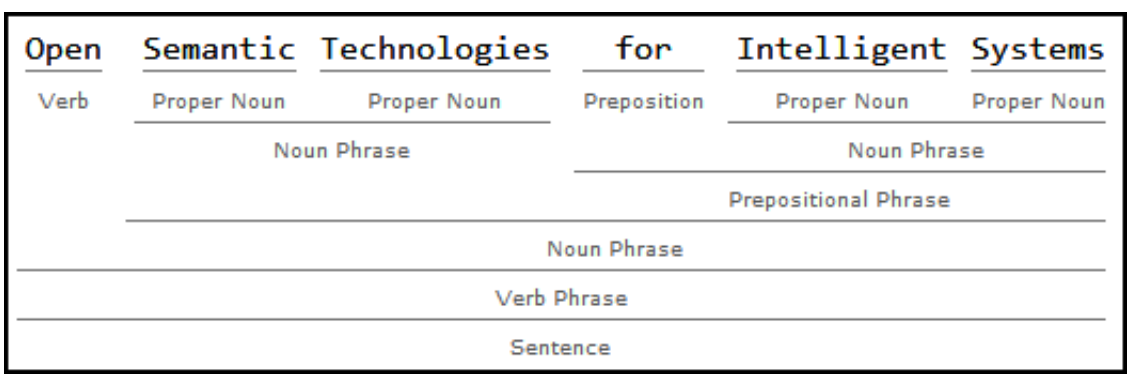
Ниже упомянуты, приведены несколько представительных примеров с пояснениями функций *WL* групп *Structural Text Manipulation*, *Text Analysis*, *Natural Language Processing*.

В некотором смысле приведенные категории условны, функций и реализуемых ими возможностей много. Например, к подгруппе *Structural Text Manipulation* можно отнести следующие: *TextCases — extract symbolically specified elements (TextCases[text, form] gives a list of all cases of text identified as being of type form that appear in text); TextSentences — extract a list of sentences (TextSentences["string"] gives a list of the runs of characters identified as sentences in string); TextWords — extract a list of words (TextWords["string"] gives a list of the runs of characters identified as words in string); SequenceAlignment — find matching sequences in text; TextStructure["text"] generates a nested collection of TextElement objects representing the grammatical structure of natural language text*.

Пример и результат выполнения функции *TextStructure* к тексту *“Open Semantic Technologies for Intelligent Systems”* с опцией *“PartsOfSpeech”* показан на *Рисунок. Составляющие фрагмента текста*

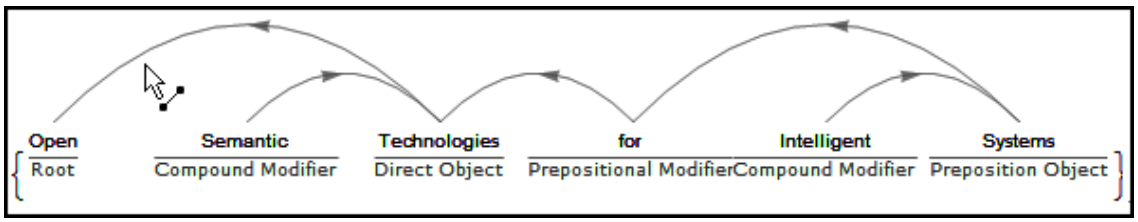
Рисунок. Составляющие фрагмента текста

=



Вариант вывода с опцией *“DependencyGraphs”* показан на *Рисунок. Составляющие фрагмента текста в формате DependencyGraphs*

Рисунок. Составляющие фрагмента текста в формате DependencyGraphs
=



NLP. Примеры использования функции FindTextualAnswer.

Ответы на вопросы на естественном языке из текста. Найти текстовый ответ *NLP*. В приведенных ниже двух примерах объектом обработки является текст *“International scientific and technical conference proceedings Open Semantic Technologies for Intelligent Systems (OSTIS). Established: 2011. Scientific areas of the conference: 05.13.11, 05.13.15, 05.13.17”*.

В варианте запроса с опцией *“Date of establishment of the conference?”* ответом является

“2011”

В варианте запроса с опциями *“Date of establishment of the conference?”*, “Scientific *areas of the conference?”* ответом является список

“2011”, “05.13.11, 05.13.15, 05.13.17”

Примеры извлечения знаний, сущности или темы из статей Википедии.

Данные *Википедии* используют *программный интерфейс MediaWiki* для извлечения содержимого статей и категорий, а также метаданных из *Википедии*. Статья может быть указана, как строка или объект языка *Wolfram*. Извлечение статей, ассоциированных с сущностями языка, обеспечивает функция *Wolfram Mathematica TextSentences*, в частности, можно работать с ресурсами *Википедии*. Ниже на *Рисунок. Сведения из Wikipedia о космонавте Leonov* представлен результат выполнения функции *TextSentences*, с параметрами *WikipediaData*, *Entity*, *“Person”*, *“AlexeiLeonov”*.

Рисунок. Сведения из Wikipedia о космонавте Leonov
=

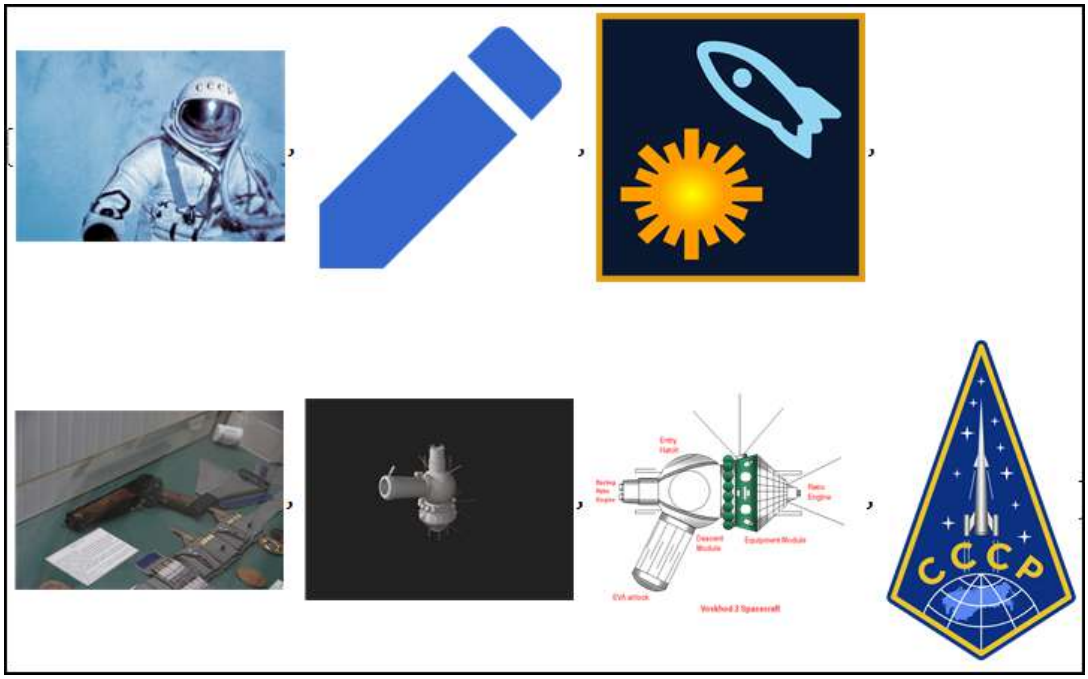
```
TextSentences[WikipediaData[Entity["Person", "AlexeiLeonov"]]][[ ;; 3]]
```

```
{Alexei Arkhipovich Leonov (30 May 1934 - 11 October 2019) was a
  Soviet and Russian cosmonaut, Air Force major general, writer, and artist.,
On 18 March 1965, he became the first person to conduct a spacewalk, exiting
  the capsule during the Voskhod 2 mission for 12 minutes and 9 seconds.,
He was also selected to be the first Soviet person to land on the
  Moon although the project was cancelled.}
```

Рисунок. Выборка в Wikipedia иллюстраций по Voskhod 2 иллюстрирует ответ системы на выполнение функции *WikipediaData* с параметрами *“Voskhod 2”*, *“ImageList”*.

Рисунок. Выборка в Wikipedia иллюстраций по Voskhod 2

=



Приведенные примеры работы с базами знаний средствами *Wolfram Mathematica*, так как функции ядра системы можно использовать в разработанных на других платформах программах, можно трактовать, как предложения инновационного совершенствования имеющихся инструментальных средств, компонентов любых интеллектуальных компьютерных систем, и конечно же *Экосистемы OSTIS*.

Пункт 7.4.2.3. Пример интеграции прототипа обучающей ostis-системы по дискретной математике и Wolfram Mathematica

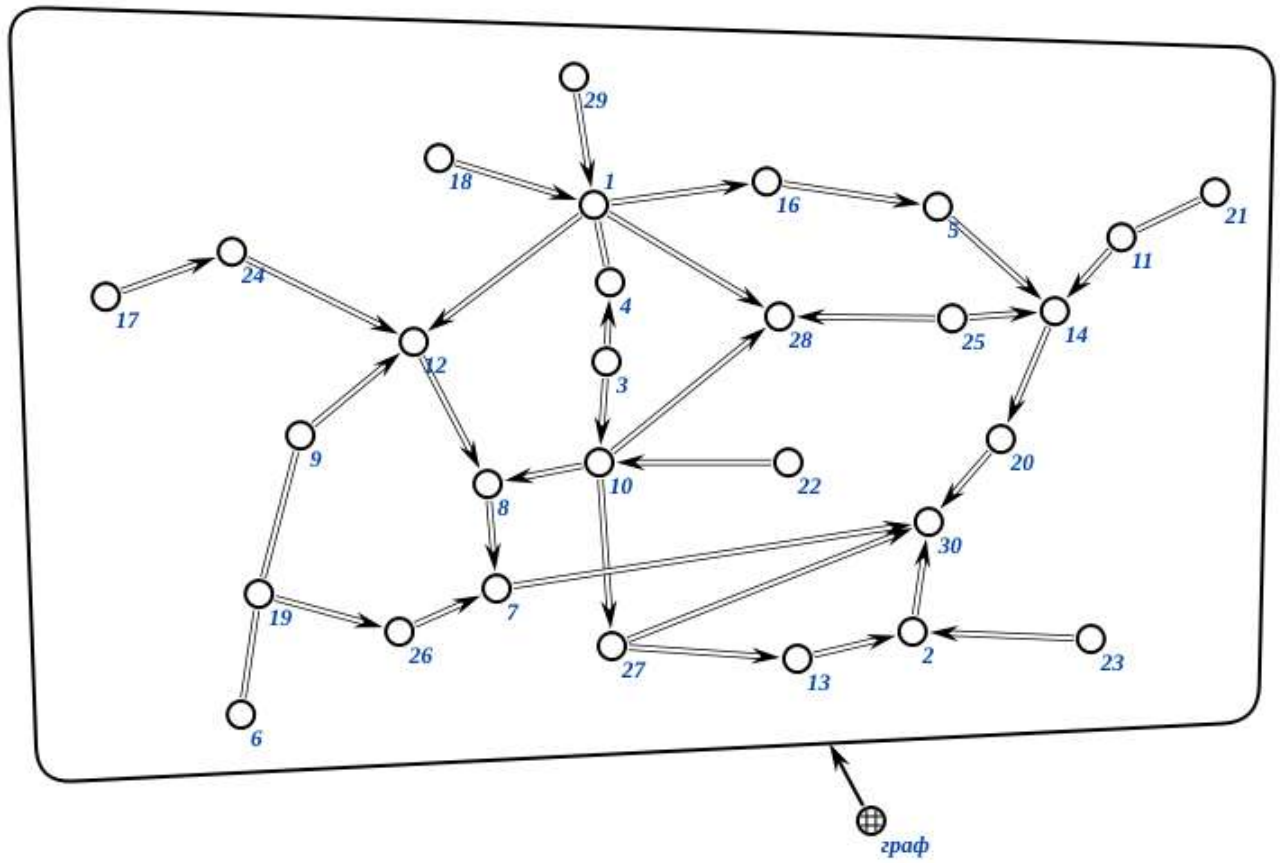
.

Приведем иллюстрации совместного использования при работе с графами прототипа обучающей *ostis-системы* по дискретной математике, входящей в состав *Экосистемы OSTIS*, и *Wolfram Mathematica*. Отмеченные ниже результаты показывают возможности использования в ostis-системе выполняемых в *Wolfram Mathematica* результатов расчетов и визуализации. Причем, реализации доступны с использованием соответствующего программного интерфейса (можно выполнить размещенный в облаке Wolfram код на Wolfram Language в рамках пользовательской программы, например, на *Python* или *C++* — https://reference.wolfram.com/language/ref/APIFunction.html]) или средств импорта, экспорта.

Отдельно отметим, какие форматы поддерживаются системой *Wolfram Mathematica*. Согласно системной функции *Mathematica* $ImportFormats, имеем следующий перечень: *3DS, ACO, Affymetrix, AgilentMicroarray, AIFF, ApacheLog, ArcGRID, AU, AVI, Base64, BDF, Binary, Bit, BMP, BSON, Byte, BYU, BZIP2, CDED, CDF, Character16, Character8, CIF, Complex128, Complex256, Complex64, CSV, CUR, DAE, DBF, DICOM, DIF, DIMACS, Directory, DOT, DXF, EDF, EML, EPS, ExpressionJSON, ExpressionML, FASTA, FASTQ, FCS, FITS, FLAC, GenBank, GeoJSON, GeoTIFF, GIF, GPX, Graph6, Graphlet, GraphML, GRIB, GTOPO30, GXL, GZIP, HarwellBoeing, HDF, HDF5, HIN, HTML, HTTPRequest, HTTPResponse, ICC, ICNS, ICO, ICS, Ini, Integer128, Integer16, Integer24, Integer32, Integer64, Integer8, JavaProperties, JavaScriptExpression, JCAMP-DX, JPEG, JPEG2000, JSON, JVX, KML, LaTeX, LEDA, List, LWO, M4A, MAT, MathML, MBOX, MCTT, MDB, MESH, MGF, MIDI, MMCIF, MO, MOL, MOL2, MP3, MPS, MTP, MTX, MX, MXNet, NASACDF, NB, NDK, NetCDF, NEXUS, NOFF, OBJ, ODS, OFF, OGG, OpenEXR, Package, Pajek, PBM, PCAP, PCX, PDB, PDF, PGM, PHPIni, PLY, PNG, PNM, PPM, PXR, PythonExpression, QuickTime, Raw, RawBitmap, RawJSON, Real128, Real32, Real64, RIB, RLE, RSS, RTF, SCT, SDF, SDTS, SDTSDEM, SFF, SHP, SMA, SME, SMILES, SND, SP3, Sparse6, STL, String, SurferGrid, SXC, Table, TAR, TerminatedString, TeX, Text, TGA, TGF, TIFF, TIGER, TLE, TSV, UBJSON, UnsignedInteger128, UnsignedInteger16, UnsignedInteger24, UnsignedInteger32, UnsignedInteger64, UnsignedInteger8, USGSDEM, UUE, VCF, VCS, VTK, WARC, WAV, Wave64, WDX, WebP, WLNet, Wolfram MathematicaLF, WXF, XBM, XHTML, XHTMLMathML, XLS, XLSX, XML, XPORT, XYZ, ZIP.*

В рассматриваемом ниже примере исходные данные (некоторый конкретный граф) поступают (выполняется импорт) из обучающей *ostis-системы по дискретной математике*, визуализируются средствами графики *Wolfram Mathematica*, затем осуществляется решение типичной задачи и предпочтительные итоговые результаты экспортируются обратно в обучающую *ostis-систему* по дискретной математике. Исходные данные к задаче, используемый далее конкретный граф показан на *SCg-текст. Используемый далее граф, вершины и дуги (отображение в ostis-системе)*:

SCg-текст. Используемый далее граф, вершины и дуги (отображение в ostis-системе)
=



Следующие иллюстрации получены в *Wolfram Mathematica*.

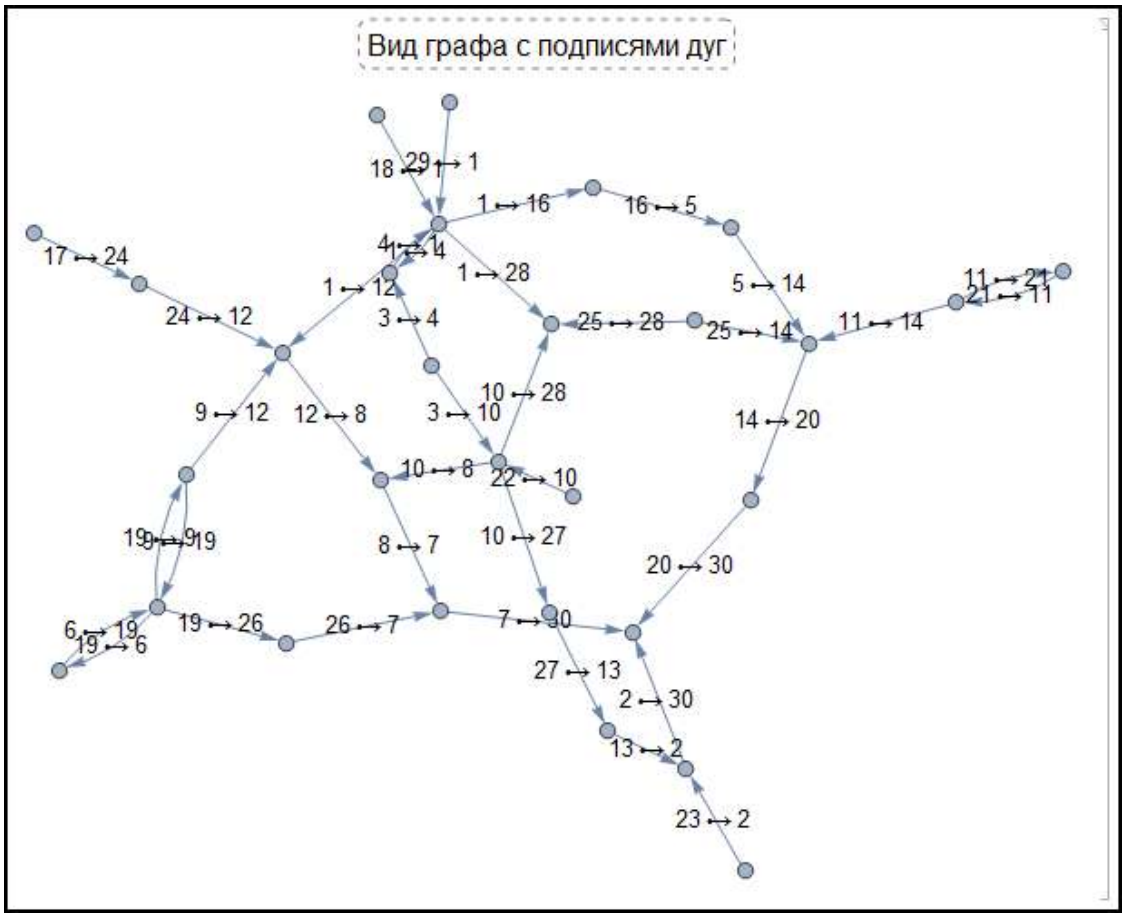
Для импортированного графа в *Wolfram Mathematica* можно получить общую информацию, например: число вершин, дуг, список ребер (напомним, что граф выше сформирован в обучающей *ostis-системе* по дискретной математике), визуализировать уже в *Wolfram Mathematica*. На *Рисунок. Используемый граф, общая информация (вывод в Wolfram Mathematica)* показаны результаты вывода в *Wolfram Mathematica* списка вершин (VertexList), числа дуг (EdgeCount), дуги (EdgeList):

Рисунок. Используемый граф, общая информация (вывод в Wolfram Mathematica)
=

```
{4, 1, 12, 16, 18, 28, 13, 2, 23, 30, 3, 10, 5, 14,
 6, 19, 8, 7, 26, 9, 22, 27, 11, 21, 24, 25, 20, 17, 29}

38

{4 ↔ 1, 1 ↔ 12, 1 ↔ 16, 18 ↔ 1, 1 ↔ 28, 13 ↔ 2, 23 ↔ 2, 2 ↔ 30, 3 ↔ 4,
 3 ↔ 10, 5 ↔ 14, 16 ↔ 5, 6 ↔ 19, 8 ↔ 7, 26 ↔ 7, 7 ↔ 30, 10 ↔ 8,
 12 ↔ 8, 9 ↔ 12, 9 ↔ 19, 22 ↔ 10, 10 ↔ 27, 10 ↔ 28, 11 ↔ 14,
 21 ↔ 11, 24 ↔ 12, 27 ↔ 13, 25 ↔ 14, 14 ↔ 20, 17 ↔ 24, 19 ↔ 26,
 20 ↔ 30, 25 ↔ 28, 11 ↔ 21, 19 ↔ 6, 19 ↔ 9, 1 ↔ 4, 29 ↔ 1}
```

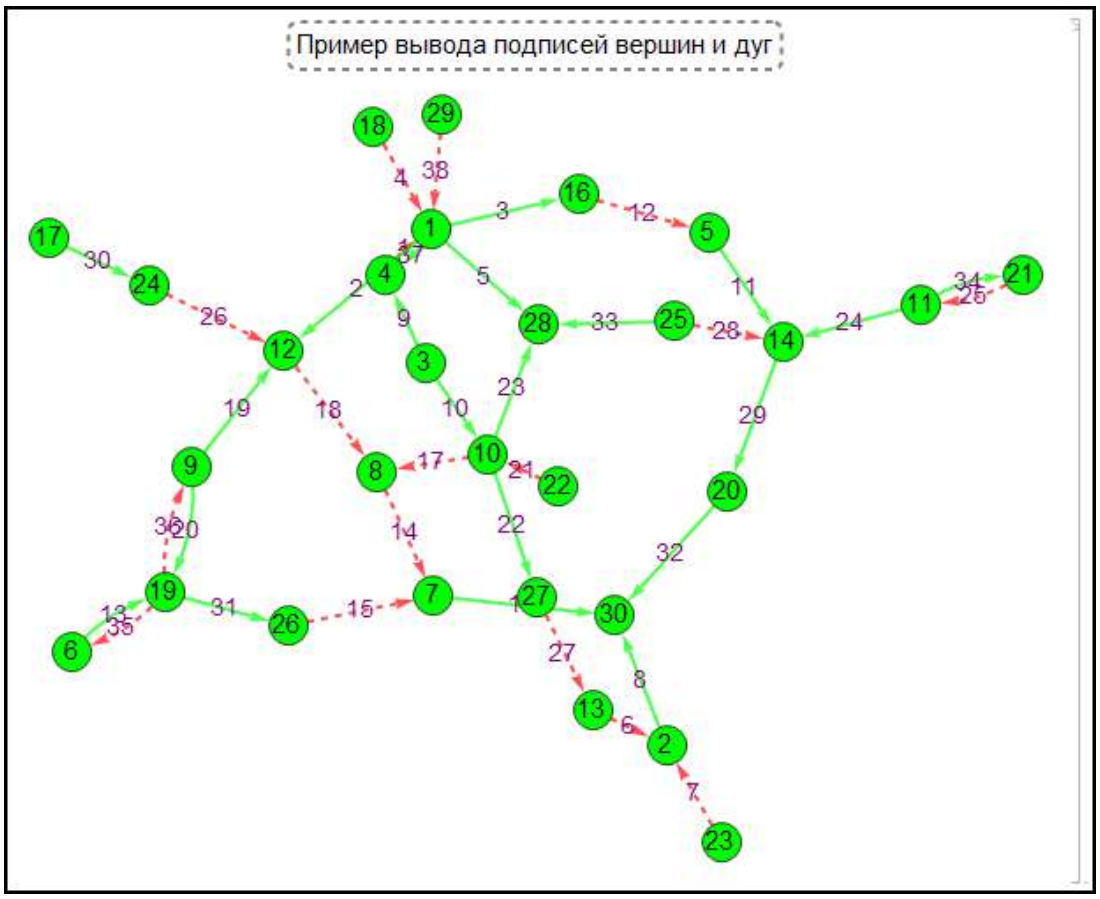
Для примера визуализации ниже показаны 3 варианта вывода. На *Рисунок. Используемый граф, дуги (вывод в Wolfram Mathematica)* приведены связи с указанием направлений:

Рисунок. Используемый граф, дуги (вывод в Wolfram Mathematica)
=



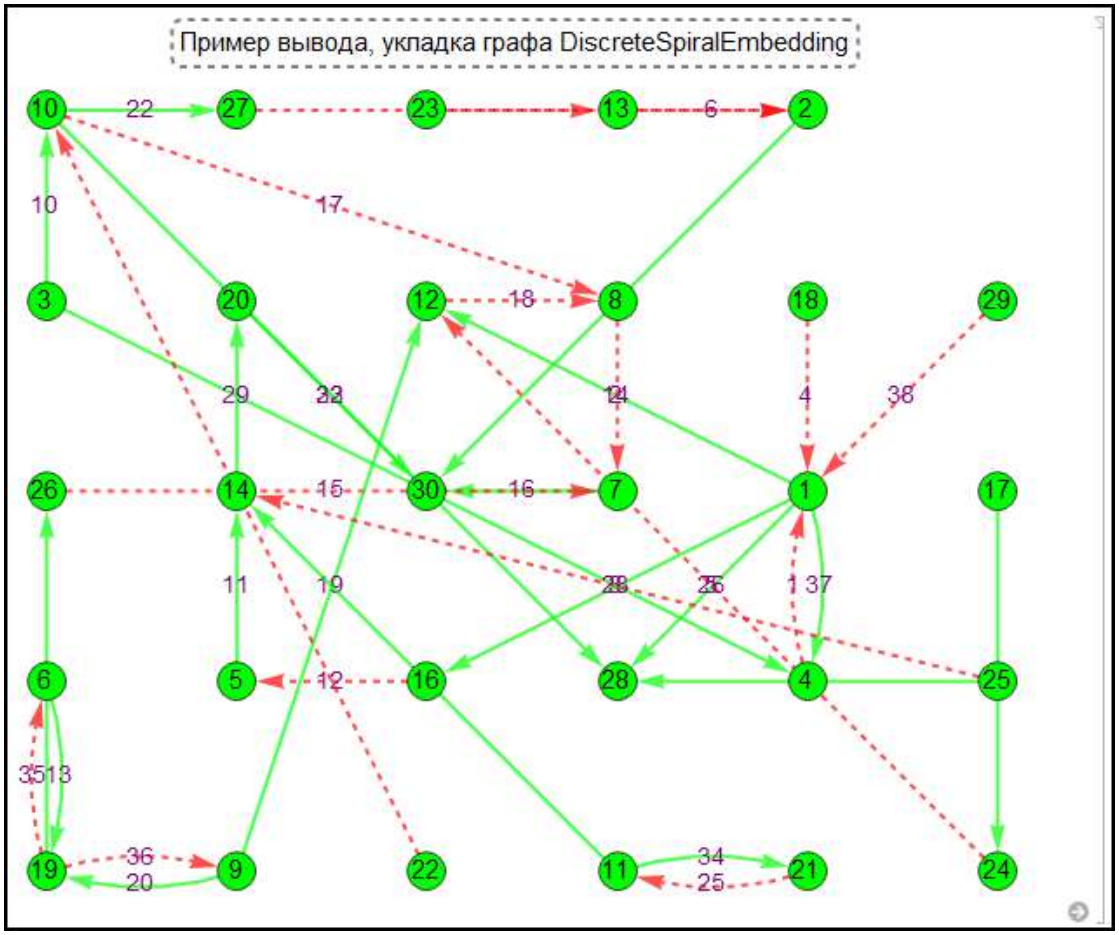
Вывод на *Рисунок. Используемый граф с оформлением по правилам (вывод в Wolfram Mathematica)* реализован с указанием стилей вершин и их номеров, задано, что все дуги от узла с большим номером к узлу с меньшим выводятся пунктирными красными линиями, а остальные сплошными зелеными:

Рисунок. Используемый граф с оформлением по правилам (вывод в Wolfram Mathematica)
=



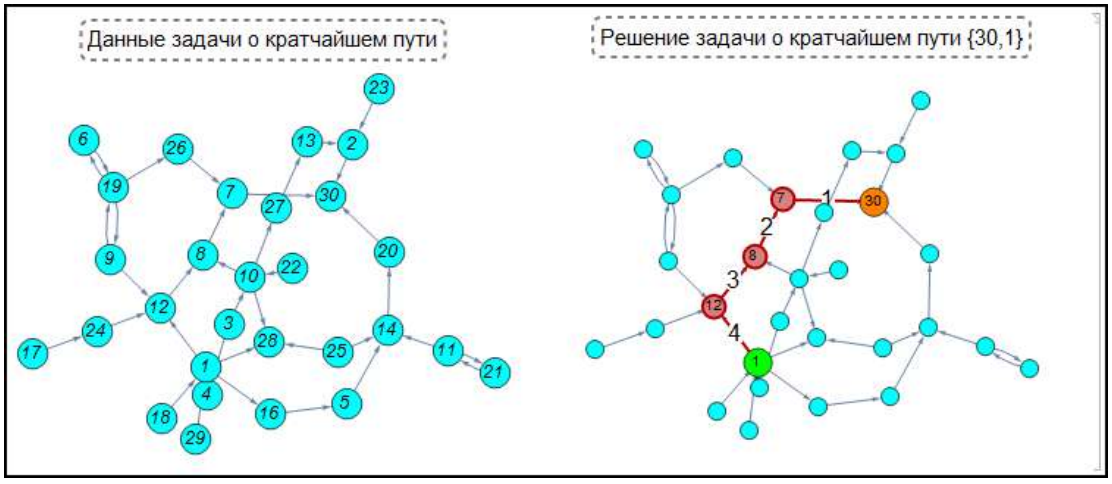
Пример вывода с заданием формата укладки графа DiscreteSpiralEmbedding, применения опций оформления вершин реализован при выводе на *Рисунок. Используемый граф с оформлением DiscreteSpiralEmbedding (вывод в Wolfram Mathematica)*

Рисунок. Используемый граф с оформлением DiscreteSpiralEmbedding (вывод в Wolfram Mathematica)
=



Пример решения задачи нахождения кратчайшего пути между двумя вершинами иллюстрирует *Рисунок. Используемый граф с оформлением DiscreteSpiralEmbedding (вывод в Wolfram Mathematica)*. (При решении использованы функции *Wolfram Mathematica*: GraphDistance, NeighborhoodGraph, Sow, DirectedEdge, Placed, Union, Flatten).

Рисунок. Решение задачи нахождения кратчайшего пути между двумя вершинами (вывод в Wolfram Mathematica)
=



Полученные и рассмотренные результаты включают трудоемкие для реализации на языках программирования задачи графики, а также математически и алгоритмически сложные задачи предметной области. Представленные варианты визуализации, нахождения решения требуют только внимательного изучения примеров системы помо-

щи *Wolfram Mathematica*, определенных навыков программирования, то есть доступны большинству инженеров-программистов. Перенести результаты в другие программные приложения также не сложно, потому что *Wolfram Mathematica* предоставляет возможности экспорта в любые типовые форматы.

Заключение к § 7.4.2.

Системы компьютерной алгебры на настоящий момент представляют собой мощные инструментальные комплексы, возможности которых давно вышли за рамки алгебраических вычислений и даже классической математики в целом. Лидеры *с.к.а.* предоставляют множество возможностей вычислений, алгоритмов обработки, анализа, визуализации. Одним из лидеров является система *Wolfram Mathematica*, ядро которой содержит более 6000 функций. Компанией *Wolfram* также разработаны много уникальных проектов, в число которых кроме системы *Wolfram Mathematica* входит вопросно-ответная система *Wolfram Alpha* (см. *Wolfram-2018el*), содержащая обширную базу знаний и набор вычислительных алгоритмов.

В основе представления фактографических, логических и процедурных знаний для систем семейства *Wolfram* лежит мультипарадигмальный язык программирования *Wolfram Language*. Наличие такого внутреннего языка описания функций систем *Wolfram* и в целом высокий уровень документированности этих функций выгодно отличает системы *Wolfram* от других сервисов, позволяющих решать общие и частные задачи. Отличие заключается в том, что во многих случаях система семейства *Wolfram* может не просто решить задачу, но и объяснить ход решения, а также помочь пользователю в выборе той или иной функции, подходящей для решения его задачи, или предложить набор функций, которые можно применить к данным, полученным в результате решения исходной задачи. Другим достоинством систем семейства *Wolfram* является им комплексность, позволяющая решать достаточно сложные задачи в рамках одного приложения и без необходимости интеграции разнородных сервисов.

Учитывая перечисленное, можно сделать вывод о целесообразности интеграции системы компьютерной алгебры *Wolfram Mathematica* с *ostis-системами*, входящими в состав *Экосистемы OSTIS*. В параграфе были рассмотрены возможные принципы такой интеграции и показаны соответствующие примеры.

Заключение к Главе 7.4.

При *интеграции* различных *сервисов* и *информационных ресурсов* с *Экосистемой OSTIS* пользователь может получить ряд значительных преимуществ, которые могут повысить эффективность его деятельности:

- повышение точности и качества данных, получение более точной и полной информации, что может повысить качество принимаемых решений (это особенно важно в условиях быстро меняющейся среды, когда точность и качество данных играют решающую роль);
- улучшение управления и контроля процессов обмена информацией, что может помочь в принятии быстрых и правильных решений;
- снижение затрат на разработку и поддержку приложений, улучшение скорости разработки новых приложений, а также повышение качества их реализации (это связано с тем, что *Экосистема OSTIS* позволяет использовать уже существующие компоненты, что сокращает время и затраты на разработку).

Глава 7.5.

Автоматизация образовательной деятельности в рамках Экосистемы OSTIS

⇒ *автор**:
- *Гулякина Н. А.*
- *Козлова Е. И.*
- *Гракова Н. В.*
- *Головатый А. И.*
- *Самодумкин С. А.*
- *Ли В.*

⇒ *библиографическая ссылка**:
- *Xu G.P..Resea oITS-2009art*
- *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Li W..OntolAfQGaKC-2020art*
- *Li W..Devel oaPSfAAV-2021art*
- *МетасOSTIS-2022эл*
- *Qian L..OntolAfCLID-2020art*
- *Papasalouros A..AutomGoMCQ-2008art*
- *Protege-2016el*
- *Li H.Resea oIAGBoD-2012art*
- *Shahmirzadi O..aTextSiVSM-2019art*
- *Ji M..aShortTSCM-2022art*
- *Anderson P..SPICE-2016art*
- *Fujiwara M..PreneNFTiS-2021art*
- *Шункевич Д.В..МетодКПСУЗ-2013ст*
- *Голенков В.В..ИнтелОСиВУ-2001кн*
- *Mousavinasab E..IntelTSaSRoC-2018art*
- *Bhatia A..AutomGoMCQ-2013art*
- *Li H.Resea oIAGBoD-2012art*
- *Wan C..aRevie oTSCM-2019art*
- *Li X.Reali oASAfSQ-2009art*
- *Wan HR..RevieoRPoTS-2019art*
- *Zeng K..aComprSoEAfKG-2021art*
- *Sun Z..aBenchSoEEA-2020art*
- *Rujiang W..Revie oCPTaM-2011art*
- *Kowalski R.PrediLaPL-1974art*
- *Krom M.tDecisPfFiPC-1970art*
- *Zhang J..AutomPoTPfTiEGItHIPT-1995art*
- *Zhang J..Self-EAPBoPGftPoWMI-2019art*

⇒ *подраздел**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*
- *§ 7.5.2. Автоматизация среднего образования с помощью ostis-систем*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем

⇒ *ключевое понятие**:
- *интеллектуальная обучающая система*
- *семантический электронный учебник*

- *компьютерные средства обучения*
- *обучение пользователя компьютерной системы*

⇒ *ключевое знание**:
- *Этапы преобразования традиционного учебника в семантический электронный учебник*
- *Предметная область методов и средств реализации целенаправленного и персонифицированного процесса обучения пользователей для каждой ostis-системы, входящей в состав Экосистемы OSTIS*
- *Преимущества интеллектуальных обучающих ostis-систем*

⇒ *библиографическая ссылка**:
- *IMS-IMS-2023el*
- *Бир С.Кибер иМ-2006кн*
- *Боргест Н.М.СтратИиЕОПР-2019ст*
- *Голенков В.В..ВиртуКиИОС-2001ст*
- *Голенков В.В..ВиртуК-2002ст*
- *Голенков В.В..СеманУиВК-2004ст*
- *Голенков В.В..ЭлектУНПО-2006ст*
- *Гулякина Н.А.КорпоСВК-2003ст*
- *Гулякина Н.А.ИнтелТвЭО-2004ст*
- *Гулякина Н.А..ЭлектУНПО-2006ст*
- *Гулякина Н.А..ФормаОМПИ-2011ст*
- *Гулякина Н.А..МетодПСМИ-2013ст*
- *Соловов А.В.ПроекКСУН-1995кн*
- *Solovov A.V..Desig aOotECC-2023art*

Организация образовательной деятельности сегодня во многом определяет уровень развития любого государства и общества. Поэтому вполне объясним большой интерес к применению телекоммуникационных и компьютерных технологий в целях повышения эффективности этой деятельности. В этой связи особую актуальность приобретает такое направление исследований из области *Искусственного интеллекта*, как *интеллектуальные обучающие системы* и автоматизация образовательной деятельности. *интеллектуальные обучающие системы* (и.о.с.) должны стать частью комплекса подготовки специалиста. В то же время, такие системы можно эффективно использовать и в процессе обеспечения повышения квалификации, при осуществлении непрерывного образования. При этом одной из особенностей разработки *интеллектуальных обучающих систем* становится то, что пользователями таких систем будут и неспециалисты в области компьютерных и телекоммуникационных технологий, люди, существенно разные и по возрасту, и по уровню знаний в той или иной *предметной области*. Это является стимулом для развития технологий *Искусственного интеллекта* в различных прикладных областях деятельности человека для реализации возможности построения и.о.с. для подготовки специалистов разных профессиональных направлений. Наиболее перспективным с точки зрения разработки и внедрения и.о.с. в учебный процесс ВУЗов видится использование *Экосистемы OSTIS* для формирования как элементов обучающих и образовательных систем, так и их интеграции в единые комплекс.

Выделяется три основных направления интеллектуализации учебного процесса, соответствующих трем уровням учебной деятельности:

- Во-первых, это — самообучение на уровне одной дисциплины. В предположении, что обучаемый положительно мотивирован, процесс обучения строится таким образом, чтобы предоставить ему максимальную свободу, помогая быстро ориентироваться в незнакомой предметной области. В связи с этим учебный материал должен быть так структурирован, чтобы его изучение было максимально удобным и, следовательно, эффективным. Здесь требуется совместная кропотливая работа эксперта в соответствующей предметной области и эксперта-педагога. В настоящее время актуальной является проблема повышения степени наглядности, когнитивности учебной информации электронного учебника с целью повышения самостоятельной познавательной деятельности обучаемого. Для решения этой задачи предлагается *семантический электронный учебник*, который представляет собой интерактивный интеллектуальный самоучитель по некоторой *предметной области*, содержащий подробные методические рекомендации по ее изучению и предназначенный для мотивированного, самостоятельного и активного пользователя, желающего овладеть знаниями по соответствующей дисциплине.
- Во-вторых, это — управление обучением на уровне отдельной дисциплины. В связи с повышением сложности и информационной насыщенности компьютерных средств обучения возникает необходимость в осуществлении управления обучением и процессом взаимодействия с пользователем. Поскольку обучающая система становится более сложной и многофункциональной и предназначена для различных категорий пользователей, то требуется адаптация к индивидуальным особенностям и обстоятельствам для каждого конкретного пользователя. Способность обучающей системы адаптироваться к пользователю является одним из показателей ее эффективности и, как следствие, интеллектуальности. *интеллектуальные обучающие системы* представляет собой сложную иерархическую систему, состоящую из совокупности взаимодействующих между собой

подсистем, каждая из которых решает определенный класс задач. В качестве базового компонента *интеллектуальных обучающих систем* используется *семантический электронный учебник*.

- В-третьих, это — управление учебной деятельностью на уровне специальности. Учебная организация и процесс обучения — это не просто совокупность автоматизированных и *интеллектуальных обучающих систем* по определенным дисциплинам, обладающих средствами мультимедиа, гибкими стратегиями обучения, подсистемами адаптации к пользователю и так далее. Для эффективного использования всех этих средств необходима инфраструктура, в которой осуществляется обработка информации, взаимодействие пользователей и подсистем, совместное решение задач, в которое вовлекаются как пользователи, так и подсистемы.

В связи с развитием средств компьютерной поддержки процесса *обучения* и создания автоматизированных *обучающих систем* и *систем дистанционного обучения*, появилась острая необходимость в интеллектуализации всего процесса обучения, так как традиционные системы уже не в силах удовлетворить всех потребностей, как учащихся, так и преподавателей. Это произошло отчасти и потому, что автоматизированные обучающие системы предполагали лишь наличие разветвленной системы ссылок, предлагая обучаемому самому, вне зависимости от его уровня знаний, выбирать путь для дальнейшего обучения. Эффективность обучения определяется не только выбором среды обучения, но и формами представления знаний. Именно форма представления знаний играет важную роль в развитии дидактического принципа "интеллектуальной наглядности" среды обучения. К числу основных задач, направленных на интеллектуализацию *компьютерных средств обучения* (*к.с.о.*) можно отнести ориентацию на семантическое представление используемых знаний (учебного материала, знаний об обучаемых, знаний о методах и технологиях обучения, знаний об учебных задачах, знаний об учебных лабораторных работах). Семантическое представление знаний позволяет, например, обеспечить достаточно эффективную компьютерную реализацию такой формы обучения, как консультация обучаемого по заданному предмету.

Современные *к.с.о.* должны обеспечивать структуризацию, систематизацию и интеграцию учебного материала, возможность навигации по семантическому пространству учебного материала и ассоциативный доступ к любому его фрагменту, а также адекватное и наглядное представление обучаемому семантической структуры этого материала. Необходимо сокращать время разработки *к.с.о.*, используя технологии параллельного проектирования, в основе которой лежит разбиение учебного материала соответствующей учебной дисциплины на достаточно самостоятельные разделы, с последующей интеграцией в единую интегрированную *интеллектуальную обучающую систему* по всей дисциплине.

Главной задачей любой обучающей системы является предоставление пользователю информации об изучаемой дисциплине в понятном и наглядном виде. Решение этой задачи возлагается на электронный учебник, который является неотъемлемой частью любой компьютерной системы обучения. Электронный учебник представляет собой *компьютерное средство обучения*, ориентированное на самостоятельную работу обучаемого и обеспечивающее:

- хранение учебного материала в электронном виде;
- отображение учебного материала обучаемому;
- возможность навигации по учебному материалу;
- редактирование учебного материала (для разработчиков электронного учебника).

Однако на современном этапе, когда объемы информации стремительно возрастают, появляется необходимость в разработке таких электронных учебников, которые бы обеспечивали:

- возможность семантической структуризации учебного материала;
- интеграцию различных форм представления учебного материала;
- ассоциативный доступ к фрагментам учебного материала;
- возможность интеграции учебной информации с учебно-методической информацией;
- возможность интеграции самих электронных учебников.

Одним из подходов к решению задачи представления и обработки знаний в *к.с.о.* является использование графодинамических моделей обработки знаний, представленных однородными семантическими сетями с базовой теоретико-множественной интерпретацией. Особенностями таких моделей являются:

- приспособленость к распределенной, асинхронной и параллельной обработке знаний;
- наглядная визуализация сложноструктурированных знаний и метазнаний;
- интеграция семантического представления знаний с любыми другими формами представления информации;
- интегрируемость различных механизмов обработки знаний.

семантический электронный учебник является тем к.с.о., которое имеет более развитые средства отображения семантической структуры предметной области с соответствующими навигационными возможностями.

Существенное отличие *семантического электронного учебника* от традиционного электронного учебника состоит в представлении учебного материала на семантическом уровне. В результате этого, с практической точки зрения, усиливаются возможности электронного учебника как обучающей среды, появляется возможность эффективного самостоятельного изучения предмета.

семантический электронный учебник — это электронный учебник, в основе которого лежит представление учебного материала в виде гипермедийной семантической сети и обеспечивается возможность семантической навигации и ассоциативного доступа к любому фрагменту учебного материала.

Учебная база знаний *семантического электронного учебника* представляет собой гипермедийную семантическую сеть, структурирующую учебный материал. В виде гипермедийной семантической сети представляется вся учебная информация, такая как, тематические планы обучения, содержательная структура учебного материала, непосредственно сам учебный материал. Основным преимуществом такой формы представления учебной информации является, во-первых, приспособленность семантических сетей для представления неформальных предложений естественного языка, во-вторых, обеспечение наглядности такого представления.

Формирование учебных материалов *семантического электронного учебника* происходит путем формализации знаний предметной области и описания их на соответствующих языках представления знаний, а также с использованием различных традиционных технологий представления информации. Технология разработки гипермедийных семантических сетей является результатом интеграции традиционных гипермедийных технологий, мультимедиа-технологий, а также технологий, связанных с представлением структуры знаний в виде семантических сетей.

Основным языком представления знаний для *семантического электронного учебника* является *SC-код* (см. *Главу 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*), который является ядром открытого семейства графовых языков представления знаний, построенных на теоретико-множественной основе. Главным достоинством *SC-кода* является то, что он представляет собой удобную основу для создания целого семейства языков, имеющих различное назначение и легко интегрируемых друг с другом.

Для навигации по семантическому пространству учебного и учебно-методического материала *семантического электронного учебника* разработаны следующие операции:
- навигационно-поисковые операции для баз знаний, представленных на *SC-коде*;
- операции поиска заданного фрагмента учебного материала;
- операции поиска понятий;
- операции поиска учебных вопросов и задач.

При интеграции семантических электронных учебников решаются следующие проблемы:
- поиск противоречий;
- выявление пар синонимичных элементов;
- локализация предположительно синонимичных пар элементов.

семантические электронные учебники представляют собой принципиально новый тип электронных учебников, в которых явно (в визуальной форме) представлена семантическая структура учебного материала. В *семантических электронных учебниках* полностью сохраняется преемственность с традиционными формами представления учебного материала. К достоинствам семантических учебников можно отнести:
- обеспечение семантической структуризации учебного материала;
- возможность семантической навигации по учебному материалу с помощью навигационно-поисковой графодинамической ассоциативной машины;
- простые механизмы интеграции семантических электронных учебников, в основе которых лежит явное описание междисциплинарных связей, позволяет разрабатывать электронные учебники по комплексу смежных учебных дисциплин.

Рассмотрим *Этапы преобразования традиционного учебника в семантический электронный учебник*:

1 этап. Описание структуры исходного учебного материала и библиографических атрибутов.

2 этап. Разбиение текста традиционного учебника на семантически элементарные фрагменты с указанием последовательности этих фрагментов в исходном тексте.

3 этап. Установление семантической типологии выделенных элементарных фрагментов текста.

4 этап. Выделение в указанных текстовых фрагментах ключевых понятий и формирование соответствующих им *sc-узлов*, а также указание связей соответствующих фрагментов с указанными *sc-узлами*.

5 этап. Перевод на *SC-код* указанных выделенных фрагментов исходного учебного материала. Установление связей семантической эквивалентности между исходными текстовыми фрагментами и их формализованной записью в *SC-коде*.

6 этап. Построение определений или пояснений указанных выше ключевых понятий (если таковые отсутствуют в учебнике), а также ключевых узлов, которые введены для описания структуры предметной области на естественном языке и на *SC-коде*, с установлением связи семантической эквивалентности текстов между ними. Кроме того, на данном этапе для выделенного набора понятий и отношений предметной области необходимо отобрать и сформулировать их основные определения, комментарии к ним, расшифровать их семантику и выделить группы семантически связанных элементов предметной области.

7 этап. Построение теоретико-множественной классификационной схемы выделенных понятий.

8 этап. Указание синонимов и омонимов выделенных понятий.

9 этап. Описание наиболее важных соотношений между указанными понятиями (кроме указанной выше теоретико-множественной классификации понятий).

Технологически семантический электронный учебник можно построить полностью сохраняя традиционный учебник, лишь надстраивая над ним соответствующую семантическую сеть. Это актуально на сегодняшний день, поскольку существует большое количество качественного учебно-методического материала, представленного в традиционной форме.

Предметная область методов и средств реализации целенаправленного и персонифицированного процесса обучения пользователей для каждой ostis-системы, входящей в состав Экосистемы OSTIS
- ∈ *предметная область*
- ∋ *максимальный класс объектов исследования′*:
  обучаемость
- ∋ *класс объектов исследования′*:
  - *обучаемость пользователя компьютерной системы*
  - *интеллектуальная компьютерная система*
  - *подсистема обучения пользователей интеллектуальной системы*
  - *интеллектуальная обучающая система*
  - *интеллектуальная обучающая ostis-система*

обучение пользователя компьютерной системы
- ⊂ *процесс*
- ⇒ *разбиение**:
  - **{•** *обучение пользователя компьютерной системы принципам работы с этой компьютерной системой*
    ≔ [обучение конечного пользователя компьютерной системы]
  - *обучение пользователя компьютерной системы принципам устройства и эволюции этой компьютерной системы*
    ≔ [обучение разработчика компьютерной системы]
  - *обучение пользователя компьютерной системы знаниям из некоторой предметной области, заложенным в эту компьютерную систему*
  - **}**

Для реализации третьего из перечисленных аспектов обучения существует отдельный класс систем, называемых ***обучающие системы***. В то же время первые два из перечисленных аспектов не менее важны, поскольку неумение конечного пользователя работать с компьютерной системой приводит к ее неэффективному использованию, а незнание разработчиком принципов работы системы приводит к дополнительным накладным расходам при ее эволюции, а иногда и к невозможности или нецелесообразности такой эволюции — проще переделать систему, чем разобраться в том, как она устроена и как ее улучшить. В особенности это актуально для *интеллектуальных систем*, принципы работы с которыми и принципы функционирования которых сложнее, чем у традиционных компьютерных систем.

интеллектуальная компьютерная система
- ≔ [сложная техническая система, разработка и даже использование которой требует высоких профессиональных качеств]
- ⇒ *Проблемы текущего состояния**:
  - **{•** [Недостаточная эффективность использования современных *интеллектуальных систем*, трудоемкость их внедрения и сопровождения, которые в значительной мере определяются высоким порогом вхождения конечных пользователей в *интеллектуальные системы*.]
  - [Пользователь часто не использует значительную часть функций даже традиционных компьютерных систем просто по той причине, что не знает об их наличии и не имеет простого механизма, позволяющего о них узнать. Для интеллектуальных систем данная проблема стоит еще более остро.]
  - [Высоки затраты на обучение разработчиков *интеллектуальных систем*, на их адаптацию под особенности устройства конкретной *интеллектуальной системы*.]
  - **}**
  - ⇒ *примечание**:
    [Перечисленные трудности связаны не только с естественной сложностью *интеллектуальных компьютерных систем* по сравнению с традиционными компьютерными системами, но с низким уровнем документации для таких систем, неудобством использования такой документации, трудоемкостью локализации средств и области решения той или иной задачи, как для конечного пользователя, так и для разработчика.]
  - ⇒ *предлагаемый подход**:

[Для решения указанных проблем, предлагается подход, предполагающий дополнение каждой *интеллектуальной системы* модулем, представляющим собой интеллектуальную обучающую подсистему, целью которой является обучение конечного пользователя и разработчика основной системы принципам работы с ней, принципам ее функционирования и развития.]

⇒ *основная идея**:

[Независимо от того, для решения каких задач разрабатывается интеллектуальная система, она должна обладать некоторыми функциями обучающей системы, даже если система изначально не является обучающей.]

⇒ *следствие**:

- пользователь должен иметь возможность обучаться как принципам работы с *интеллектуальной системой*, так и иметь возможность получать новые знания о той *предметной области*, для которой создается *интеллектуальная система*;
- разработчик *интеллектуальных систем* должен иметь возможность обучаться принципам внутреннего устройства системы, принципам ее функционирования, назначению конкретных компонентов системы, иметь возможность локализовать ту часть системы, в которой он должен разобраться для внесения изменений в функциональные возможности системы.

⇒ *примечание**:

[Для реализации данной идеи *интеллектуальная система* должна содержать не только знания о той предметной области, для которой она разработана, но и:

- знания о самой себе, своей архитектуре, компонентах, функциях, принципах работы и так далее;
- знания о пользователе, его опыте, навыках, предпочтениях, интересах;
- знания о задачах, которые решает сама система в текущий момент и задачах которые планируются к решению в будущем;
- знания об актуальных задачах по развитию системы и ее сопровождению.

]

⇒ *реализация**:

подсистема обучения пользователей интеллектуальных компьютерных систем

:= [подсистема обучения конечных пользователей и разработчиков интеллектуальных компьютерных систем]

⇒ *реализуемая функция**:

- обучение пользователя компьютерной системы принципам работы с этой компьютерной системой
- обучение пользователя компьютерной системы принципам устройства и эволюции этой компьютерной системы

подсистема обучения конечных пользователей и разработчиков интеллектуальных компьютерных систем

⇒ *технологическая основа**:

Технология OSTIS

⇒ *преимущества**:

{

- [В основе *Технологии OSTIS* лежит *SC-код*, который позволяет в унифицированном (одинаковом) виде представить любую информацию, что позволит сделать предлагаемый подход универсальным и подходящим для любого класса интеллектуальных систем.]
- [*Технология OSTIS* и, в частности, *SC-код*, легко интегрируется с любыми современными технологиями, что позволит применить предлагаемый подход для большого числа уже разработанных интеллектуальных систем.]
- [*SC-код* позволяет хранить и описывать в *базе знаний ostis-системы* любую внешнюю (инородную) по отношению к *SC-коду* информацию в виде внутренних файлов *ostis-систем*. Таким образом, *база знаний* обучающей подсистемы может содержать в явном виде фрагменты уже имеющейся документации к системе, представленной в любой форме.]
- [В рамках *Технологии OSTIS* уже разработаны модели *баз знаний ostis-систем*, *решателей задач ostis-систем* и *интерфейсов ostis-систем*, предполагающие полное их описание в базе знаний системы. Таким образом для *ostis-систем* предлагаемый подход к обучению конечных пользователей и разработчиков реализуется значительно проще и дает дополнительные преимущества.]
- [Одним из основных принципов *Технологии OSTIS* является обеспечение гибкости (модифицируемости) систем, разрабатываемых на ее основе. Таким образом, использование *Технологии OSTIS* обеспечит возможность эволюции самой интеллектуальной обучающей подсистемы.]

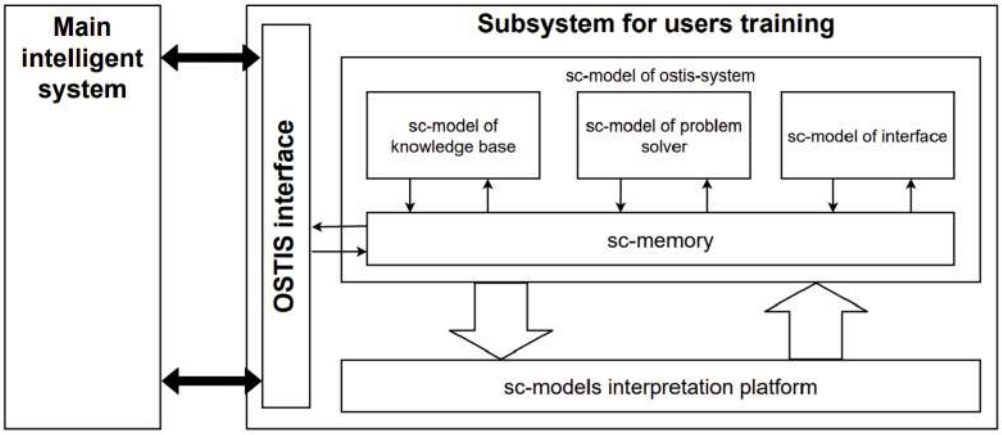
}

подсистема обучения пользователей интеллектуальных систем

⇒ *пояснение**:

[Для реализации взаимодействия *подсистемы обучения пользователей интеллектуальных систем*, реализуемой на основе *Технологии OSTIS* с основной *интеллектуальной системой*, которая в общем случае может быть реализована на основе какой-либо другой технологии, предполагается разработка интерфейсного компонента, который также является частью подсистемы. Важно отметить, что для разных *интеллектуальных систем* такие компоненты будут в значительной степени пересекаться, что, в свою очередь, позволит снизить затраты на интеграцию подсистемы обучения и основной *интеллектуальной системы*.]

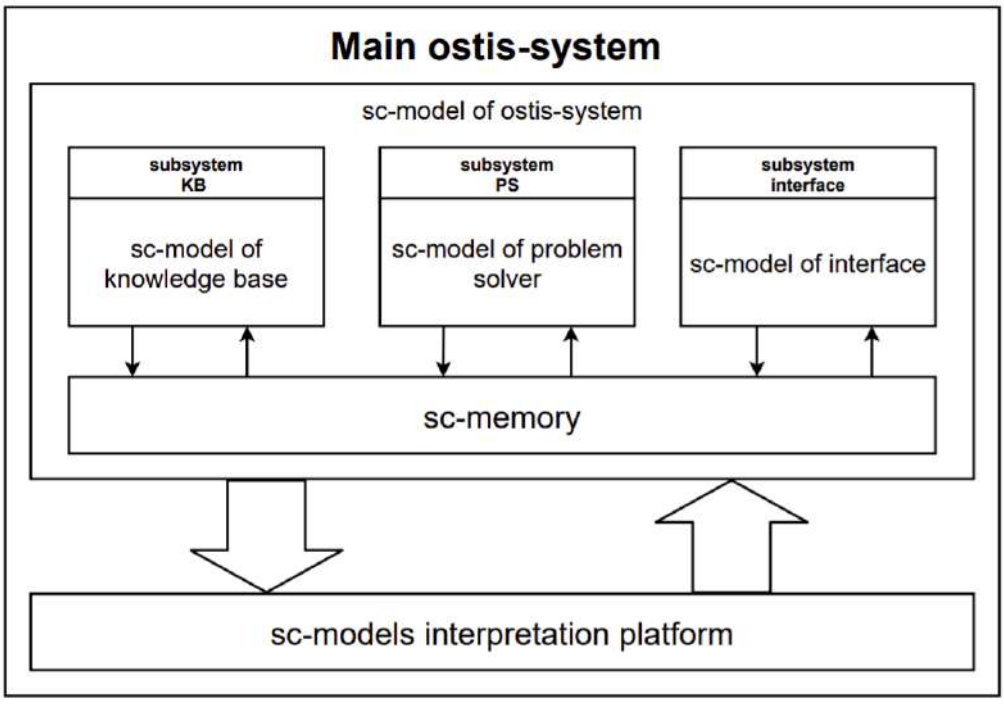
⇒ *иллюстрация**:



⇒ *примечание**:

[В случае, если рассматриваемая интеллектуальная система является *ostis-системой*, ее интеграция с подсистемой обучения пользователей интеллектуальных систем осуществляется более глубоко. Компоненты *подсистемы обучения пользователей интеллектуальных систем* просто дополняют уже существующие в основной ostis-системе компоненты, что позволяет максимально снизить затраты на интеграцию *подсистемы обучения пользователей интеллектуальных систем* и основной *ostis-системы*.]

⇒ *иллюстрация**:



интеллектуальная обучающая система
⇒ *часто используемый sc-идентификатор**:
[и.о.с.]
⊂ *интеллектуальная система*
⇒ *примечание**:
[Такого рода системы по сравнению с традиционными системами электронного обучения (например, электронными учебниками) предоставляют обладают рядом существенных преимуществ.]
⊂ *интеллектуальная справочная система*
⇒ *пояснение**:
[Каждая *интеллектуальная обучающая система* в качестве простейшего средства изучения учебного материала предполагает наличие средств навигации по этому материалу и средств задания по нему различных вопросов. Системы, обладающие только таким ограниченным набором возможностей, названы *интеллектуальными справочными системами*. Таким образом, можно сказать, что *интеллектуальная обучающая система* обязательно реализует в себе функции *интеллектуальной справочной системы*.]
⊃ *интеллектуальная обучающая ostis-система*
≔ [интеллектуальная обучающая система, построенная на основе Технологии OSTIS]
Рассмотрим *Преимущества интеллектуальных обучающих ostis-систем* в целом, а также их *баз знаний*, *решателей задач* и *пользовательских интерфейсов*.

интеллектуальная обучающая ostis-система
⇒ *обобщенная часть**:
- *база знаний интеллектуальной обучающей ostis-системы*
⇒ *преимущества**:
{
- [*SC-код* позволяет представлять знания любого рода, в том числе конкретные факты, логические утверждения (аксиомы, теоремы, определения), текстовые и мультимедийные иллюстрации и комментарии, примеры конкретных задач с решениями, в том числе доказательства и так далее.]
- [Пользователю становятся доступны достаточно полные сведения об изучаемой предметной области, отражены все ее аспекты, благодаря явному помещению в *базу знаний* всех предметных закономерностей и взаимосвязей понятий.]
- [*База знаний* системы рассматривается как иерархия предметных областей и соответствующих им онтологий, то есть позволяет произвести семантическую структуризацию предлагаемого учащемуся материала, что существенно облегчает процесс обучения за счет систематизации знаний на основе именно их семантики, а не каких-либо других сторонних факторов. Кроме этого, знания в базе могут делиться на логические разделы, каждый из которых соответствует какому-либо фрагменту излагаемого материала. *база знаний* позволяет осуществлять свободную навигацию по любым ассоциативным связям, изучая таким образом материал в той последовательности, какая кажется более логичной для самого обучаемого. С другой стороны, такой подход позволяет указать рекомендуемую последовательность изучения материала. При необходимости структура предметных областей может быть легко перестроена.]
- [Пользователю в явном виде представляется семантическая структура изучаемого учебного материала и изучаемой предметной области. При этом обеспечивается наглядная визуализация любого уровня указанной семантической структуры.]
- [Знания из различных областей представляются в схожем, подобном виде, что позволяет говорить не о семействе не связанных между собой обучающих систем по различным предметным областям, а о глобальном смысловом пространстве, объединяющем в себе знания всего семейства разрабатываемых систем. В свою очередь, наличие такого смыслового пространства обеспечивает ряд дополнительных возможностей:
  - каждая система при необходимости может использовать знания, относящиеся к другим системам, что позволяется задавать не только вопросы, касающиеся конкретной предметной области, но и вопросы, носящие междисциплинарный характер;
  - в рамках глобального смыслового пространства можно выделить часть знаний, которые имеют отношение ко многим системам из всего комплекса, например базовые знания из области математики, логики и так далее. Концепция глобального смыслового пространства позволяет записывать такие фрагменты знаний только в одной из систем, а затем использовать их во всех остальных, что существенно уменьшает количество дублирований, сокращает сроки разработки систем и снижает накладные расходы.

  ]
- [Унифицированное представление знаний позволяет не ограничивать номенклатуру пользовательских запросов только специально выделенными для этого командами, а задавать произвольный запрос системе с использованием универсального языка отображения знаний, что

делает перечень возможных запросов зависящим только от количества и разнообразия знаний, внесенных в базу знаний системы.]

}

- *решатель задач интеллектуальной обучающей ostis-системы*
  ⇒ *преимущества**:
  {• [Пользователю предоставляется возможность задавать системе любые вопросы и задачи по изучаемой предметной области. Это достигается включением в и.о.с. решателя задач, способного решать задачи по их формулировкам, в том числе, введенным пользователем. При этом указанный решатель задач может находить путь решения задачи даже, если соответствующий способ решения (например, алгоритм) ему неизвестен.]
  }
- *пользовательский интерфейс интеллектуальной обучающей ostis-системы*
  ⇒ *преимущества**:
  {• [Унификация моделей пользовательских интерфейсов позволяет отображать знания различного рода в унифицированном виде независимо от предметной области, к которой эти знания относятся. Таким образом, все разрабатываемые системы будут обладать пользовательским интерфейсом, построенным по одним и тем же принципам, что позволит существенно сократить срок ознакомления учащегося со всем семейством систем. Данный факт не отрицает возможность и необходимость разработки отдельных компонентов интерфейса, ориентированных на конкретную предметную область, например, редактора геометрических чертежей, виртуальной лаборатории для проведения химических опытов и так далее.]
    - [*и.о.с.* имеет интеллектуальный пользовательский интерфейс с компьютерными (виртуальными) моделями различных объектов изучаемой предметной области, что позволяет системе "понимать" смысл (анализировать семантику) пользовательских действий по преобразованию этих объектов. Все это существенно повышает уровень интерактивной виртуальной лабораторной среды электронного учебника.]
    - [Каждый компонент пользовательского интерфейса также является отображением определенного элемента из базы знаний, что позволяет, во-первых, легко менять интерфейс системы даже во время ее работы, а, во-вторых, позволяет пользователю задавать системе вопросы не только касательно предметной области, которой посвящена данная система, но и касательно любого из компонентов интерфейса и других частей системы. Таким образом, пользователю достаточно научиться задавать системе несколько простейших вопросов, чтобы в дальнейшем изучить все тонкости работы системой уже в процессе общения с ней.]
    - [При общении с системой пользователю предоставляется свобода в выборе любого из множества синонимичных терминов (идентификаторов), зарегистрированных в *базе знаний* системы. При этом указанные термины могут принадлежать различным естественным языкам.]
    - [Появляется принципиальная возможность реализации естественно-языкового интерфейса с пользователем (благодаря широким возможностям семантического анализа пользовательских сообщений и возможностям синтеза на семантическом уровне сообщений, адресуемых пользователям).]
    - [Достаточно легко осуществляется переориентация *и.о.с.* на обслуживание пользователей с другим естественным языком (так как основная часть базы знаний *и.о.с.*, непосредственно описывающая семантику соответствующей предметной области, абсолютно не зависит от внешнего языка, в том числе от естественного).]
  }

⇒ *преимущества**:
{• [Помимо возможности чтения текстов и иллюстративных материалов учебника предоставляется возможность навигации по семантическому пространству предметной области.]
- [Пользователю предоставляется возможность под контролем системы тренироваться (приобретать практические навыки) в решении самых различных задач по изучаемой предметной области. При этом система
  - осуществляет семантический анализ правильности решения задач как по свободно конструируемым ответам (результатам), так и по протоколам решения;
  - локализует допущенные пользователем ошибки в решении задач, определяет их причину и выдает соответствующие рекомендации пользователю.

  ]
- [Пользователю предоставляется полная свобода в выборе последовательности изучения учебного материала (маршрута навигации по учебному материалу). Тем не менее, система выдает соответствующие рекомендации.]
- [Пользователю предоставляется полная свобода в выборе решаемых им задач (в сборнике задач и лабораторных работ), но соответствующие рекомендации выдаются *интеллектуальной обучающей ostis-*

системой. Эти рекомендации направлены на то, чтобы минимизировать число решаемых задач, обеспечивающих приобретение требуемых практических навыков.]

- [Достаточно легко осуществляется интеграция нескольких самостоятельных и.о.с. по смежным дисциплинам в единый учебник, что, в частности, предоставляет возможность задавать вопросы и задачи на стыке этих дисциплин.]
- [Пользователь *и.о.с.* работает под наблюдением и контролем интеллектуального персонального ассистента, который помогает пользователю быстро и эффективно освоить возможности системы. По сути это не что иное, как руководство пользователя и.о.с., оформленное как семантический электронный учебник.]
- [При проектировании базы знаний *и.о.с.* появляется уникальная возможность проверять семантическую корректность формируемого информационного ресурса:
  - корректность определений и утверждений;
  - корректность использования различных понятий;
  - корректность алгоритмов;
  - корректность доказательств теорем;
  - и так далее.

  ]

}

⇒ *примечание**:

[Часть из перечисленных возможностей (а в предельном случае и все их них) могут быть реализованы в рамках *подсистемы обучения пользователей интеллектуальных систем*.]

§ 7.5.2. Автоматизация среднего образования с помощью ostis-систем

⇒ *ключевое знание**:

- *Принципы программ обучения и процесса обучения*
- *Требования, предъявляемые к интеллектуальным обучающим системам*

⇒ *библиографическая ссылка**:

- *Елисеева О.Е..КомпоПИОС-2013ст*

интеллектуальные компьютерные системы создаются для того, чтобы на них можно было перенаправить часть человеческой деятельности. Использование *Технологии OSTIS* при разработке не только *и.к.с.*, но и при разработке обучающихся систем различного назначения позволяет сократить сроки обучения как пользователя системы, так и обучить саму себя. Поскольку *интеллектуальная компьютерная система* может довольно быстро обучить саму себя, то она может обучить и других. Если изначально человек составляет алгоритмы, программы для обучения компьютерной системы, то затем он может получать и использовать знания, полученные интеллектуальной системой из информационной базы данных. Следовательно, *интеллектуальная компьютерная система* способна сама обучать человека, то есть стать помощником в такой важной сфере человеческой деятельности, как образование. Здесь особую роль играют семантические интеллектуальные системы, которые используют смысловые взаимосвязи, то есть делают то же, что и человек, но в миллионы раз быстрее. Семантические интеллектуальные системы позволяют, с одной стороны, ускорить процесс получения знаний на основе получения моментального доступа к огромному информационному полю, а с другой стороны, интеллектуальные системы должны помочь выбрать оптимальный путь обучения, путь по которому в наиболее короткие сроки будут получены полные знания.

Школьное образование является важным этапом, который формирует все дальнейшее развитие образования индивидуума. Если школьное образование соответствует слишком низкому уровню, то никакие последующие этапы этого не исправят. Невозможно из неграмотного, неподготовленного человека получить хорошего инженера. Если нет основы знаний, то нельзя построить и базис знаний более высокого уровня. Поэтому рассмотрение вопроса образования мы начнем со школьного образования, тем более многие проблемы и вопросы, возникающие в сфере образования более высоких ступеней совпадают с проблемами школьного образования, либо являются их следствием.

Существует много интересного в различных сферах: истории, географии и так далее. Очень интересно смотреть фильмы, читать книги и расширять свой кругозор. Можно участвовать в различных викторинах, квестах, разгадывать кроссворды. Но цель образования не просто расширить кругозор знаний ученика, а помочь ему определиться с выбором сферы деятельности согласно его способностям и потребностям общества.

При рассмотрении вопросов и способов достижения этой цели необходимо исходить из определения характеристик продукта, который надо получить в итоге. К обобществленным характеристикам качества и количества знаний выпускников различных учебных заведений можно отнести:

- компетенции — комбинация знаний, навыков и опыта, необходимых для качественного выполнения поставленных задач;
- широта знаний — набор знаний из различных областей, способных дополнять друг друга и формировать единую картину окружающего мира;
- глубина знаний — характеристика, показывающая в каком объеме и на каком уровне сложности человек обладает знаниями по тому или иному вопросу или явлению.

Но чтобы правильно выстроить процесс образования той или иной ступени необходимо конкретно очертить набор знаний (явлений, законов, формул и тому подобное), которыми должен обладать выпускник по тому или иному конкретному учебному предмету.

- ступень образования — самостоятельный завершенный этап обучения и воспитания системы образования;
- учебный предмет — система знаний, умений и навыков, отобранных их определенной отрасли человеческой деятельности.

Также необходимо чтобы учебные программы каждого года обучения по предметам дополняли и расширяли, а не дублировали, знания, полученные в предыдущие годы обучения. Программы обучения по предметам и сам процесс обучения должны соответствовать некоторым принципам (правилам, требованиям), которые будут рассмотрены далее.

Принципы программ обучения и процесса обучения следующие:

- **При составлении учебных программ** надо учитывать взаимосвязь между различными учебными дисциплинами. Знания, полученные по одним учебным дисциплинам, используются при изучении других дисциплин. Например, при изучении по физике тем раздела "простые механизмы" вычисление момента силы требует определения величин проекций, а для этого необходимы знания о соотношениях между величинами углов и сторон в прямоугольных треугольниках, знания простейших тригонометрических функций. При изучении в механике векторных характеристик движения и взаимодействия тел требуются знания понятия "вектор" и простейших операций с векторами, а также знания соотношении углов и сторон в прямоугольных прямоугольниках для нахождения проекций векторов на заданные оси. Эти знания требуются для рассмотрения всех видов сил и нахождения их равнодействующих. На основе знаний, полученных при изучении различных разделов физики, можно объяснять различные темы, изучаемые в других дисциплинах: химии, географии и других.
- В тоже время **использование знаний**, полученных в других предметах, **позволяет закрепить эти знания**. Например, использование знаний о тригонометрических функциях, проекциях, векторах при решении физических задач позволяет закрепить, углубить и обосновать эти знания, полученные из математики. Применение этих знаний в физике **позволяет сократить время**, отведенное для изучения и освоения тем в геометрии.
- Также **рассмотрение нового материала**, его освоение в рамках решения задач, **должно способствовать повторению ранее пройденного материала** по этому и другим, смежным предметам. Если на начальном этапе даются задачи непосредственно по этому материалу (на формулу). Но для действительного знания темы нужно уметь решать задачи, при решении которых надо использовать наряду с материалом изучаемой темы знания, полученные ранее при изучении других тем в рамках этого или других предметов. Это позволяет улучшить усвоение материала, обеспечить практически постоянную повторяемость материала, постепенное усложнение решаемых задач, учесть взаимосвязь различных явлений, выйти на новый уровень знания и сократить время, затрачиваемое на простое, банальное повторение материала. Также это дает возможность исключить дублирование при рассмотрении материала в старших классах.
- **Рассмотрение явлений и законов**, изучаемых в различных разделах учебных предметов (независимо от времени изучения) на каждом этапе должно быть **полным** и не может иметь незаконченный вид из-за того, что школьники не обладают какими-либо предварительными данными. Также необходимо учитывать логичность и связность получаемых знаний, чтобы знания не превращались в набор отрывочных сведений и определений, требующих банального запоминания. Логическое осмысливание материала, построение связей этого материала с имеющимися знаниями и окружающей действительностью являются главными составляющими индивидуального опыта. Знания выступают в форме понятий и отношений между ними, а также производных от них суждений и умозаключений обучающегося. Именно такие знания в виде навыков и умений лучше хранятся в памяти обучаемого.
- **Взаимодополнение** одних предметов другими возможно, если изучение различных дисциплин в школьном курсе будет тесно увязано, как по изучаемым темам, так по времени прохождения учебного материала в том или ином курсе. На данный момент то, что ученики восьмых классов не проходят до этого времени по геометрии тригонометрических функций, не дает возможности полностью рассмотреть такие темы по физике, как рычаги, силы и взаимодействие тел, преломление света и другие. Поэтому необходимо корректировать программы изучения дисциплин для своевременного использования знаний при изучении других дисциплин.
- Необходимо также рассмотреть вопрос о наполнении содержания учебного материала по каждой теме в каждой отдельной дисциплине. Необходимо учитывать, что **существуют определенные ограничения** как по сложности, так и по объему нового материала, который **может воспринять школьник** в отведенное для этого время, чтобы избежать перегрузок, которые могут негативно отразиться на его здоровье. Наиболее эффективно осваиваются темы, которые отображены в окружающем нас физическом и информационном пространстве

в силу их жизненной востребованности. Темы, которые не находят отображения в окружении и не требуются для получения необходимых жизненных навыков должны быть выведены из обязательного программного школьного материала и даваться в наиболее компактном виде на уровне ознакомления. Более расширенное рассмотрение этих тем должно быть выведено из программного школьного материала и проводиться в рамках факультативного и дополнительного образования. Таким образом, изучаемые дисциплины должны быть освобождены от излишнего сопутствующего материала, который увеличивает количество информации, нагрузку на память школьника, но не несет никакой дополнительной информации, помогающей понимать суть явлений и изучать их, а соответственно просто отнимает учебное время. Это касается каждого изучаемого предмета, каждой темы на определенных этапах их изучения. Формирование содержания учебной дисциплины в рамках программы должно исходить не из того, что один или два урока в неделю выделяется для изучения предмета, а из количества и содержания учебного материала, который **необходимо** освоить в текущем учебном году. Соответственно, на некоторые предметы может быть выделено 10 учебных часов, а на другие в 2–20 раз больше. Это особенно актуально стало в данное время в связи с тем, что стремительно увеличивается количество информации и появляются новые предметные области, освоение которых необходимо в современной жизни.

Удовлетворение программ обучения и самого процесса *обучения* этим принципам способствует тому, что различные дисциплины будут формировать всесторонние знания об окружающем мире.

При рассмотрении вопросов, связанных с образованием, нельзя обойти вниманием аспект, связанный с индивидуальностью каждого обучаемого. У каждого школьника есть способности и предрасположенности к тем или другим предметам. Учет этих факторов обеспечивает максимально возможное раскрытие творческого потенциала каждого человека. Развитию этих способностей и подготовке школьников к выбору профессиональной деятельности в дальнейшей жизни должно служить дополнительное и факультативное образование. На уровне такого образования можно осуществить более персонализированный подход к каждому ученику, при котором становится возможным полное раскрытие творческого потенциала каждого. При этом надо давать по выбранным дисциплинам более обширные и глубокие знания. Получение дополнительного образования должно учитываться при расчетах временной и физическо-умственной нагрузки учащихся.

Кроме того вопросы организации среднего образования необходимо тесно увязывать с организацией образования на последующих ступенях. Фундамент знаний, их база, сформированные в школе являются той стартовой точкой, с которой начинается следующий этап в жизни обучающегося. Рамки этих знаний четко обозначены школьной программой, следовательно, на основе этого становится понятным, какие разделы и на каком уровне должны изучаться далее для освоения той или иной специальности.

К сожалению, надо отметить, что последние годы наблюдается спад уровня школьного образования выпускников. Введение новых учебных программ не позволяет исправить данную ситуацию. Существуют различные субъективные и объективные причины, приводящие к этой ситуации. Это и резкое увеличение количества информации по различным разделам, вызванное развитием науки и увеличением доступности информации. Это инертность изменчивости школьных программ. Одной из причин является и то, что программы и учебники по разным курсам составляются разными людьми, специализирующимися в отдельных областях знаний. Эти люди не видят общей картины формирующихся знаний, пытаются наполнить свой предмет как можно более глубокими новыми данными и определениями, увеличить количество часов, отведенных для изучения предмета в школе. Иногда такое увеличение материала приводит к тому, что школьная программа по некоторым разделам предмета практически не отличается от программ высших учебных заведений. По мнению этих авторов это приводит к заинтересованности в их предмете, улучшению качества знаний по предмету. На самом деле, это приводит только к тому, что школьникам приходится запоминать намного больше информации (иногда не связанной), что приводит к перегрузке и запутыванию учащихся.

Никакая группа специалистов не сможет до конца учесть и просчитать все вопросы, связанные с образованием, так как эти проблемы являются многонаправленными и многоуровневыми. Традиционная образовательная система не может обеспечить выпускникам своевременный должный уровень знаний. Образование является той сферой деятельности, которой предстоит особенно серьезная перестройка. Для поддержания высокого уровня востребованности на рынке труда учащиеся должны высокими темпами обновлять необходимые знания, объем которых удваивается в среднем каждые полтора года, что требует постоянной переподготовки. Решить проблему можно с помощью создания интеллектуальных компьютерных систем образования, базирующихся на очень объемных базах знаний. Такие системы должны позволить исправить ошибки, допускаемые в подходах к освоению знаний, более быстро и эффективно учитывать изменения в требованиях к компетенциям выпускников, которые появляются с развитием научных и технических знаний и материально-технической базы общества. Цифровые технологии позволят создать систему образования, которая будет более эффективной и сбалансированной.

Что же должна уметь интеллектуальная система и какой базой знаний она должна обладать?

- База знаний должна быть многоуровневой.
- В интеллектуальной системе должна быть сформирована *база знаний*, охватывающая все знания, которыми при окончании школы должен обладать выпускник. В противном случае будет отсутствовать глубокая конвергенция между деятельностью по подготовке учеников и тем багажом знаний, которым они должны обладать

по окончании школы. Эта часть общей базы знаний должна охватывать все предметы школьной программы, но только в объеме, достаточном для понимания и усвоения информации по каждому предмету. Также в *базе знаний* должна быть учтена проблема объема информации с учетом сложности материала, который может быть освоен в тот или иной период обучения. На основании этих знаний интеллектуальная система должна сформировать программу обучения, распределить в какой период обучения изучаются конкретные разделы различных предметов и на каком уровне формируются знания в этот период. При этом интеллектуальная система должна учитывать, какими знаниями обладает обучаемый, когда приступает к изучению новой темы. К этому моменту школьник должен знать все необходимые предпосылки, обладать знаниями по этому и другим предметам, необходимыми для изучения темы. В результате должно быть сформировано сбалансированное распределение всего изучаемого материала из различных предметов по времени. Так как объем изучаемого материала с учетом его сложности и время обучения имеют конечное значение, то интеллектуальная система должна ограничить количество материала, предоставленного на обучение в каждый период времени, и отрезать материал, который только увеличивает количество запоминаемых данных, но не несет никакой дополнительной информации, необходимой для понимания и освоения основных законов и явлений. В результате должна быть создана модель предметной области по каждой дисциплине с учетом междисциплинарных связей. В этой связи очень важно обеспечить технологические средства "перехода" границ между учебными материалами различных учебных дисциплин. Модель предметной области играет главную роль, поскольку она используется для решения задач структуризации и систематизации учебного материала, реализации навигационно-поисковых алгоритмов по учебному материалу и реализации адаптивного управления обучением и других. В идеале, обучаемый должен иметь возможность при решении целого ряда задач работать с учебным материалом не в масштабе отдельной учебной дисциплины, а в масштабе всех дисциплин, касающихся изучаемого вопроса. Необходимо упомянуть и о логической организации учебного материала в рамках специальности, которая позволяет выявить связи учебных дисциплин, определенных тем этих учебных дисциплин, их составляющих фрагментов (теорем, определений понятий и других) с другими учебными дисциплинами, темами, фрагментами учебного материала последующими и предыдущими. Появляется возможность определить более рациональную последовательность изучения учебного материала. Система управления знаниями, находящаяся в *базе знаний*, должна также обеспечивать сбор и систематическую организацию, анализ данных и знаний из различных источников, пополнение *базы знаний* изнутри самой системы.

- Кроме того, должна быть создана интеллектуальная подсистема с открытой самодополняющейся *базой знаний*, которая будет расти по мере прохождения тем по различным предметам. Образовательный подход на основе постепенного наполнения обучающегося знаниями и умениями через постепенное их представление в рамках набора дисциплин остается актуальным. На каждом этапе обучения ученику должен быть доступен уровень знаний в базе знаний, соответствующий пройденному им материалу. обучения на основе этих оценок, то есть интеллектуальная система должна сама состоять из целого ряда подсистем, содержащих базы знаний, семантически коррелированные между собой.

Индивидуальная деятельность учащихся является важной составляющей образования, которая формирует навыки обучаемого, способствует наиболее эффективному усвоению знаний, определяет его наклонности и способствует их развитию. Индивидуальная деятельность присутствует при различных способах изучения информации: обучение по обязательной программе, дополнительное факультативное индивидуальное обучение. При дополнительных видах обучения может использоваться информация, не входящая в обязательную школьную программу. Такая более широкая информация должна быть доступна школьникам, но находиться она должна в общей базе знаний и использоваться для дополнительного, факультативного или самостоятельного образования. Освоение этой информации должно происходить вне основного школьного обучения, как по временным, так и по информационно-программным компонентам и способствовать развитию индивидуальных способностей обучающихся.

При любом виде обучения с участием *интеллектуальных обучающих систем* учащему необходим помощник — интеллектуальный персональный ассистент, который поможет наладить общение с интеллектуальной системой и сделать ее использование наиболее эффективным. Персональный ассистент каждого участника-пользователя включается в систему, чтобы уже на этапе первого взаимодействия пользователя с интеллектуальным интерфейсом обеспечить как безопасность системы, так и комфорт работы пользователя. Адаптивный подход к проектированию пользовательского интерфейса в обучающих системах является одним из перспективных направлений развития указанного класса систем. Данный подход предусматривает создание гибкой структуры диалога системы с пользователем в соответствии с рядом индивидуальных характеристик пользователя: подготовленность к работе с системой, характеристик взаимодействия с системой, интерфейсных предпочтений, индивидуальных психологических характеристик.

Поскольку развитие образовательной системы предполагается организовывать на базе общей *ostis-системы*, то естественно необходимо использовать и общие подходы при построении отдельных подсистем. Разрабатываемые общие модели интерфейса *Метасистемы OSTIS* необходимо адаптировать для образовательной ostis-системы. Это позволит значительно снизить материальные и временные затраты на построение интерфейса системы.

Наряду с задачей создания персонального *пользовательского интерфейса*, у персонального ассистента существует много других задач. Он формирует спектр информации об обучаемом. Определяет его наклонности и способности в той или иной области знаний.

Персональный ассистент при рассмотрении отдельных изучаемых тем помогает сформировать вариант навигации по семантическим связям, когда обучаемый либо система формирует некоторый путь, раскрывает признак, переходит на следующий признак и так дальше по учебному материалу, предлагает задачи на законы из изучаемого материала, на следующем этапе — задачи, в которых наряду с этим используется материал из ранее пройденных тем, в том числе и из других предметов, далее предлагаются задачи еще более сложного уровня.

Кроме того персональный ассистент должен сформировать систему оценки результатов на каждом этапе обучения. Это постоянная ненавязчивая диагностика состояния обучаемого и корректировка его модели обучения на основе анализа результатов контроля. Она формируется, с одной стороны, на основе информации о том, сколько времени потратил ученик на изучение темы, задачи какой сложности смог решить и так далее.

С другой стороны, персональный ассистент должен сформировать "разумную" вопросно-ответную систему по указанному материалу (то есть систему, способную находить ответы на достаточно большое количество вопросов, касающихся смысла, семантики соответствующего материала).

Корректировка процесса обучения должна происходить на основе этих оценок, формируя набор необходимой информации для следующего уровня, то есть интеллектуальная образовательная система должна сама состоять из целого ряда подсистем, содержащих базы знаний, семантически коррелированные между собой. Специализированные базы данных и знаний, электронные учебники, в том числе подготовленные с использованием средств гипермедиа и мультимедиа, а также сетевые источники на основе средств Интернет. Таким образом, постоянно повышаются требования к эффективности и практикоориентированности обучающих систем, что приводит к неизбежному осознанию актуальности проблемы разработки таких компьютерных обучающих систем, в которых должна обеспечиваться:

- обработка больших объемов сложноструктурированной информации различного типа;
- гибкость и легкая модифицируемость системы;
- интеграция различных моделей и механизмов решения задач;
- поддержка различных моделей обучения и управления взаимодействием с пользователем;
- интеграция различных программных систем в составе одной системы и осуществление управления их функционированием и взаимодействием;
- широкое использование средств мультимедиа;
- работа в реальном масштабе времени.

Сформулировать основные *Требования, предъявляемые к интеллектуальным обучающим системам*, следующим образом:

- **Универсальность**. Наличие средств представления и обработки учебных и учебно-методических знаний, ориентированных на любую предметную область.
- **Адаптивность**. Наличие средств формирования модели обучаемого и средств адаптации образовательного процесса к обучаемому.
- **Гибкость**. Наличие средств, позволяющих реализовывать различные модели обучения, а также поддерживающих различные формы обучения.
- **Расширяемость**. Возможность модифицировать существующие свойства системы и добавлять новые свойства без нарушения концептуальной целостности системы.
- **Распределенность**. Наличие средств организации удаленного доступа к системе. Возможность работы с системой из разных мест (локально и дистанционно) в любое время.

Современный человек в информационном обществе обязан уметь адаптироваться к быстро меняющимся информационным потокам. Формирование таких навыков — главная задача учебных организаций, к которым в современных условиях предъявляются все более высокие требования.

Дальнейшее развитие образования невозможно без совершенствования методов и средств его информатизации. Как и раньше, существуют проблемы развития мотивированного отношения к обучению, формирования навыков самообучения, несогласованности учебных материалов. Для преодоления этих проблем существует острая необходимость в применении технологий искусственного интеллекта в процессе обучения, так как традиционные компьютерные системы обучения уже не в силах удовлетворить всем требованиям, как со стороны учащихся, так и со стороны преподавателей.

Очевидной становится проблема нехватки времени на образование. Для того, чтобы получить хотя бы базовые знания, человеку приходится обучаться в системах среднего и высшего образования в течение 11–18 лет, не считая дошкольного и последипломного образования. Кроме этого, темпы развития современного общества и, в частности, различных технологий, показывают, что каждому человеку для того, чтобы сохранять профессиональную пригодность и быть полноценным членом общества необходимо постоянно развиваться и обучаться.

Предлагаются следующие этапы работ по реализации предлагаемого комплексного инновационного проекта:

Проект 1. Разработка *семантических электронных учебников* по основным дисциплинам школьного образования, имеющих средства редактирования, верификации, интеграции *баз знаний*, а также средства навигации по *базе знаний*.

Проект 2. Разработка интеллектуальных решателей задач для каждого *семантического электронного учебника*.

Проект 3. Разработка специальных средств пользовательских интерфейсов для каждого *семантического электронного учебника*.

Проект 4. Построение на базе разработанных семантических учебников *интеллектуальных обучающих систем*, осуществляющих управление обучением на основе индивидуальных особенностей обучаемого.

Проект 5. Разработка интегрированного комплекса обучающих систем, обеспечивающего комплексное обучение, соответствующее среднему образованию.

Рассмотрим реализацию указанного подхода к разработке *и.о.с.* на примере *и.о.с. по Геометрии Евклида*.

Основой любой интеллектуальной системы является *база знаний*. Это наиболее динамичный компонент, который меняется в течение всего жизненного цикла. Поэтому сопровождение интеллектуальных систем — серьезная задача.

Наиболее важный параметр *базы знаний* — качество содержащихся знаний. Лучшие *базы знаний* включают самую релевантную и актуальную информацию, имеют совершенные системы поиска информации и тщательно продуманную структуру и формат знаний. Поэтому стадия концептуального анализа или структурирования знаний традиционно является "узким местом" в жизненном цикле разработки интеллектуальных систем. Из этого следует, что разработчиками баз знаний может быть ограниченный круг специалистов, что не позволяет массово разрабатывать качественные интеллектуальные системы.

Применим методику проектирования баз знаний, основанную на *Технологии OSTIS*, при проектировании базы знаний *интеллектуальной справочной системы* по геометрии.

Согласно методике, первым этапом проектирования баз знаний является разработка первой версии тестового сборника предполагает выделение семантически полного набора вопросов, ответы на которые должны содержаться в стартовой версии базы знаний. Для системы по геометрии были выделены следующие классы вопросов: запросы определений, запросы основных свойств заданного объекта, запрос доказательств высказываний, запросы минимального высказывания (минимального фрагмента базы знаний), описывающего семантически значимую связь между всеми объектами заданного множества объектов, запросы пар высказываний, описывающих отличающиеся свойства заданных двух объектов и другие.

На следующем этапе проектирования необходимо записать ответы на выделенные вопросы, тем самым будет формироваться стартовая версия базы знаний. В процессе записи ответов на вопросы на *SCg-коде* выделяются ключевые узлы описываемой предметной области. К ключевым узлам, являющимися классами объектов исследования геометрии, относятся следующие ключевые узлы: геометрическая фигура, точка, отрезок, луч, линия, плоскость, многоугольник, треугольник, четырехугольник и так далее. К ключевым узлам, являющимися отношениями и составляющими предмет исследования, относятся: параллельность, перпендикулярность, пересечение, конгруэнтность, сторона, внутренний угол, лежать между, лежать против, вписанность и другие.

На следующем этапе проектирования каждый выделенный ключевой узел геометрии анализируется на предмет его свойств и описывается в псевдоестественной форме с использованием специализированного языка *SCn-кода*. Данная форма представления является промежуточной между представлением на естественном языке и формальном, и в дальнейшем она будет использоваться в качестве исходных данных для автоматической трансляции конструкций в формальное представление.

База знаний по геометрии на *SCn-коде* представляет собой семантически структурированный гипертекст. Все понятия из предметной области Геометрии имеют связи между собой, связи реализуются в виде ссылок, что позволяет переходить от понятия к понятию по определенной связи, указываемой при помощи ключевого слова *SCn-кода*.

В зависимости от типа описываемого понятия, статьи описания делятся на различные виды. В интеллектуальной справочной системе по геометрии были выделены следующие типы статей описания:

формальная теория (пример описываемой сущности — *Геометрия Евклида*)
- синонимичные названия теории;
- объект исследования;
- предмет исследования;
- декомпозиция на разделы;
- надтеория;
- система аксиом, лежащих в основе теории;
- неопределяемые понятия;
- противоречивые теории.

персона (пример описываемой сущности – *Евклид*)
- годы жизни;
- место рождения и жизни;
- объект, автором чего персона является;
- учителя и ученики персоны;
- биография.

понятие, не являющееся отношением (примеры понятий — *отрезок*, *треугольник*, *многоугольник*)
- синонимы понятия;
- классификация понятия по различным признакам;
- надклассы понятия и подклассы понятия;
- отношения, заданные на понятии;
- определение, пояснение понятия;
- понятия, на основе которых определяется указанное понятие;
- основные утверждения, описывающие свойства понятия;
- аналоги понятия и антиподы понятия.

отношение (примеры понятий — *внутренний угол**, *площадь**, *периметр**, *граничная точка**, *быть между'*)
- синонимичные названия отношения;
- область определения отношения;
- схема отношения;
- домены отношения;
- классы отношения по признакам классификации:
  - арность
  - ориентированность
  - наличие кратных связок и встречных связок
  - наличие мультисвязок
  - транзитивность, рефлексивность
- подклассы отношения;
- аналоги отношения;
- утверждения, описывающие свойства данного отношения или его подмножеств.

высказывание (пример описываемой сущности — *Теорема Пифагора*)
- в состав какой теории входит высказывание;
- логический статус (определение, аксиома, теорема);
- формулировка высказывания;
- логический тип (существование, несуществование, единственность, всеобщность, эвивалентность, необходимость и достаточность)
- на основании каких высказываний доказывается указанное высказывание;
- доказательство.

конкретный объект исследования (пример – *Треугольник ABC*)
- идентификаторы объекта;
- рисунок;
- описание связей с другими фигурами и точками (стороны, вершины, биссектрисы, медианы, высоты, граница, точки пересечения и другие);
- числовые характеристики (периметр, площадь, величина углов, величина сторон и другие);
- типология по различным признакам.

Таким образом, в базе знаний хранится большое многообразие видов знаний — классы геометрических фигур (планарные фигуры, линейные фигуры, фигуры, имеющие граничные точки), конкретные фигуры (треугольник, трапеция, окружность), классы отношений, конкретные отношения, утверждения различного типа (аксиомы, теоремы, утверждения определяющего типа), доказательства утверждений (доказательства теорем, лемм), алгоритмы решения задач (в том числе геометрических построений), изображения, иллюстрирующие различные геометрические чертежи, видео, флеш, иллюстрирующие решение задач и различные геометрические построения. Возможность представления знаний различного вида позволяет описывать информацию в базе знаний с различных ракурсов, что в свою очередь переводит базу знаний на другой, более высокий интеллектуальный уровень.

Технология проектирования интеллектуальных решателей задач основана на задачно-ориентированной методологии. В связи с этим проектирование системы операций (sc-агентов, см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*) состоит из четырех основных этапов:
- создание тестового сборника задач, которые решаются в рамках исследуемой предметной области;
- определение списка операций, которые будут использоваться при решении задач из тестового сборника;
- создание семантической спецификации каждой из операций;
- реализация и отладка операций.

Такая методика проектирования операций позволяет создавать предметно-независимые операции для решения конкретных прикладных задач из исследуемой предметной области. Спроектированные таким образом операции являются *многократно используемыми компонентами* (см. *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*) и могут быть использованы в других интеллектуальных справочных системах.

Опишем этапы проектирования системы операций для интеллектуального решателя задач по геометрии на примере конкретной прикладной задачи.

Пусть дан некоторый плоский треугольник с известными длинами сторон. Необходимо найти величину площади заданного треугольника. С точки зрения геометрии задача является довольно тривиальной, однако ее решение будет достаточно непростым, если предположить, что пользователь не определился, что он хочет найти в треугольнике: площадь, углы, периметр или другие характеристики. При помощи *SC-кода* можно универсально представить знания о данном треугольнике таким образом, чтобы все задачи, перечисленные выше, решались с помощью одного и того же набора операций.

Тестовый сборник задач будет состоять из единственной задачи, описанной выше.

Определим те операции, которые будут использованы при решении задачи о нахождении площади:
- операция поиска в базе знаний информации о значении искомой величины (площади) данного треугольника;
- операция поиска в базе знаний формулы, позволяющей по имеющейся информации найти значение искомой величины (площади);
- операция, осуществляющая прямой логический вывод (modus ponens);
- операция, которая осуществляет унификацию найденной (сгенерированной) формулы с учетом заданных параметров;
- операция, вычисляющая значение арифметического выражения;
- элементарные арифметические операции (сложение (вычитание), умножение (деление), возведение в степень (извлечение корня) и так далее).

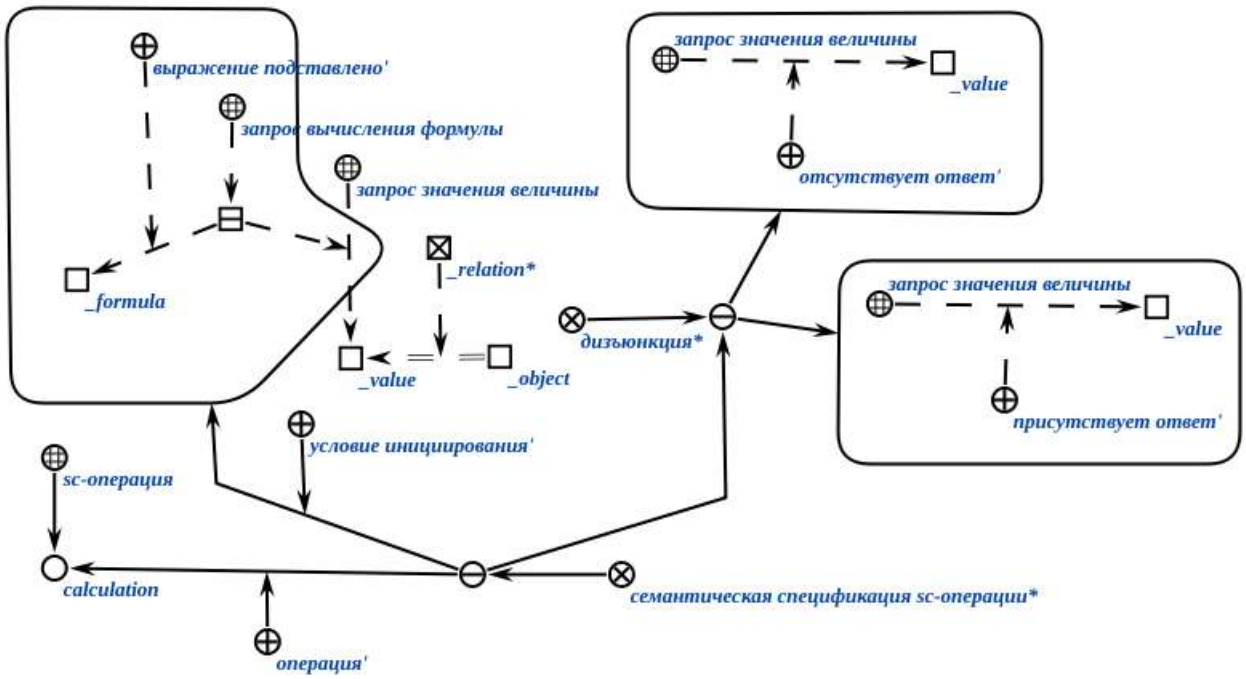
Далее опишем часть третьего этапа проектирования на примере операции вычисления значения формулы.

Пример семантической спецификации операции

Семантическая спецификация операции — это sc-конструкция, которая описывает интерфейс проектируемой операции: условие инициирования, название операции, возможные результаты работы (см. *SCg-текст. Семантическая спецификация операции вычисления формулы*).

Условием инициирования для этой операции является факт появления в памяти sc-дуги, выходящей из узла “запрос вычисления формулы”. После появления в памяти этой дуги вызывается операция “calculation”. В результате работы этой операции либо вычисляется значение некоторой искомой величины, либо генерируются знания о том, что ответ не найден. Отрицательный результат работы этой операции может быть использован другими операциями для того, чтобы попытаться каким-либо другим способом вычислить значение искомой величины.

SCg-текст. Семантическая спецификация операции вычисления формулы
=



Завершающим этапом является реализация и тестирование спроектированной операции. Этап реализации также можно разбить на два шага:

- Разработка алгоритма операции
- Реализация программы на подходящем языке. Предпочтительно использовать специально адаптированный *Язык SCP* (см. *§ 3.3.5. Базовый язык программирования ostis-систем*).

Опишем алгоритм работы операции:

1. Проверяем корректность структуры, которая представляет собой запрос.
2. Из запроса получаем информацию об искомой величине.
3. Находим узел, обозначающий искомую величину в формуле.
4. Находим в формуле все связки отношений *сложение**, *произведение** и *возведение в степень**.
5. Просматриваем все связки, найденные в шаге 4.
    1. Если связка принадлежит классу отношения *сложение**, то инициируем операцию сложения.
    2. Если связка принадлежит классу отношения *произведение**, то инициируем операцию умножения.
    3. Если связка принадлежит классу отношения *возведение в степень**, то инициируем операцию возведения в степень.
    4. Если количество выполняемых операций превысило определенный пользователем предел, то генерируем факт отсутствия ответа и завершаем работу операции.
6. Если значение искомой величины не посчитано, то переходим к шагу 5.
7. Генерируем факт присутствия ответа.

Пример протокола решения задачи

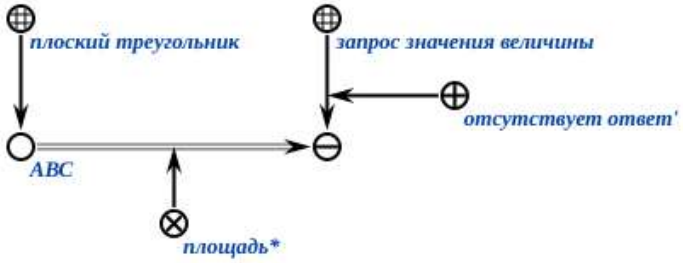
Опишем протокол решения вышеописанной задачи. Протокол отражает последовательность запуска и выполнения операций, текущие условия их запуска, а также результаты выполнения каждой из операций.

1. В памяти появляется конструкция (см. *SCg-текст. Запрос значении площади треугольника ABC*).

SCg-текст. Запрос значении площади треугольника ABC
=

Запускаются операции find_value и find_formula, при этом выполнение find_formula завершается, так как конструкция не полностью соответствует условиям запуска. find_value проверяет наличие значения указанной величины и, не найдя его, заявляет о его отсутствии (см. *SCg-текст. Результат работы операции find_value*).

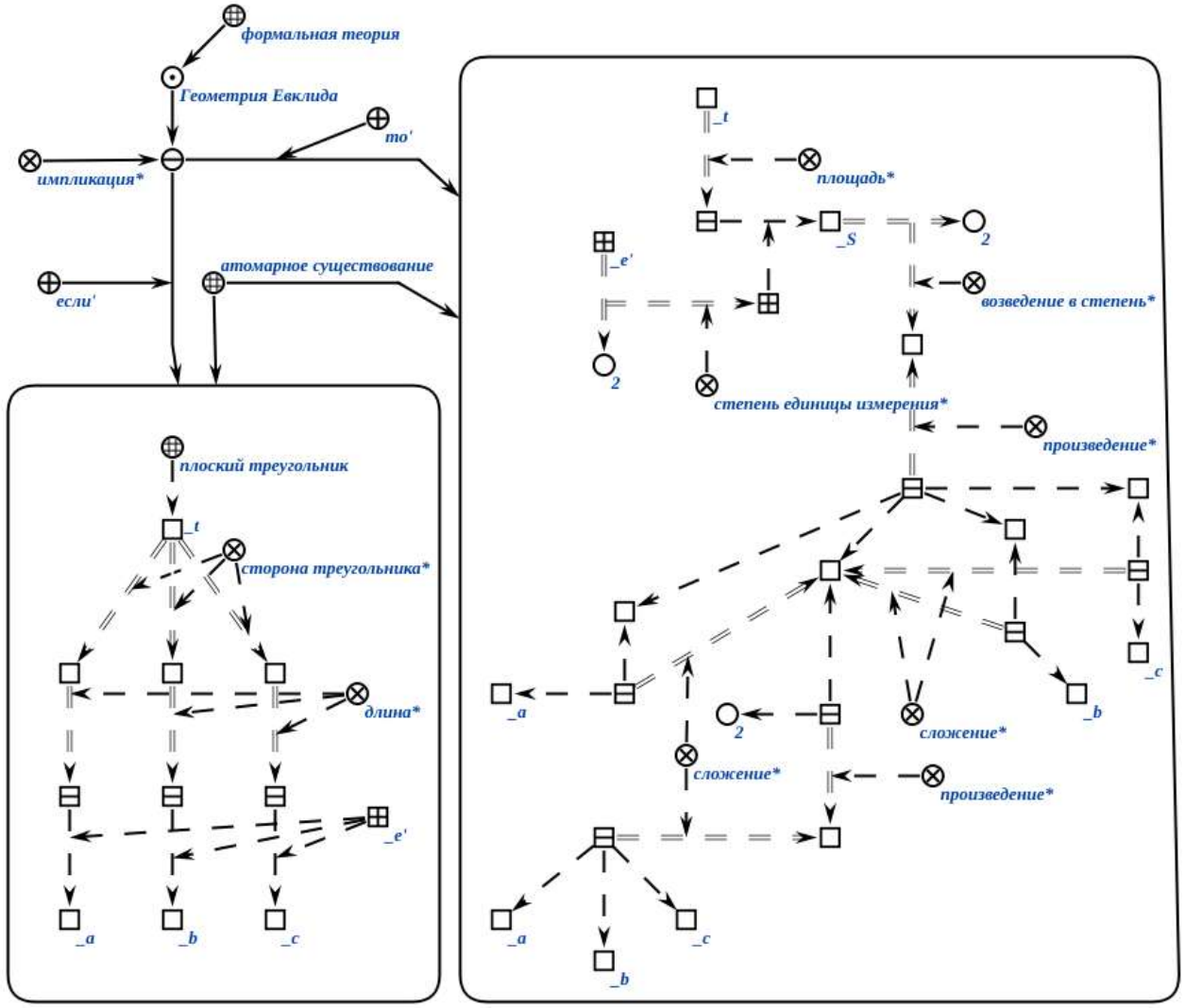
SCg-текст. Результат работы операции find_value
=



2. Снова запускаются операции find_value и find_formula, при этом выполнение find_value завершается, так как конструкция не полностью соответствует условиям запуска.

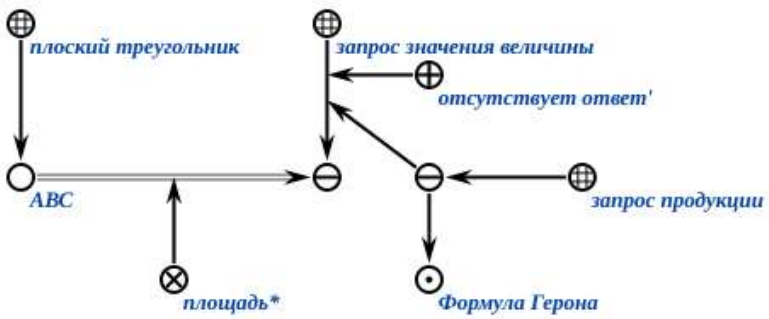
Операция find_formula производит поиск в базе знаний подходящей формулы, которая позволила бы вычислить значение требуемой величины у указанного объекта. Под формулой понимается некоторое импликативное логическое высказывание, справедливое для произвольного класса объектов, которому принадлежит рассматриваемый объект (в данном случае — треугольник), в посылке которого описаны требуемые для вычисления значения, а в заключении — собственно арифметическое выражение, вычисление которого приводит к получению требуемого результата (см. *SCg-текст. Пример формулы — формула Герона для вычисления площади треугольника*). В данном случае также используется сокращенная форма высказывания о всеобщности, то есть по умолчанию квантором всеобщности связываются те переменные, которые присутствуют в обеих частях высказывания.

SCg-текст. Пример формулы — формула Герона для вычисления площади треугольника
=



При просмотре каждой из формул производится проверка на соответствие предполагаемого результата желаемому и проверка наличия в базе знаний всей необходимой информации. Например, в данном случае проверяется тот факт, что формула Герона позволяет вычислить именно площадь (а не периметр) треугольника (а не четырехугольника или круга). Далее проверяется тот факт, что в базе присутствуют длины всех трех сторон данного треугольника, что позволит воспользоваться именно формулой Герона. В противном случае перебор формул продолжается. В данном случае перебор формул заканчивается на формуле Герона. В памяти генерируется запрос на подстановку конкретных значений в формулу (см. *SCg-текст. Запрос на унификацию формулы*).

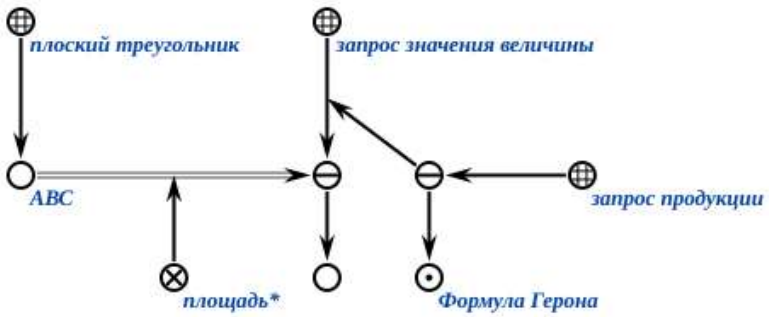
SCg-текст. Запрос на унификацию формулы
=



3. Запускается операция find_value_production, задачей которой является унификация предложенной формулы константами из базы знаний. Операция использует посылку формулы для поиска значений, соответствующих именно указанному объекту (в данном случае – длин треугольника ABC, а не какого-либо другого треугольника). После этого отбираются значения переменных, связанных квантором всеобщности, и производится генерация арифметического выражения с подстановкой в него значений связанных переменных из формулы. Результатом работы является константное арифметическое выражение, требующее вычисления, о чем сообщается путем генерации соответствующей конструкции.
4. Далее запускается операция calculation, алгоритм и условия срабатывания которой подробно рассмотрены выше.

В результате последовательного выполнения указанного набора операций у указанного объекта явно указывается значение требуемого параметра (см. *SCg-текст. Результат вычисления значения площади треугольника ABC*).

SCg-текст. Результат вычисления значения площади треугольника ABC
=



§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах

⇒ *подраздел**:
- *Пункт 7.5.3.1. Существующие научно-исследовательские результаты и проблемы в области автоматизации контроля знаний*
- *Пункт 7.5.3.2. Предлагаемый подход к автоматизации контроля знаний*
- *Пункт 7.5.3.3. Семантическая модель базы знаний подсистемы контроля знаний*
- *Пункт 7.5.3.4. Семантическая модель решателя задач подсистемы контроля знаний*

⇒ *ключевое понятие**:
- *генерация тестовых вопросов*
- *интеллектуальные обучающие системы*
- *проверка ответов*

⇒ *ключевое знание**:
- *установление отношений отображения онтологии*
- *вычисление семантического сходства*

⇒ *библиографическая ссылка**:
- *Xu G.P..Resea oITS-2009art*
- *Голенков В.В..ИнтелОСиВУ-2001кн*
- *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Li W..OntolAfQGaKC-2020art*
- *Li W..Devel oaPSfAAV-2021art*
- *МетасOSTIS-2022эл*
- *Qian L..OntolAfCLID-2020art*
- *Mousavinasab E..IntelTSaSRoC-2018art*
- *Bhatia A..AutomGoMCQ-2013art*
- *Papasalouros A..AutomGoMCQ-2008art*
- *Protege-2016el*
- *Li H.Resea oIAGBoD-2012art*

- *Wan C..aRevie oTSCM-2019art*
- *Li X.Reali oASAfSQ-2009art*
- *Wan HR..RevieoRPoTS-2019art*
- *Shahmirzadi O..aTextSiVSM-2019art*
- *Ji M..aShortTSCM-2022art*
- *Anderson P..SPICE-2016art*
- *Fujiwara M..PreneNFTiS-2021art*
- *Zeng K..aComprSoEAfKG-2021art*
- *Sun Z..aBenchSoEEA-2020art*
- *Rujiang W..Revie oCPTaM-2011art*
- *Kowalski R.PrediLaPL-1974art*
- *Krom M.tDecisPfFiPC-1970art*
- *Zhang J..AutomPoTPfTiEGItHIPT-1995art*
- *Zhang J..Self-EAPBoPGftPoWMI-2019art*
- *Шункевич Д.В..МетодКПСУЗ-2013cm*

Введение в Параграф § 7.5.3.

Данный параграф посвящен проблеме генерации тестовых вопросов и проверки ответов пользователей в *интеллектуальных обучающих системах*. В данном параграфе подробно представлен подход к автоматической генерации тестовых вопросов различных типов на основе *базы знаний* в *интеллектуальных обучающих системах*, разработанных с использованием *Технологии OSTIS*, и подход к реализации автоматической проверки ответов пользователей на основе различных семантических структур описанных знаний.

Применение технологии искусственного интеллекта в сфере образования может не только повысить эффективность обучения учащихся, но и стать важным средством обеспечения справедливости образования. Особенно после вспышки COVID-19 в 2020 году была подчеркнута важность и актуальность разработки *интеллектуальных обучающих систем* (см. *Xu G.P..Resea oITS-2009art*). По сравнению с традиционной мультимедийной обучающей системой (м.о.с.), и.о.с. имеет следующие характеристики:

- способен вести свободный человеко-машинный диалог;
- предоставление персонализированной педагогической услуги;
- автоматическое решение тестовых вопросов;
- автоматическая генерация тестовых вопросов;
- автоматическая проверка ответов пользователей;
- и так далее.

Среди перечисленных характеристик автоматическая генерация тестовых вопросов и автоматическая проверка ответов пользователей являются самыми основными и важными функциями и.о.с. Они позволяют автоматизировать весь процесс от генерации тестовых вопросов и формирования экзаменационных билетов до автоматической проверки ответов пользователей и оценки экзаменационных билетов. Это может не только значительно повысить эффективность тестирования уровня знаний пользователей, но и снизить стоимость их обучения, при этом исключая человеческий фактор, чтобы максимально обеспечить справедливость процесса тестирования.

Хотя в последние годы с развитием семантической сети, обработки естественного языка (NLP) и других соответствующих технологий некоторыми научно-исследовательскими группами были предложены и разработаны подходы и системы для автоматической генерации тестовых вопросов и автоматической проверки ответов пользователей, эти подходы и системы имеют много недостатков, таких как:

- только генерация простых объективных вопросов;
- большинство существующих подходов и систем проверки ответов поддерживает только проверку ответов пользователей на объективные вопросы;
- некоторые существующие подходы к проверке ответов пользователей на субъективные вопросы основаны на сопоставлении ключевых слов и статистике вероятности и не учитывают семантическое подобие между ответами;
- частично основанные на семантике подходы к проверке ответов пользователей на субъективные вопросы могут вычислять только подобие между ответами с простыми семантическими структурами;
- компоненты, разработанные с использованием существующих подходов к генерации тестовых вопросов и проверке ответов пользователей, могут быть использованы только в соответствующих системах;
- не поддерживается автоматизированная реализация всего процесса от генерации тестовых вопросов до проверки ответов пользователей (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014cm*, *Golenkov V.V..Metho aTfECoC-2019art*, *Li W..OntolAfQGaKC-2020art*).

К типу вопросов с уникальным стандартным ответом относятся объективные вопросы, которые включают в себя: вопросы на выбор, вопросы суждения и вопросы на толкование определений. Субъективные вопросы не имеют уникальных ответов, а общие субъективные вопросы включают вопросы на доказательство, вопросы на толкование определений и решение задачи (см. *Li W..Devel oaPSfAAV-2021art*).

В связи с этим в данном параграфе представлен подход к автоматической генерации тестовых вопросов и автоматической проверке ответов пользователей в обучающих системах, разработанных с использованием *Технологии OSTIS*, и на основе предложенного подхода разработана универсальная подсистема для автоматической генерации тестовых вопросов и автоматической проверки ответов пользователей. Основной принцип автоматической генерации тестовых вопросов в данной работе заключается в том, чтобы сначала обобщить ряд стратегий генерации тестовых вопросов на основе структуры базы знаний ostis-систем и структуры представления знаний в ней, а затем использовать эти стратегии генерации тестовых вопросов для извлечения соответствующих семантических фрагментов из базы знаний и генерировать семантические модели, соответствующие тестовым вопросам (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*, *МетасOSTIS-2022эл*). Основной принцип проверки ответа на тестовый вопрос заключается в том, чтобы сначала вычислить подобие между семантическим фрагментом стандартного ответа и семантическим фрагментом ответа пользователя, а затем осуществить автоматическую проверку ответа пользователя на основе вычисленного подобия и стратегии оценки соответствующего тестового вопроса. Семантический фрагмент представляет собой сеть, которая отображает семантические отношения между понятиями. В ostis-системах семантический фрагмент строится с помощью *SC-кода* (см. *МетасOSTIS-2022эл*, *Li W..OntolAfQGaKC-2020art*). Следует подчеркнуть, что семантический фрагмент, соответствующий тестовому вопросу, и соответствующее ему описание на естественном языке преобразуются друг в друга с помощью естественно-языковых интерфейсов (см. *Qian L..OntolAfCLID-2020art*). Подход, предложенный в данном параграфе, должен решить следующие задачи:

- автоматическая генерация ряда тестовых вопросов из базы знаний и сохранение их в соответствующих разделах базы знаний подсистемы;
- проектирование и построение баз знаний подсистем для хранения сгенерированных тестовых вопросов;
- извлечение соответствующих типов тестовых вопросов и составление экзаменационных билетов в соответствии с потребностями пользователей;
- вычисление подобия между семантическими фрагментами ответов на объективные вопросы;
- вычисление подобия между семантическими фрагментами ответов на вопросы на толкование определений;
- вычисление подобия между семантическими фрагментами ответов на вопросы на доказательство и на решение задачи;
- автоматическая проверка ответов на тестовые вопросы и автоматическая оценка экзаменационных билетов на основе вычисленного подобия и стратегии оценки соответствующих тестовых вопросов.

Следует подчеркнуть, что предлагаемый в данном параграфе подход не опирается на какой-либо естественный язык, но для того, чтобы объяснить принцип работы предлагаемого подхода, подобранные в данном параграфе семантические фрагменты и иллюстрации представлены на русском языке. Среди них *ostis-система* по дискретной математике и *ostis-система* по евклидовой геометрии будут использоваться в качестве демонстрационных систем для подсистемы, разработанной с использованием предлагаемого подхода.

Пункт 7.5.3.1. Существующие научно-исследовательские результаты и проблемы в области автоматизации контроля знаний

⇒ *подраздел**:

- *Подпункт 7.5.3.1.1 Автоматическая генерация тестовых вопросов*
- *Подпункт 7.5.3.1.2 Автоматическая проверка ответов пользователей*

Подпункт 7.5.3.1.1 Автоматическая генерация тестовых вопросов

Подход к автоматической генерации тестовых вопросов в основном изучает, как использовать электронные документы, корпуса текстов и базы знаний для быстрой и гибкой автоматической генерации тестовых вопросов. Благодаря тому, что знания в базе знаний представляют собой высокоструктурированные знания, прошедшие фильтрацию, и с развитием семантических сетей, использование базы знаний для автоматической генерации тестовых вопросов стало важнейшим направлением исследований в области автоматической генерации тестовых вопросов (см. *Xu G.P..Resea oITS-2009art; Mousavinasab E..IntelTSaSRoC-2018art; Bhatia A..AutomGoMCQ-2013art*). Некоторые результаты исследований приведены ниже:

- подход к использованию классов, экземпляров, атрибутов и отношений между ними в онтологии OWL для генерации вопросов на выбор представлен в работе (см. *Papasalouros A..AutomGoMCQ-2008art*). OWL представляет собой язык описания онтологий для семантической сети. Онтология — это вид знаний, каждое

из которых является спецификацией соответствующей предметной области, ориентированной на описание свойств и взаимосвязей понятий, входящих в состав указанной предметной области;
- подход к автоматической генерации объективных вопросов с использованием онтологии, созданной Protégé (см. *Protege-2016el*), представлен в работе (см. *Li H.Resea oIAGBoD-2012art*).

Эти подходы в основном имеют следующие проблемы:
- подход к использованию электронных документов для автоматической генерации тестовых вопросов требует большого количества шаблонов предложений;
- создание корпуса текстов требует больших человеческих ресурсов для сбора и обработки различных знаний;
- существующие подходы могут быть использованы только в соответствующих системах и не являются совместимыми;
- существующие подходы позволяют генерировать только простые объективные вопросы.

Подпункт 7.5.3.1.2 Автоматическая проверка ответов пользователей

Автоматическая проверка ответов пользователей делится на проверку ответов на объективные вопросы и проверку ответов на субъективные вопросы. Основной принцип проверки ответов на объективные вопросы относительно прост, то есть достаточно определить, совпадает ли строка стандартного ответа и строка ответа пользователя. Ответы на субъективные вопросы обычно не являются уникальными, поэтому основной принцип проверки ответов на субъективные вопросы заключается в вычислении подобия между стандартным ответом и ответом пользователя, а затем в осуществлении автоматической проверки ответов пользователя на основе вычисленного подобия и стратегии оценки соответствующих тестовых вопросов. Чем больше похожи стандартный ответ и ответ пользователя, тем выше подобие между ними (см. *Wan C..aRevie oTSCM-2019art; Li X.Reali oASAfSQ-2009art; Wan HR..RevieoRPoTS-2019art*). Проверка ответов на субъективные вопросы делится на следующие категории в соответствии с подходом, используемым для вычисления подобия:
- На основе ключевых словосочетаний

  Этот тип подхода позволяет сначала разделить предложения на ключевые словосочетания, а затем вычислить подобие между ними в соответствии с отношениями совпадения ключевых словосочетаний между предложениями. Представительные подходы включают:
  - N-gram similarity
  - Jaccard similarity
- На основе модели векторного пространства (VSM)

  Основной принцип VSM заключается в использовании традиционных алгоритмов машинного обучения для того, чтобы сначала преобразовать предложения в векторные представления, а затем вычислить подобие между ними (см. *Shahmirzadi O..aTextSiVSM-2019art*). Представительные подходы включают:
  - TF-IDF
  - Word2vec
  - Doc2Vec
- На основе глубокого обучения

  Этот тип подхода позволяет использовать модели нейронных сетей для вычисления подобия между предложениями (см. *Ji M..aShortTSCM-2022art*). Представительные модели нейронных сетей включают:
  - Tree-LSTM
  - Transformer
  - BERT
- На основе семантического фрагмента

  Основной принцип вычисления подобия между ответами с использованием данного типа подхода заключается в том, чтобы сначала преобразовать ответы (то есть предложения или короткие тексты) в представление семантического фрагмента с помощью инструментов обработки естественного языка (например, синтаксические деревья зависимостей и естественно-языковые интерфейсы), а затем вычислить подобие между семантическими фрагментами (то есть подобие между ответами). В *и.о.с.* различная информация хранится в виде семантических фрагментов, поэтому можно рассмотреть возможность вычисления подобия между любыми двумя семантическими фрагментами в базе знаний, опираясь на принципы работы данного типа подхода. Основным преимуществом этого типа подхода является вычисление подобия между ответами на основе семантики. Одним из наиболее представительных подходов является SPICE (Semantic Propositional Image Caption Evaluation) (см. *Anderson P..SPICE-2016art*).

  Подход SPICE используется для вычисления подобия между автоматически сгенерированными подписями к рисункам (подписи-кандидаты) и подписями к рисункам, помеченными вручную (подписи-образцы). Данный

подход позволяет вычислить подобие между подписями путем сопоставления одного и того же числа кортежей между семантическими кортежами подписи-кандидатов и семантическими кортежами подписи-образцов.

Эти подходы в основном имеют следующие недостатки:
- подход, основанный на ключевых словосочетаниях, не учитывает порядок между словами в предложении;
- подход на основе VSM приводит к генерации высокоразмерных разреженных матриц, что увеличивает сложность алгоритма;
- подходы на основе семантических фрагментов, поддерживающие только описание простых семантических структур;
- эти подходы не позволяют определить, являются ли предложения логически эквивалентными друг другу;
- эти подходы зависят от соответствующего естественного языка.

Поэтому на основе существующих подходов к автоматической генерации тестовых вопросов с использованием баз знаний, подходов к вычислению подобия между ответами с использованием семантических фрагментов и *Технологии OSTIS* в данном параграфе предлагается подход к автоматической генерации тестовых вопросов и автоматической проверке ответов пользователей с использованием семантики.

Пункт 7.5.3.2. Предлагаемый подход к автоматизации контроля знаний

⇒ *подраздел**:
- *Подпункт 7.5.3.2.1 Предлагаемый подход к автоматической генерации тестовых вопросов*
- *Подпункт 7.5.3.2.2 Предлагаемый подход к автоматической проверке ответов пользователей*
- *Подпункт 7.5.3.2.3 Проверка ответов на объективные вопросы*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*

Основной задачей данном параграфе является детализация подхода к автоматической генерации тестовых вопросов и автоматической проверке ответов пользователей в *ostis-системах* и разработка универсальной подсистемы на основе этого подхода. Где универсальность подсистемы означает, что подсистема может быть легко перенесена между различными *ostis-системами*. Предлагаемый подход можно разделить на две части в соответствии с реализуемыми функциями, то есть автоматическая генерация тестовых вопросов и автоматическая проверка ответов пользователей (см. *Li W..Devel oaPSfAAV-2021art*). Поэтому мы представим процесс реализации этих двух частей отдельно.

Подпункт 7.5.3.2.1 Предлагаемый подход к автоматической генерации тестовых вопросов

Основной принцип автоматической генерации различных типов тестовых вопросов (включая объективные вопросы и субъективные вопросы) в *ostis-системах* заключается в том, чтобы сначала извлечь соответствующие семантические фрагменты из базы знаний, используя ряд стратегий генерации тестовых вопросов, обобщенных на основе подхода представления знаний и структуры описания знаний в рамках *Технологии OSTIS*, затем добавить к извлеченным семантическим фрагментам информацию об описании тестового вопроса и, наконец, сохранить семантические фрагменты, описывающие полные тестовые вопросы, в соответствующем разделе подсистемы (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст*). Когда необходимо сформировать экзаменационные билеты, подсистема позволяет извлечь из базы знаний подсистемы несколько соответствующих тестовых вопросов в соответствии с параметрами, введенными пользователем, и объединить их в экзаменационные билеты. Тестовые вопросы и экзаменационные билеты в виде семантических фрагментов преобразуются в описания на естественном языке с помощью естественно-языковых интерфейсов. В работе (см. *Li W..OntolAfQGaKC-2020art*) мы подробно рассмотрели некоторые стратегии, используемые для автоматической генерации тестовых вопросов в *ostis-системах*, далее мы выберем только некоторые из стратегий генерации тестовых вопросов для представления.

- Стратегия генерации тестовых вопросов на основе класса

  Этот тип стратегии генерации тестовых вопросов используется для автоматической генерации объективных вопросов, основанных на различных отношениях между классами. Далее она делится на:

  - На основе отношения *включение**

    Отношение включения является одним из наиболее часто используемых отношений в *базе знаний ostis-систем*, которое удовлетворяется между многими классами (включая подклассы), поэтому отношение включения между классами может быть использовано для генерации объективных вопросов. Форма выражения в теории множеств отношения включения между классами выглядит следующим образом: $S_i \subseteq C$, (S — подкласс, i — номер подкласса, C — родительский класс). Ниже показан семантический фрагмент в базе знаний, который удовлетворяет отношению включения в *SCn-коде* (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст; МетасOSTIS-2022эл*):

бинарное дерево
⇐ *включение**:
ориентированное дерево
⇒ *включение**:
- *братское дерево*
- *дерево решений*
- *бинарное дерево сортировки*

В качестве примера вопроса на выбор, сгенерированного с использованием этого семантического фрагмента на основе стратегии отношений включения, ниже показана его форма на естественном языке:

«Частным случаем бинарного дерева не является ( )?»

A. дерево решений C. ориентированное дерево
B. братское дерево D. бинарное дерево сортировки

Пример семантической модели данного тестового вопроса приведен на рисунке (*SCg-текст. Пример семантической модели вопроса на выбор*) в *SCg-коде*.

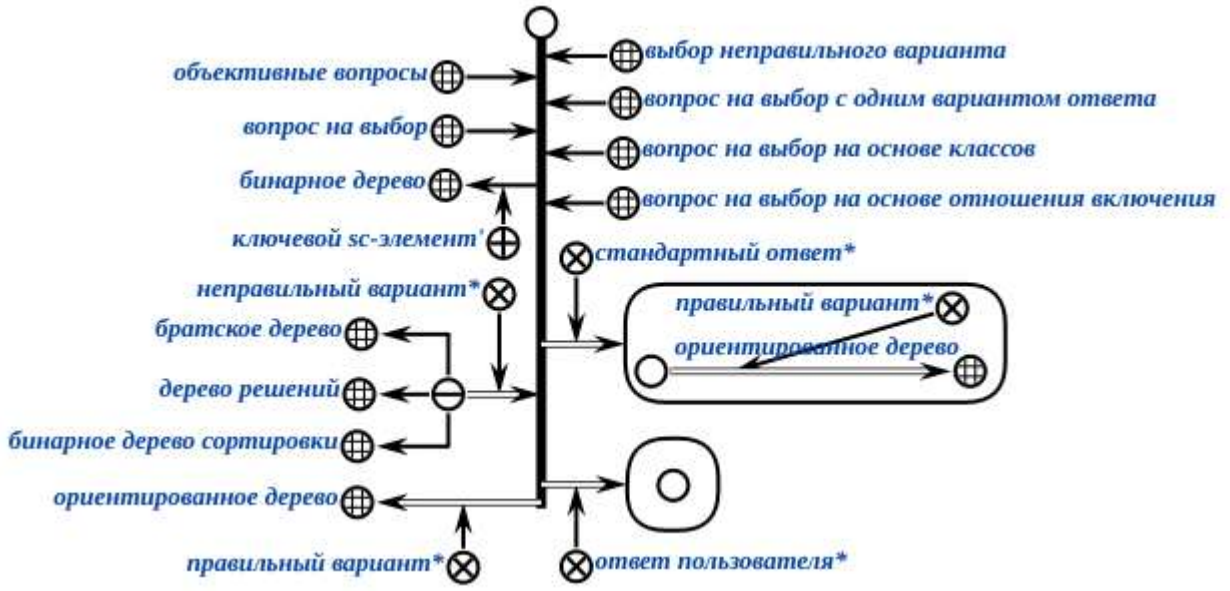
Аналогичным образом, используя эту стратегию, можно генерировать другие типы объективных вопросов;

- На основе отношения *разбиение**

Отношение разбиения — это квазибинарное ориентированное отношение, областью определения которого является семейство всевозможных множеств. В результате разбиения множества получается множество попарно непересекающихся множеств, объединение которых есть исходное множество. Отношение разбиения также является важным отношением в базе знаний, поэтому семантические фрагменты в базе знаний, удовлетворяющие этому отношению, могут быть использованы для генерации объективных вопросов. На языке теории множеств отношение разбиения между классами описывается следующим образом: $S_{i_1} \cup S_{i_2} \cdots S_{i_m} \cup S_{i_n} = S_{j_1} \cup S_{j_2} \cdots S_{j_m} \cup S_{j_n} = \cdots = U$, ($n > 1$, $S_{j_m} \cap S_{j_n} = \oslash$). Ниже приведен семантический фрагмент в базе знаний, удовлетворяющий отношению *разбиение** в *SCn-коде*:

граф
⇒ *включение**:
полуэйлеров граф
⇒ *разбиение**:
{• *невзвешенный граф*
• *взвешенный граф*
}
⇒ *разбиение**:
{• *непланарный граф*
• *планарный граф*
}
⇒ *разбиение**:
{• *несвязный граф*
• *связный граф*
}

SCg-текст. Пример семантической модели вопроса на выбор
=



- На основе отношения *строгое включение**

  Отношение строгого включения является особой формой отношения включения ($S_i \subset C$, ($i \geq 1$)). Использование отношения строгого включения для автоматической генерации объективных вопросов аналогично использованию отношения включения. Ниже приведен семантический фрагмент в базе знаний, удовлетворяющий отношению *строгое включение** в *SCn-коде*:

  Предметная область множеств
  ∋ *немаксимальный класс объектов исследования'*:
  - *счетное множество*
  - *ориентированное множество*
  - *конечное множество*

  ⇒ *включение**:
  - *пара*
  - *тройка*

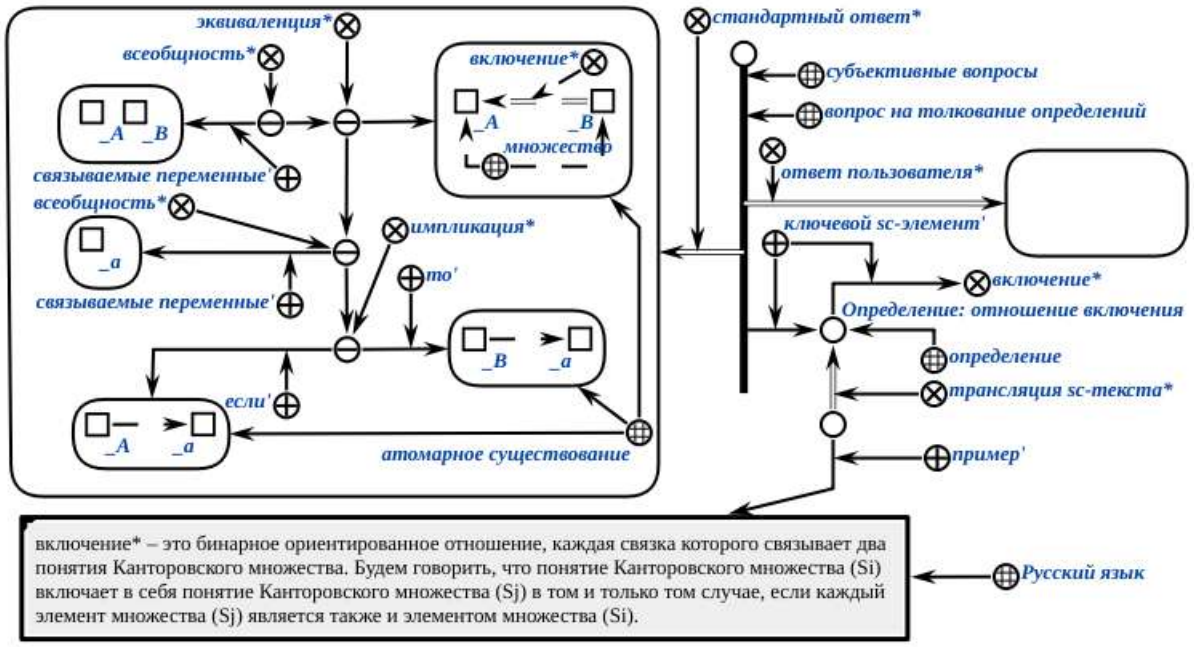
Другие стратегии, используемые для генерации объективных вопросов, также включают:
- Стратегия генерации тестовых вопросов на основе элементов;
- Стратегия генерации тестовых вопросов на основе идентификаторов;
- Стратегия генерации тестовых вопросов на основе аксиом;
- Стратегия генерации тестовых вопросов на основе атрибутов отношений;
- Стратегия генерации тестовых вопросов на основе примеров изображений.

Конкретный процесс генерации объективных вопросов с использованием перечисленных выше стратегий подробно представлен в работе (см. *Li W..OntolAfQGaKC-2020art*).

Процесс генерации субъективных вопросов с использованием стратегии генерации субъективных вопросов выглядит следующим образом:
- поиск в базе знаний семантических фрагментов, описывающих субъективные вопросы с использованием логических правил (то есть шаблоны, построенные с использованием *SC-кода*);
- хранение найденных семантических фрагментов в соответствующем разделе базы знаний подсистемы;
- использование ручных подходов или автоматических подходов (например, с помощью естественно-языковых интерфейсов) для описания определения, процесса доказательства или процесса решения соответствующего тестового вопроса в соответствии с правилами представления знаний (то есть стандартные ответы на субъективные вопросы). Среди них стандартные ответы на субъективные вопросы представлены с помощью *SCg-кода* или *Языка SCL* (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст; МетасOSTIS-2022эл*).

Пример семантической модели субъективного вопроса показан на рисунке (*SCg-текст. Семантическая модель определения отношения включения*) в *SCg-коде*.

SCg-текст. Семантическая модель определения отношения включения
=



На рисунке (*SCg-текст. Семантическая модель определения отношения включения*) описано определение отношения включения $(\forall A \forall B((A \subseteq B) \iff (\forall a(a \in A \rightarrow a \in B)))$.

Использование этих стратегий генерации тестовых вопросов, описанных выше, позволяет генерировать различные типы тестовых вопросов автоматически из базы знаний. Эти автоматически сгенерированные тестовые вопросы хранятся в базе знаний подсистемы в соответствии с их типом и соответствующей стратегией генерации тестовых вопросов. Такой тип хранения позволяет быстро и динамично генерировать экзаменационные билеты в соответствии с потребностями пользователя. В следующем подразделе мы подробно опишем построение базы знаний подсистемы и способ хранения в ней тестовых вопросов. Предлагаемый подход к генерации тестовых вопросов имеет следующие преимущества:

- *Технология OSTIS* поддерживает унифицированные подходы к представлению знаний и структуры описания знаний, поэтому предложенный подход к генерации тестовых вопросов может быть использован в различных *ostis-системах*;
- сгенерированные тестовые вопросы описываются с помощью *SC-кода*, поэтому они не опираются на какой-либо естественный язык;
- используя предложенный подход к генерации тестовых вопросов, можно генерировать не только объективные вопросы, но и субъективные вопросы.

Подпункт 7.5.3.2.2 Предлагаемый подход к автоматической проверке ответов пользователей

В *ostis-системах* тестовые вопросы хранятся в базе знаний в виде семантических фрагментов, поэтому наиболее важным этапом проверки ответов пользователей является вычисление подобия между семантическим фрагментом стандартного ответа и семантическим фрагментом ответа пользователя, и когда подобие получено и объединено со стратегией оценки соответствующих тестовых вопросов, правильность и полнота ответов пользователей могут быть проверены (см. *Golenkov V.V..Metho aTfECoC-2019art; Li W..Devel oaPSfAAV-2021art*).

В соответствии с типом тестовых вопросов проверка ответов пользователей классифицируется как:

- проверка ответов на объективные вопросы;
- проверка ответов на субъективные вопросы.

Хотя наиболее важным этапом проверки ответов является вычисление подобия между семантическими фрагментами ответов, типы знаний (фактические знания и логические знания) и структуры знаний, используемые для описания различных типов тестовых вопросов, не одинаковы, поэтому подходы к вычислению подобия между семантическими фрагментами ответов на различные типы тестовых вопросов различны. Фактические знания относятся к знаниям, которые не содержат типов переменных, и этот тип знаний выражает факты. Логические знания обычно содержат переменные, и между ними существуют логические отношения. В *ostis-системах Язык SCL* используется для описания логических знаний. В *ostis-системах* объективные вопросы, вопросы на доказательство

и решение задачи описываются с использованием фактических знаний, а вопросы на толкование определений описываются с использованием фактических и логических знаний вместе.

Подпункт 7.5.3.2.3 Проверка ответов на объективные вопросы

Семантические фрагменты, используемые для описания объективных типов тестовых вопросов и ответов на них в базе знаний, имеют одинаковую семантическую структуру, поэтому подобие между ответами на такие типы тестовых вопросов может быть вычислено с использованием того же подхода. Поскольку ответы пользователей на естественном языке на объективные вопросы уже согласованы с существующими знаниями в базе знаний, когда они преобразуются в семантические фрагменты с помощью естественно-языкового интерфейса, то есть элементы, представляющие одну и ту же семантику в базе знаний, имеют один и тот же основной идентификатор (см. *Qian L..OntolAfCLID-2020art*). Поэтому при вычислении подобия между семантическими фрагментами ответов на объективные вопросы нет необходимости учитывать различия между понятиями на уровне естественного языка, то есть подобие между ответами вычисляется на основе семантических структур. Для проверки ответов пользователей на объективные вопросы необходимо решить следующие задачи:

- вычисление подобия между семантическими фрагментами ответов на объективные вопросы;
- определение того, существует ли логическая эквивалентность между семантическими фрагментами ответов на объективные вопросы (семантическими фрагментами ответов, описанными на основе фактических знаний);
- использование вычисленного подобия и стратегий оценки объективных вопросов для проверки правильности и полноты ответов пользователей и подсчета баллов за ответы пользователей.

Логическая эквивалентность между семантическими фрагментами в *ostis-системах* делится на два типа:

- логическая эквивалентность между семантическими фрагментами, описанными на основе логических формул;
- логическая эквивалентность между семантическими фрагментами, описанными на основе различных систем понятий (различных по структуре семантических фрагментов). Этот тип логической эквивалентности далее классифицируется в зависимости от типа знания:
  - логическая эквивалентность между семантическими фрагментами, описанными на основе фактических знаний;
  - логическая эквивалентность между семантическими фрагментами, описанными на основе логических знаний.

Основной принцип вычисления подобия между семантическими фрагментами ответов на объективные вопросы показан ниже:

- декомпозиция семантического фрагмента стандартного ответа (s) и семантического фрагмента ответа пользователя (u) на подструктуры в соответствии с правилами представления фактических знаний;
- использование формул (7.5..1), (7.5..2) и (7.5..3) для вычисления точности (P_{sc}), полноты (R_{sc}) и подобия (F_{sc}) между семантическими фрагментами.

$$P_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(u)|} \tag{7.5..1}$$

$$R_{sc}(u,s) = \frac{|T_{sc}(u) \otimes T_{sc}(s)|}{|T_{sc}(s)|} \tag{7.5..2}$$

$$F_{sc}(u,s) = \frac{2 \cdot P_{sc}(u,s) \cdot R_{sc}(u,s)}{P_{sc}(u,s) + R_{sc}(u,s)} \tag{7.5..3}$$

где:

- ($\otimes$) — бинарный оператор сопоставления;
- множество $T_{sc}(s)$ — все разложенные подструктуры стандартного ответа s;
- множество $T_{sc}(u)$ — все разложенные подструктуры стандартного ответа u;

Поскольку подсистема также должна позволять вычислять подобие между любыми двумя фрагментами в базе знаний, вычисленное подобие может быть использовано в будущем для проверки ответов пользователя на новые типы тестовых вопросов и для решения других задач, таких как интеграция знаний. В базе знаний содержится большое количество семантических фрагментов, имеющих схожую структуру с семантическими фрагментами ответов на объективные вопросы, то есть семантических фрагментов, описанных с помощью фактических знаний, поэтому подход, используемый для вычисления подобия между ответами на объективные вопросы, также может быть использован для вычисления подобия между семантическими фрагментами, описанными с помощью фактических знаний. Поскольку семантический фрагмент ответов на объективные вопросы и семантический фрагмент, описанный с использованием фактических знаний, имеют схожую семантическую структуру, далее они будут единообразно именоваться семантическим фрагментом ответов на объективные вопросы. Но семантические фраг-

менты этого типа обычно имеют семантические фрагменты, логически эквивалентные им, например, определение симметричной разности может быть выражено в этих двух формах:

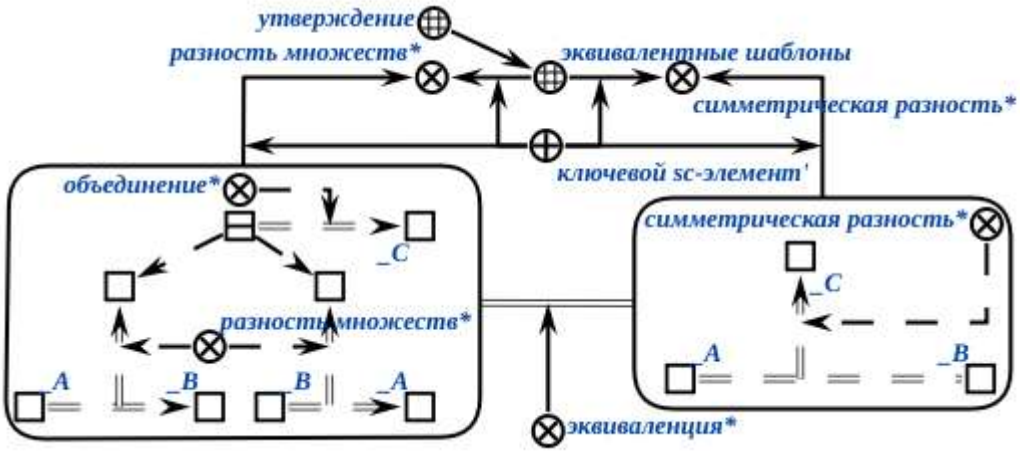
- $C = (A \setminus B) \cup (B \setminus A)$;
- $C = A \triangle B$.

Поэтому, если вычисленное подобие между семантическими фрагментами не равно 1, необходимо также определить, удовлетворяется ли между ними логическая эквивалентность. Поэтому в данном параграфе представлен подход к определению логической эквивалентности между семантическими фрагментами, описанными на основе фактических знаний.

Процесс определения логической эквивалентности между семантическими фрагментами такого типа показан ниже:

1. найдены все sc-узлы в семантическом фрагменте стандартного ответа и все sc-узлы в семантическом фрагменте ответа пользователя соответственно. Затем проверяется, существует ли пара sc-узлов между sc-узлами стандартного ответа и sc-узлами ответа пользователя, и ее два sc-узла соответственно включены в шаблон, связанный с использованием отношения “эквиваленция*”. Если такая пара sc-узлов существует, выполняется следующий шаг. Пример пары шаблонов показан на рисунке (*SCg-текст. Пара шаблонов, удовлетворяющих логической эквивалентности*) в *SCg-коде*;

SCg-текст. Пара шаблонов, удовлетворяющих логической эквивалентности
=



2. использование двух шаблонов для поиска всех изоморфных семантических фрагментов в базе знаний и проверка наличия двух фрагментов пользователя в этих найденных фрагментах, которые соответственно включены в стандартный ответ и ответ пользователя. Если существуют такие два фрагмента (соответствие различным шаблонам), выполняется следующий шаг;
3. итеративно проходятся разложенные подструктуры стандартного ответа и разложенные подструктуры ответа пользователя, и каждая подструктура сравнивается с соответствующим семантическим фрагментом, найденным на шаге 2, если каждый sc-элемент в подструктуре содержится в соответствующем семантическом фрагменте, подструктура удаляется;
4. использование формул (7.5..1), (7.5..2) и (7.5..3) для вычисления подобия между семантическими фрагментами в соответствии с остальными подструктурами. Если подобие равно 1, то два семантических фрагмента полностью совпадают.

Пример определения логической эквивалентности семантических фрагментов приведен на рисунке (*SCg-текст. Пример суждения логической эквивалентности семантических фрагментов, описанных на основе фактических знаний*) в *SCg-коде*.

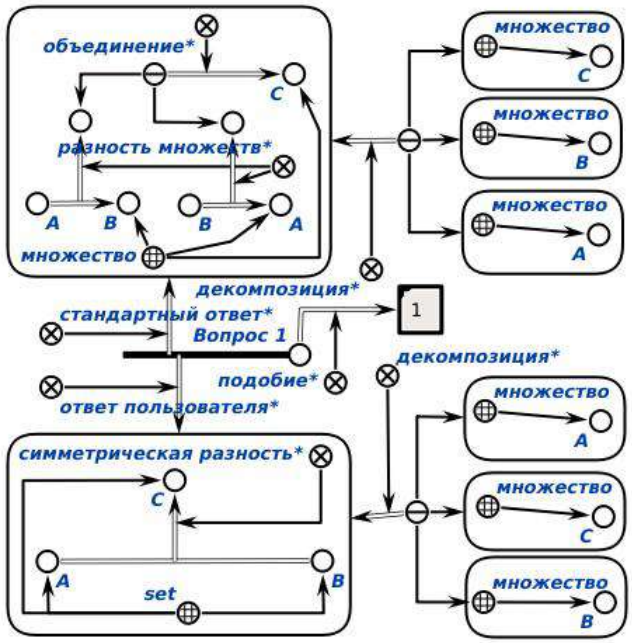
Когда получено подобие ответов, можно проверить правильность и полноту ответов пользователей на объективные вопросы, объединив их со стратегией оценки объективных вопросов. Стратегия оценки объективных вопросов показана ниже:

- если для текущего тестового вопроса существует только один правильный вариант, только если стандартный ответ и ответ пользователя точно совпадают, ответ пользователя считается правильным, и пользователь получает максимальный балл (Max_{score});
- если текущий вопрос имеет несколько правильных вариантов (например, вопросы на выбор с несколькими вариантами ответов и часть вопросов на заполнение пробелов):
  - до тех пор, пока ответ пользователя содержит неправильный вариант, ответ пользователя считается неправильным и оценка пользователя равна 0;

- если все варианты в ответе пользователя правильные, но количество правильных вариантов меньше, чем количество правильных вариантов в стандартном ответе, ответ пользователя считается правильным, но неполным. В это время оценка ответа пользователя равна $R_{sc} * Max_{score}$;
- если все варианты стандартного ответа точно совпадают со всеми вариантами ответа пользователя, то ответ пользователя точно правильный, а оценка пользователя равна Max_{score}.

SCg-текст. Пример суждения логической эквивалентности семантических фрагментов, описанных на основе фактических знаний

=



Пример проверки ответов на конкретный объективный вопрос показан на рисунке (*SCg-текст. Пример автоматической оценки вопроса на выбор с несколькими вариантами ответа*) в *SCg-коде*.

Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы

Наиболее важным этапом проверки ответов на субъективные вопросы также является вычисление подобия между семантическими фрагментами ответов, однако типы знаний и структуры знаний, используемые для описания различных типов субъективных вопросов и ответов на них, в *ostis-системах* не одинаковы. Таким образом, подход к вычислению подобия между семантическими фрагментами ответов на субъективные вопросы далее делится на:

- подход к вычислению подобия между ответами на вопросы на толкование определений;
- подход к вычислению подобия между ответами на вопросы на доказательство и на решение задачи.

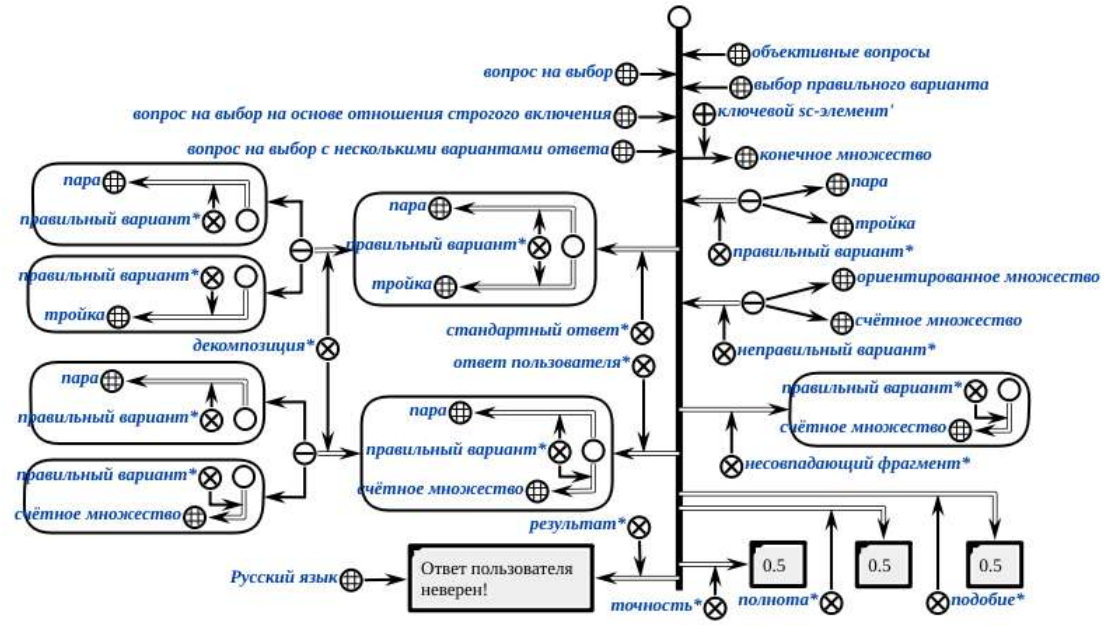
Вычисление подобия между ответами на вопросы на толкование определений

Ответы на вопросы на толкование определений в *ostis-системах* описываются в виде логических формул с использованием фактических знаний и логических знаний (*Язык SCL*). Логическая формула является мощным инструментом для формального представления знаний в рамках *Технологии OSTIS*, которая расширяется на основе формул логики предикатов первого порядка и наследует все операционные свойства формул логики предикатов первого порядка (см. *MemacOSTIS-2022эл*). Следует подчеркнуть, что при вычислении подобия между ответами на вопросы на толкование определений, фактические знания в семантических фрагментах ответов пользователей были согласованы с существующими знаниями в базе знаний (с использованием естественно-языковых интерфейсов) (см. *Qian L..OntolAfCLID-2020art*). Для вычисления подобия между семантическими фрагментами ответов на вопросы на толкование определений необходимо решить следующие задачи (см. *Fujiwara M..PreneNFTiS-2021art*):

- автоматический выбор потенциального эквивалентного стандартного ответа;
- установление отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами ответов;

- вычисление подобия между семантическими фрагментами;
- если подобие между семантическими фрагментами не равно 1, то их также необходимо отдельно преобразовать в представление префиксной нормальной формы (п.н.ф./PNF), а затем снова вычислить подобие между ними.

SCg-текст. Пример автоматической оценки вопроса на выбор с несколькими вариантами ответа
=



Некоторые вопросы на толкование определений иногда имеют несколько стандартных ответов, но логические формулы, используемые для их формального представления, не являются логически эквивалентными (описываются в соответствии с различными системами понятий). Например, определение отношения эквивалентности:

- в математике отношение эквивалентности является бинарным отношением, которое является рефлексивным, симметричным и транзитивным;
- для любого бинарного отношения, если оно является толерантным отношением и транзитивным, то оно является отношением эквивалентности.

Поэтому при вычислении подобия между ответами необходимо заранее отфильтровать стандартный ответ, который наилучшим образом соответствует ответу пользователя, из нескольких возможных стандартных ответов. Поэтому в данном параграфе предлагается подход к фильтрации стандартного ответа, который наилучшим образом соответствует ответу пользователя в соответствии с подобием предикатов между ответами. Принцип работы этого подхода показан ниже:

- нахождение всех предикатов в каждом ответе (неповторяющихся предикатов);
- вычисление подобия предикатов между ответом пользователя и каждым стандартным ответом с использованием формул (7.5..1), (7.5..2) и (7.5..3);
- стандартный ответ, который наиболее похож (максимальное подобие) на ответ пользователя, выбирается в качестве окончательного стандартного ответа.

Для того чтобы рассчитать подобие между семантическими фрагментами, наиболее важным шагом является установление отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами. Поэтому на основе существующих методов отображения онтологий (например, ASMOW, RiMOM и другие) предлагается подход к установлению отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами в соответствии с семантическими структурами (различными sc-конструкциями) (см. *Zeng K..aComprSoEAfKG-2021art; Sun Z..aBenchSoEEA-2020art; Rujiang W..Revie oCPTaM-2011art*).

В *ostis-системах* все *логические связки* (такие как *отрицание** ($\neg$) , *импликация** ($\rightarrow$) и так далее) и *кванторами* (такие как *квантор всеобщности* ($\forall$) и *квантор существования* ($\exists$)) являются *sc-связками* соответствующих отношений. Вопрос пользователя может быть представлен в виде атомарной логической формулы или *конъюнкцией** нескольких *логических формул* (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст; Rujiang W..Revie oCPTaM-2011art*). Все sc-связки и sc-коннекторы образуют дерево, которое полностью описывает логическую последовательность между связками и кванторами в логической формуле. Поскольку sc-структура, содержащая атомарную логическую формулу, связана с соответствующим sc-связкой, пока определена позиция каждой sc-связки и sc-структуры

в семантическом фрагменте, можно определить позицию каждой переменной sc-узла в этом семантическом фрагменте. В данном параграфе предлагается подход к нумерации каждой sc-связки и sc-структуры в семантическом фрагменте в соответствии со стратегией поиска в глубину (DFS). Процесс работы данного подхода показан ниже:

- каждая sc-связка и sc-структура в дереве нумеруется по очереди в соответствии со стратегией DFS и приоритетом текущей sc-связки (например, приоритет sc-связки “если’ ” выше, чем приоритет sc-структуры “то’ ”) (последовательность нумерации начинается с 0);
- в соответствии с последовательностью нумерации sc-связок, каждый sc-связка в дереве обходится от малого к большому, а sc-структура, связанная с текущей sc-связкой, нумеруется при обходе (последовательность нумерации начинается с 1).

При проверке ответа, если стандартный ответ и ответ пользователя точно равны, это означает, что атомарные логические формулы с одинаковой семантикой между ответами имеют одинаковое положение в семантическом фрагменте (то есть, последовательность нумерации sc-структуры одинакова). Поэтому в данном параграфе отношения отображения потенциальных эквивалентных пар переменных sc-узлов будут устанавливаться на основе отношений соответствия sc-конструкций в одной и той же позиции между ответами. Процесс установления отношений отображения потенциальных эквивалентных пар переменных sc-узлов между ответами показан ниже:

1. в соответствии с последовательностью нумерации sc-структур в семантическом фрагменте, каждый раз, когда из стандартного ответа и ответа пользователя найдена пара sc-структур с одинаковым номером;
2. в соответствии с порядком приоритета (от высокого к низкому) различных типов sc-конструкций, используемых для описания атомарной логической формулы, поочередно определяется, содержит ли текущая пара sc-структур одновременно данный тип sc-конструкции. Если этот тип sc-конструкции одновременно содержится в текущей паре sc-структур, то, в соответствии с отношением соответствия каждого sc-элемента между текущей sc-конструкцией в стандартном ответе и текущей sc-конструкцией в ответе пользователя, устанавливаются отношения отображения потенциальных эквивалентных пар переменных sc-узлов между текущими sc-конструкциями;
3. повторять шаг 1 — шаг 2, пока не будут установлены все отношения отображения между семантическими фрагментами (см. *Li W..Devel oaPSfAAV-2021art*).

На рисунке (*SCg-текст. Пример установления отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами*) рассмотрено установление отношений отображения между семантическими фрагментами в *SCg-коде*.

Когда отношения отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами установлены, можно вычислить подобие между ответами. Ниже показан процесс вычисления подобия между семантическими фрагментами ответов на вопросы на толкование определений:

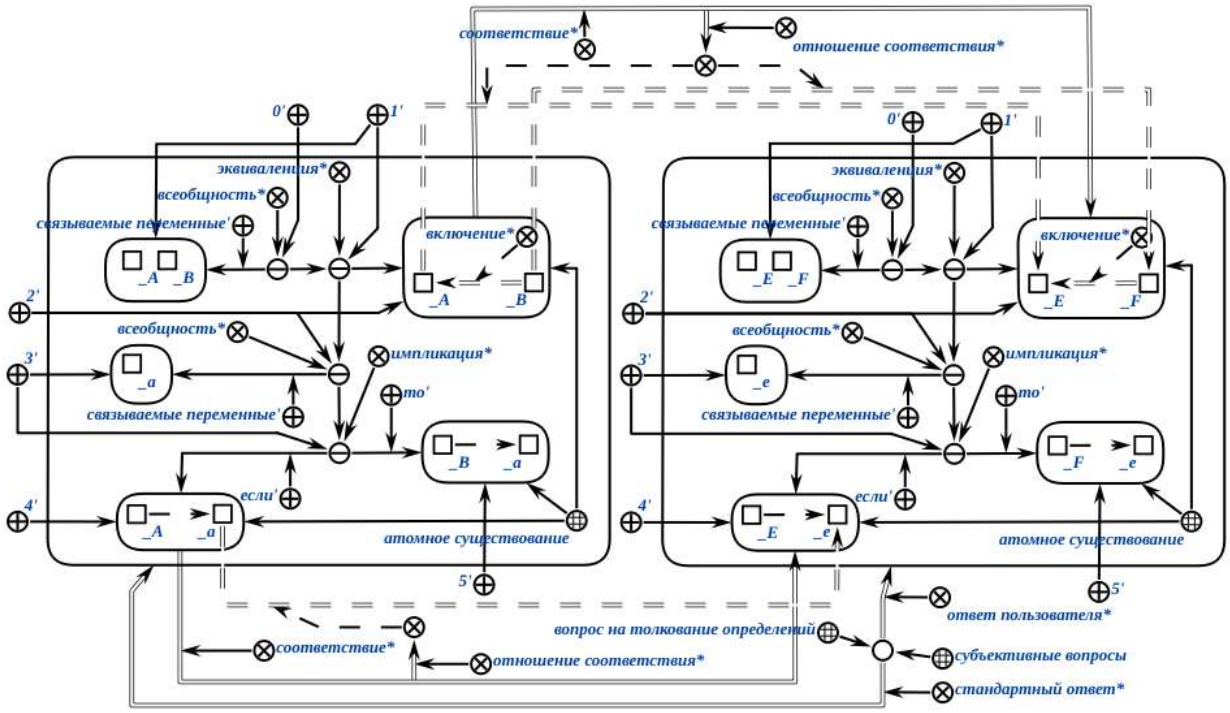
- декомпозиция семантического фрагмента стандартного ответа и семантического фрагмента ответа пользователя на подструктуры в соответствии с правилами представления фактических знаний и логических знаний;
- нумерация sc-связок и sc-структур в семантических фрагментах ответов, соответственно, и установление отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами;
- использование формул (7.5..1), (7.5..2) и (7.5..3) для вычисления точности P_{sc}, полноты R_{sc} и подобия F_{sc} между семантическими фрагментами.

Поскольку семантические фрагменты ответов на вопросы на толкование определений описываются на основе логических формул, если подобие между семантическими фрагментами не равно 1 ($F_{sc} < 1$), необходимо также определить, являются ли их логические формулы логически эквивалентными. В логике предикатов существует такая теорема: любая формула логики предикатов имеет эквивалентную ей *п.н.ф.* Поскольку логическая формула в рамках *Технологии OSTIS* расширяется на основе формулы логики предикатов, она также обладает таким свойством. Поэтому мы можем рассмотреть возможность преобразования семантических фрагментов, основанных на описаниях логических формул, в описания *п.н.ф.*, а затем определить, удовлетворяется ли между ними логическая эквивалентность (см. *Fujiwara M..PreneNFTiS-2021art; Kowalski R.PrediLaPL-1974art*). Однако *п.н.ф.* логической формулы не является уникальной, и причины, по которым п.н.ф. не является уникальной, включают:

- используемый порядок различных формул логической эквивалентности (правила преобразования). Например, преобразование $(\forall x F(x) \land \neg \exists x G(x))$ в *п.н.ф.*:
  - $\forall x F(x) \land \neg \exists x G(x)$
    $\Leftrightarrow \forall x F(x) \land \forall x \neg G(x)$
    $\Leftrightarrow \forall x (F(x) \land \neg G(x))$, (правило эквивалентности)
  - $\forall x F(x) \land \neg \exists x G(x)$
    $\Leftrightarrow \forall x F(x) \land \forall y \neg G(y)$, (правила переименования)
    $\Leftrightarrow \forall x \forall y (F(x) \land \neg G(y))$, (правила расширения области действия кванторов)
- порядок кванторов в *п.н.ф.* Например, преобразование логической формулы $(\forall x F(x) \land \exists y G(y))$ в *п.н.ф.*:
  - $\forall x F(x) \land \exists y G(y)$
    $\Leftrightarrow \forall x \exists y (F(x) \land G(y))$, (правила расширения области действия кванторов)

- $\forall x F(x) \wedge \exists y G(y)$
  $\Leftrightarrow \exists y \forall x (F(x) \wedge G(y))$, (правила расширения области действия кванторов)

SCg-текст. Пример установления отношений отображения потенциальных эквивалентных пар переменных sc-узлов между семантическими фрагментами
=



Поэтому, на основе подхода к преобразованию формул логики предикатов в *п.н.ф.* и некоторых характеристик логических формул в *ostis-системах*, в данном параграфе предлагается подход к преобразованию логических формул в уникальные (детерминированные) *п.н.ф.* в соответствии со строгими правилами ограничения. Строгие правила ограничения в основном включают следующее:

- чтобы решить проблему, заключающуюся в том, что *п.н.ф.* не являются уникальными из-за порядка использования различных формул логической эквивалентности, мы указываем, что правило переименования должно использоваться предпочтительно при преобразовании логических формул в *п.н.ф.*;
- для решения проблемы, что *п.н.ф.* не является уникальной из-за порядка кванторов, в данном параграфе предлагается подход, позволяющий перемещать все кванторы в передний конец логической формулы строго в соответствии с приоритетом кванторов. Процесс перемещения кванторов показан ниже:
  - если в начале логической формулы не существует кванторов, то все кванторы существования перемещаются в начало логической формулы по преимуществу;
  - если последний квантор в переднем конце логической формулы является квантором всеобщности, то кванторы всеобщности в логической формуле будут преимущественно перемещены в начало формулы;
  - если последний квантор в переднем конце логической формулы является квантором существования, то кванторы существования в логической формуле будут перемещены преимущественно в начало формулы.

- логическая формула, используемая для представления ответа на вопрос на толкование определений, обычно может быть выражена в следующей форме: $(Q_1 x_1 Q_2 x_2 \cdots Q_n x_n (A \leftrightarrow B))$, где $Q_i \, (i = 1, \cdots n)$ представляет собой квантор (см. *Li W..Devel oaPSfAAV-2021art; Krom M.tDecisPfFiPC-1970art*). A используется для описания определения понятия на целостном уровне, и кванторы в него не включены. B используется для объяснения семантического оттенка определения на уровне детализации, и обычно эта формула является логической формулой, содержащей кванторы (также известной как логическая подформула). Поэтому, исходя из характеристик логической формулы и для упрощения обработки знаний, необходимо лишь преобразовать логическую формулу B в *п.н.ф.*;
- для упрощения обработки знаний при преобразовании логических формул в *п.н.ф.* необходимо исключить только связку импликации;

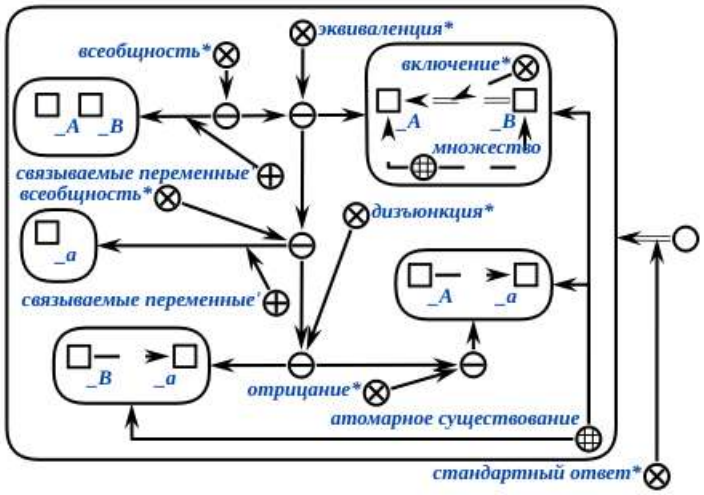
- несколько атомарных логических формул, соединенных с помощью одной и той же связки конъюнкции, предпочтительно объединяются в одно целое (то есть они объединяются в одну sc-структуру).

Процесс преобразования семантических фрагментов ответов на вопросы на толкование определений в описание п.н.ф. в соответствии со строгими правилами ограничения показан ниже:

- если в семантическом фрагменте имеется несколько sc-структур, соединенных одной и той же связкой конъюнкции, то содержащиеся в них sc-конструкции объединяются в одну sc-структуру;
- исключение всех связок импликации в семантических фрагментах;
- перемещение всех связок отрицания в семантических фрагментах в передний конец соответствующей sc-структуры;
- использование правила переименования, чтобы все связанные переменные в семантических фрагментах не были одинаковыми;
- перемещение всех кванторов в первый конец логической формулы;
- снова объединение sc-структур, которые могут быть объединены в семантическом фрагменте.

Пример преобразования логической формулы в *п.н.ф.* показан на рисунке (*SCg-текст. Пример преобразования семантического фрагмента в представление п.н.ф.*) в *SCg-коде* ($\forall A \forall B((A \subseteq B) \leftrightarrow \forall a(a \in A \rightarrow a \in B)) \Leftrightarrow \forall A \forall B((A \subseteq B) \leftrightarrow \forall a(\neg(a \in A) \vee (a \in B)))$).

SCg-текст. Пример преобразования семантического фрагмента в представление п.н.ф.
=



Следует подчеркнуть, что если вычисленное подобие между семантическими фрагментами представления *п.н.ф.* не равно 1 ($F_{sc} < 1$), то в качестве окончательного подобия ответа используется подобие между семантическими фрагментами, вычисленное в первый раз. Когда подобие между ответами получено, а затем объединено со стратегией оценки субъективных вопросов, можно проверить правильность и полноту ответов пользователей (см. *Li W..Devel oaPSfAAV-2021art*).

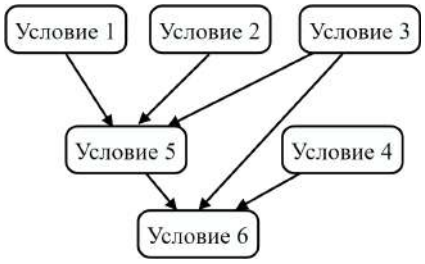
Вычисление подобия между ответами на вопросы на доказательство и на решение задачи

Как вопросы на доказательство, так и решение задачи в математике следуют общему процессу решения задач:

1. набор условий (Ω), состоящий из некоторых известных условий;
2. выведение промежуточного вывода с использованием некоторых известных условий в Ω и добавление его к Ω. Каждый элемент в Ω можно рассматривать как шаг решения;
3. повторять шаг 2 до получения окончательного результата (см. *Zhang J..AutomPoTPfTiEGItHIPT-1995art; Zhang J..Self-EAPBoPGftPoWMI-2019art*).

Этот процесс решения задачи абстрагируется в виде направленного графа, структура которого в большинстве случаев представляет собой перевернутое дерево (в особых случаях направленный граф будет содержать цикл, см. *Рисунок. Пример дерева рассуждений*), и называется деревом рассуждений (то есть деревом рассуждений стандартного ответа).

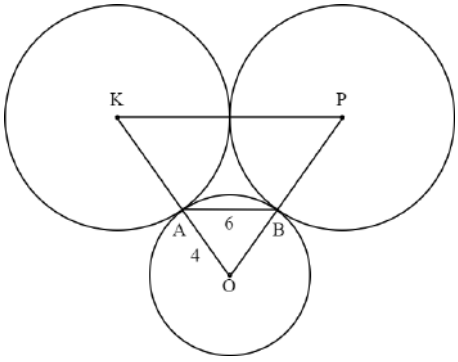
Рисунок. Пример дерева рассуждений
=



Ответ пользователя на вопрос на доказательство или на решение задачи представляет собой линейную структуру, состоящую из некоторых шагов решения (то есть известных условий, промежуточных условий или выводов), каждый из которых удовлетворяет строгим отношениям выведения и логическим отношениям, если ответ пользователя полностью правильный. Процесс автоматической проверки ответов пользователя на данный тип тестовых вопросов аналогичен традиционной ручной проверке ответов, то есть проверка того, является ли текущий шаг решения ответа пользователя правильным заключением частичного шага решения, предшествующего этому шагу. Это означает, всегда ли шаг решения в ответе пользователя, соответствующий родительскому узлу в дереве рассуждений, располагается после шагов решения в ответе пользователя, соответствующих дочерним узлам.

Семантические фрагменты ответов пользователя на вопросы на доказательство и на решение задачи в *ostis-системах* представляют собой линейные структуры, состоящие из некоторых семантических подфрагментов для описания шагов решения и некоторых семантических фрагментов для описания логического порядка и процессов преобразования между семантическими подфрагментами (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст; МетасOSTIS-2022эл*). Процесс построения и семантическая спецификация семантических фрагментов ответов пользователя на вопросы на доказательство и на решение задачи подробно описаны в работе (см. *Шункевич Д.В..МетодКПСУЗ-2013ст*). Семантические фрагменты стандартных ответов на тестовые вопросы такого типа представляют собой деревья рассуждений, состоящие из некоторых шаблонов поиска (которые могут абстрагироваться как узлы в дереве). Каждый шаблон поиска построен строго в соответствии со стандартными шагами решения соответствующего тестового вопроса (то есть в соответствии с известными условиями, промежуточными условиями и выводами в Ω). Шаблон поиска в *ostis-системах* используется для поиска в базе знаний всех соответствующих ему семантических фрагментов, и он построен на основе *Языка SCL*. Далее в качестве примера взято реальное решение задачи, чтобы представить построение его семантического фрагмента ответа пользователя (рисунок *SCg-текст. Пример семантической модели ответа пользователя на решение задачи*) и семантического фрагмента стандартного ответа (дерева рассуждений), (рисунок *SCg-текст. Пример семантической модели дерева рассуждений стандартного ответа*). Описание решения задачи: «Две равные окружности внешне касаются другой и третьей окружности, радиус которой равен 4. Отрезок, торой соединяет точки касания двух равных окружностей с третьей, равен 6. Найдите радиусы равных окружностей.», (см. *Рисунок. Пояснительный рисунок к решению задачи*).

Рисунок. Пояснительный рисунок к решению задачи
=



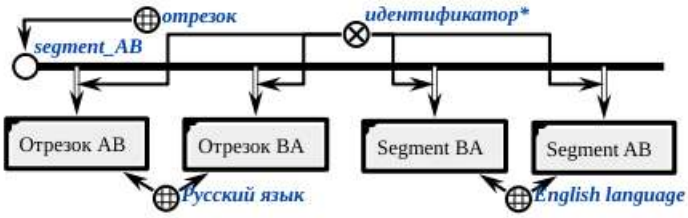
Описание ответа пользователя на естественном языке:
1. $\because KP = 2 * R$
2. $\because KO = 4 + R$
3. $\therefore \Delta AOB \backsim \Delta KOP$

4. $\therefore KA = R = 12$

Ответы пользователей на естественном языке преобразуются в семантические фрагменты с помощью естественно-языковых интерфейсов. Поэтому при вычислении подобия между семантическими фрагментами ответов нет необходимости учитывать различия понятий на уровне *естественного языка* (см. *Qian L..OntolAfCLID-2020art*). Пример спецификации конкретного понятия показан на рисунке (*SCg-текст. Пример семантической спецификации отрезка AB*).

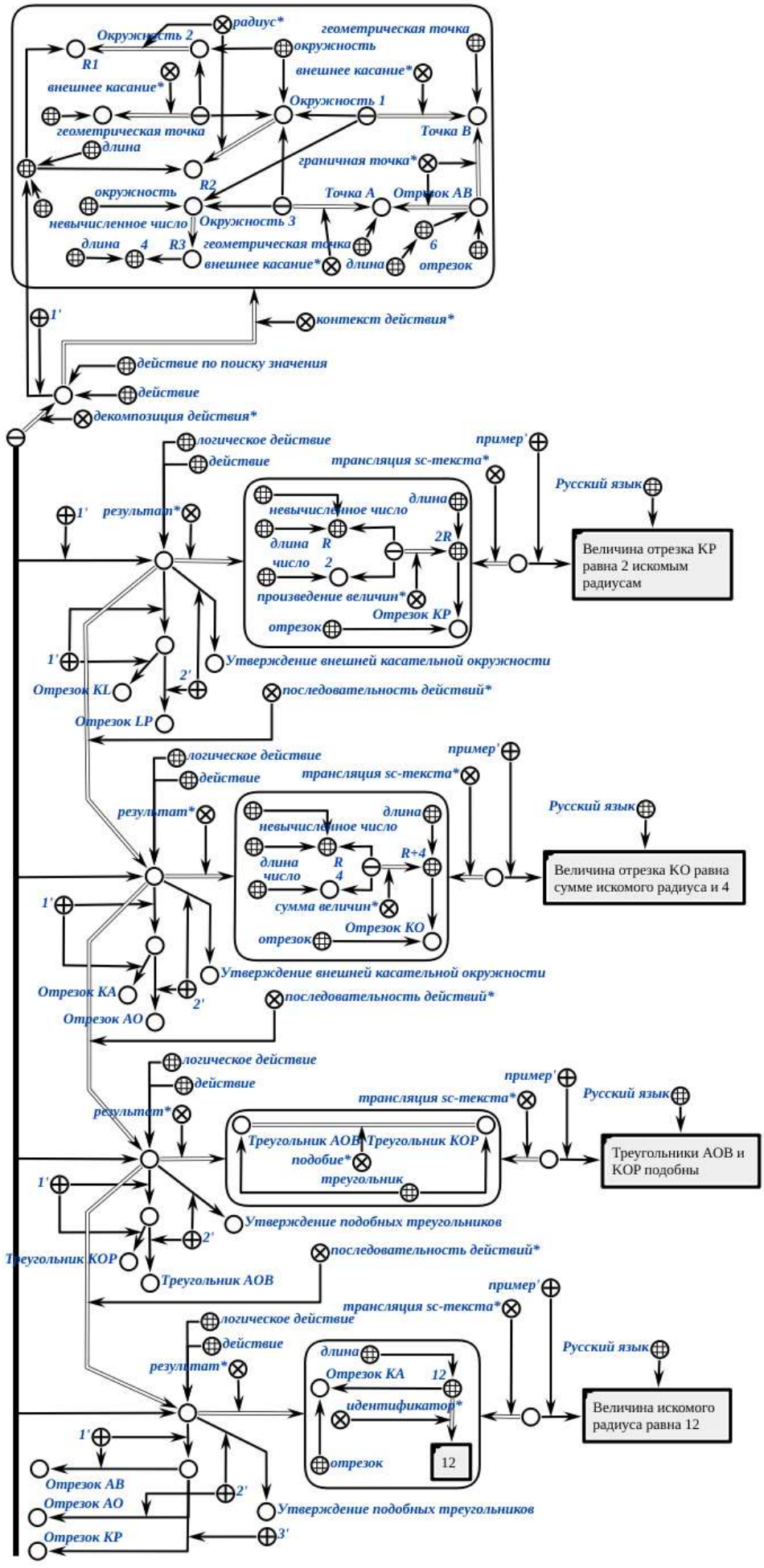
SCg-текст. Пример семантической спецификации отрезка AB

=

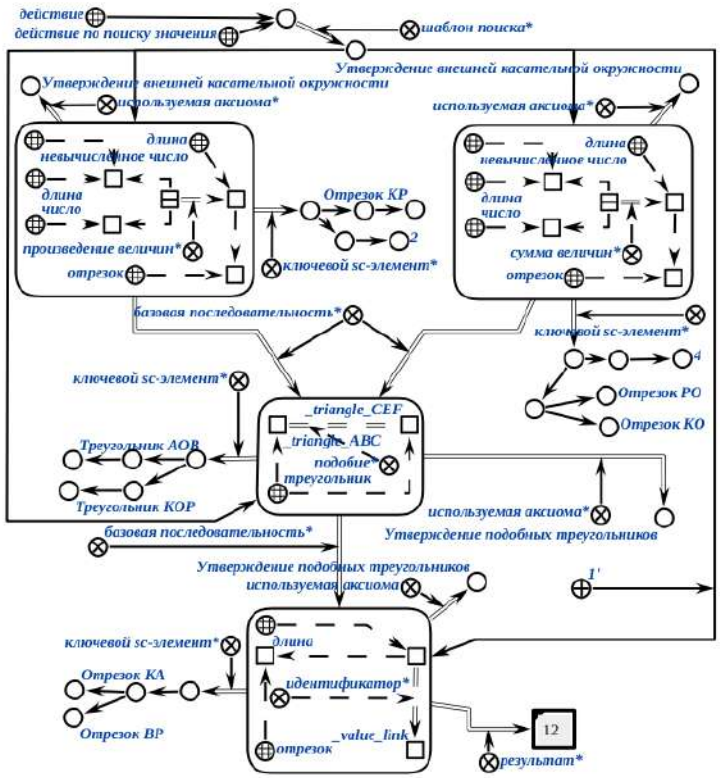


SCg-текст. Пример семантической модели ответа пользователя на решение задачи

=



SCg-текст. Пример семантической модели дерева рассуждений стандартного ответа
=



Из рисунка (*SCg-текст. Пример семантической спецификации отрезка AB*) видно, что *Отрезок AB* и *Отрезок BA* представлены одним и тем же sc-узлом, они просто являются двумя идентификаторами sc-узла. Поэтому на основе ранее представленного принципа автоматической проверки ответов пользователя на вопросы на доказательство и на решение задачи и семантических моделей ответов в *ostis-системах*, в данном параграфе предлагается подход к вычислению подобия между семантическими фрагментами ответов на вопросы на доказательство и на решение задачи в соответствии с деревом рассуждений стандартного ответа (семантический фрагмент стандартного ответа). Процесс вычисления подобия между семантическими фрагментами показан ниже:

1. нумерация каждого семантического подфрагмента (шага решения) в семантическом фрагменте ответов пользователей (порядок нумерации начинается с 1);
2. каждый узел (шаблон поиска) в дереве рассуждений обходится по очереди в соответствии со стратегией DFS. В то же время, соответствующий семантический подфрагмент, включенный в семантический фрагмент ответа пользователя, ищется в базе знаний с использованием шаблона поиска, который обходится в данный момент. Если такой семантический подфрагмент существует, то определить, меньше ли нумерация найденного семантического подфрагмента, чем нумерация семантического подфрагмента, соответствующего шаблону поиска родительского узла текущего шаблона поиска (кроме корневого узла дерева рассуждений), и если да, то найденный семантический подфрагмент считается правильным;
3. повторять шаг 2, пока не будут обойдены все шаблоны поиска в дереве рассуждений и одновременно подсчитано количество правильных семантических подфрагментов;
4. использование формул (7.5..1), (7.5..2) и (7.5..3) для вычисления точности P_{sc}, полноты R_{sc} и подобия F_{sc} между ответами. Параметры в формуле переопределены:
   - $|T_{sc}(u)|$ — количество всех семантических подфрагментов в семантическом фрагменте ответа пользователя u;
   - $|T_{sc}(s)|$ — количество всех шаблонов поиска в дереве рассуждений s;
   - $|T_{sc}(u) \otimes T_{sc}(s)|$ — количество правильных семантических подфрагментов.

После получения подобия между ответами на вопросы на доказательство и на решение задачи, правильность и полнота ответов пользователей может быть проверена в сочетании со стратегией оценки субъективных вопросов.

Стратегия оценки субъективных вопросов показана ниже:

- если подобие между ответами равно 1 (F_{sc} = 1), то ответ пользователя полностью правильный и пользователь получает максимальный балл (Max_{score});
- если подобие между ответами меньше 1 (F_{sc} < 1) и точность равна 1 (P_{sc} = 1), то ответ пользователя правильный, но неполный, и оценка пользователя равна $R_{sc} * Max_{score}$;

- если подобие между ответами больше 0 и меньше 1, а точность меньше 1 ($0 < F_{sc} < 1$ и $P_{sc} < 1$), то ответ пользователя является частично правильным и оценка пользователя равна $P_{sc} * Max_{score}$;
- если подобие между ответами равно 0 ($F_{sc} = 0$), то ответ пользователя является неправильным и оценка пользователя равна 0 (см. *Li W..Devel oaPSfAAV-2021art*).

Предлагаемый подход к автоматической проверке ответов пользователей имеет следующие преимущества:
- проверка правильности и полноты ответов пользователя на основе семантики;
- можно проверить правильность и полноту ответов пользователя на любые типы тестовых вопросов и определить логическую эквивалентность между ответами;
- позволяет вычислять подобие между любыми двумя семантическими фрагментами в базе знаний;
- предложенный подход может быть использован в различных *ostis-системах*.

Пункт 7.5.3.3. Семантическая модель базы знаний подсистемы контроля знаний

База знаний подсистемы в основном используется для хранения автоматически сгенерированных тестовых вопросов различных типов, а также позволяет автоматически извлекать ряд тестовых вопросов и формировать экзаменационные билеты в соответствии с требованиями пользователя. Поэтому для повышения эффективности доступа к базе знаний подсистемы и эффективности извлечения тестовых вопросов в данном параграфе предлагается подход к построению базы знаний подсистемы в соответствии с типом тестовых вопросов и стратегией генерации тестовых вопросов. Основой базы знаний любой *ostis-системы* (точнее, sc-моделью *базы знаний*) является иерархическая система предметных областей и соответствующих им онтологий (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014ст; Шункевич Д.В..МетодКПСУЗ-2013ст; МетасOSTIS-2022эл*). Рассмотрим иерархию базы знаний подсистемы в *SCn-коде*:

Раздел. Предметная область тестовых вопросов
⇒ *декомпозиция раздела**:
{• *Раздел. Предметная область субъективных вопросов*
⇒ *декомпозиция раздела**:
{• *Раздел. Предметная область вопроса на толкование определений*
• *Раздел. Предметная область вопроса на доказательство*
• *Раздел. Предметная область решения задачи*
}
• *Раздел. Предметная область объективных вопросов*
⇒ *декомпозиция раздела**:
{• *Раздел. Предметная область вопроса на выбор*
• *Раздел. Предметная область вопроса на заполнение пробелов*
• *Раздел. Предметная область вопроса суждения*
}
}

Далее, взяв в качестве примера предметную область объектных вопросов, рассмотрим ее структурную спецификацию в *SCn-коде*:

Предметная область объективных вопросов
∈ *предметная область*
∋ *максимальный класс объектов исследования'*:
объективный вопрос
∋ *немаксимальный класс объектов исследования'*:
• *вопрос на выбор*
• *вопрос на заполнение пробелов*
• *вопрос суждения*

Объективные типы тестовых вопросов могут быть разложены на более конкретные типы в соответствии с их характеристиками и соответствующими стратегиями генерации тестовых вопросов. Далее, взяв в качестве примера вопрос на выбор, рассмотрим его семантическую спецификацию в *SCn-коде*:

вопрос на выбор
∈ *максимальный класс объектов исследования'*:
Предметная область вопроса на выбор
⇒ *разбиение**:

- {• *вопрос на выбор на основе свойств отношений*
  - *вопрос на выбор на основе идентификаторов*
  - *вопрос на выбор на основе примеров изображения*
  - *вопрос на выбор на основе аксиом*
  - *вопрос на выбор на основе элементов*
- ⇒ *разбиение**:
  - {• *вопрос на выбор на основе бинарного отношения*
    - *вопрос на выбор на основе ролевого отношения*
    }
  - *вопрос на выбор на основе классов*
- ⇒ *разбиение**:
  - {• *вопрос на выбор на основе отношения разбиения*
    - *вопрос на выбор на основе отношения строгого включения*
    - *вопрос на выбор на основе отношения включения*
    }
  }
- ⇒ *разбиение**:
  - {• *вопрос на выбор с несколькими вариантами ответа*
    - *вопрос на выбор с одним вариантом ответа*
    }
- ⇒ *разбиение**:
  - {• *выбор неправильного варианта*
    - *выбор правильного варианта*
    }

Пункт 7.5.3.4. Семантическая модель решателя задач подсистемы контроля знаний

Решатель задач любой *ostis-системы* (точнее, sc-модель решателя задач *ostis-системы*) представляет собой иерархическую систему sc-агентов обработки знаний в семантической памяти (sc-агенты), которые взаимодействуют только путем указания действий, выполняемых ими в указанной памяти (см. *Голенков В.В..ПроекОСТКПИСЧ2-2014cm*).

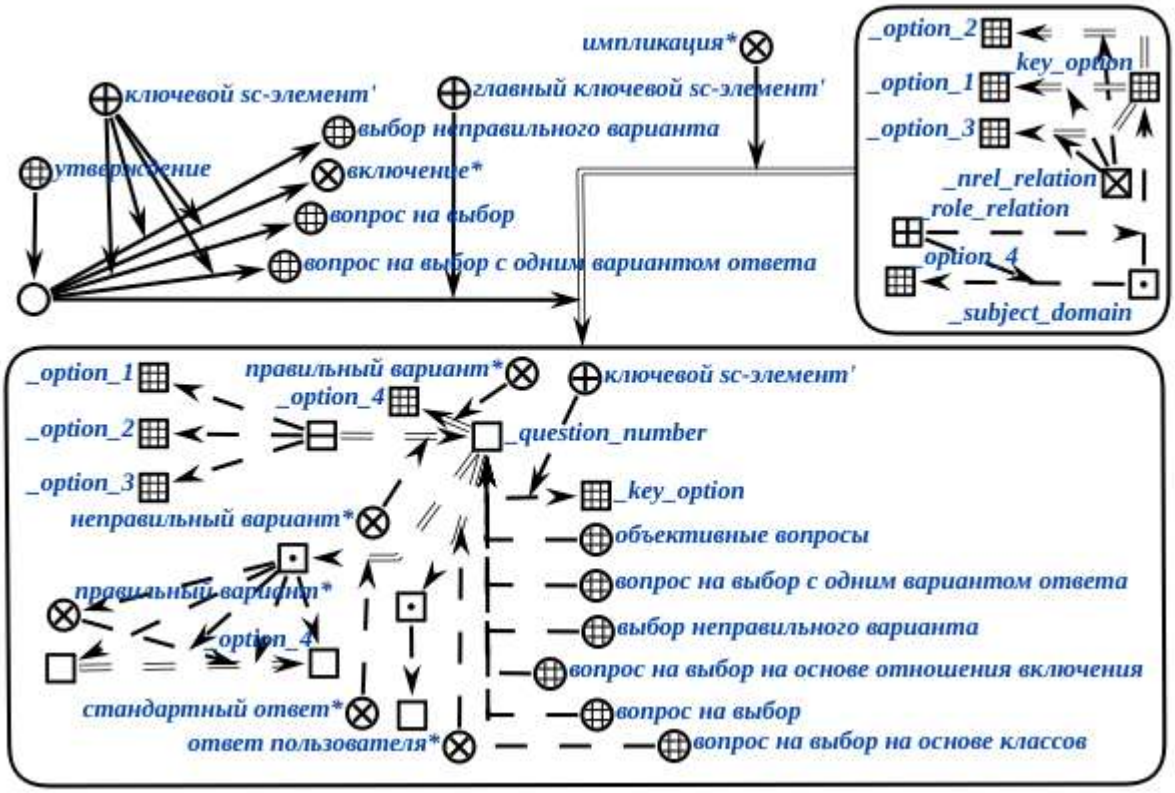
Поэтому для решения соответствующих задач в данном параграфе приведена реализация решателя задач для автоматической генерации тестовых вопросов и автоматической проверки ответов пользователей, иерархия которого представлена следующим образом в *SCn-коде*:

Решатель задач для автоматической генерации тестовых вопросов и автоматической проверки ответов пользователей

- ⇒ *декомпозиция абстрактного sc-агента**:
  - {• *Абстрактный sc-агент для автоматической генерации тестовых вопросов*
  - ⇒ *декомпозиция абстрактного sc-агента**:
    - {• *Абстрактный sc-агент для быстрой генерации тестовых вопросов и экзаменационных билетов*
      - *Абстрактный sc-агент для генерации тестовых вопросов одного типа*
      - *Абстрактный sc-агент для генерации единого экзаменационного билета*
      }
    - *Абстрактный sc-агент для автоматической проверки ответов пользователей*
  - ⇒ *декомпозиция абстрактного sc-агента**:
    - {• *Абстрактный sc-агент для автоматической оценки экзаменационных билетов*
      - *Абстрактный sc-агент для вычисления подобия между ответами на объективные вопросы*
      - *Абстрактный sc-агент для оценки логической эквивалентности между семантическими фрагментами, описанными на основе фактических знаний*
      - *Абстрактный sc-агент для вычисления подобия между ответами на вопросы на толкование определений*
      - *Абстрактный sc-агент для преобразования логической формулы в п.н.ф.*
      - *Абстрактный sc-агент для вычисления подобия между ответами на решение задачи и на вопросы на доказательство*
      }
  }

Основная функция абстрактного *sc-агента* для быстрой генерации тестовых вопросов и экзаменационных билетов заключается в автоматизации всего процесса от генерации тестовых вопросов до генерации экзаменационных билетов путем инициирования соответствующих *sc-агентов* (абстрактный *sc-агент* для генерации тестовых вопросов одного типа и абстрактный sc-агент для генерации единого экзаменационного билета). Основной функцией абстрактного sc-агента для генерации тестовых вопросов одного типа является автоматическая генерация ряда тестовых вопросов из базы знаний с использованием логических правил, построенных на основе *SC-кода* (см. *MetacOSTIS-2022эл*). Логические правила для генерации тестовых вопросов построены строго в соответствии со стратегиями генерации тестовых вопросов, описанными ранее. Пример логического правила приведен на рисунке (*SCg-текст. Пример логического правила для генерации вопроса на выбор*).

SCg-текст. Пример логического правила для генерации вопроса на выбор
=



Основная функция абстрактного *sc-агента* для автоматической оценки экзаменационных билетов заключается в реализации автоматической проверки ответов пользователей на различные типы тестовых вопросов и автоматической оценки экзаменационных билетов путем инициирования абстрактных *sc-агентов* для вычисления подобия между ответами пользователей, абстрактного *sc-агента* для оценки логической эквивалентности между семантическими фрагментами, описанными на основе фактических знаний и абстрактного *sc-агента* для преобразования логической формулы в *п.н.ф.*

§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем

⇒ *ключевое понятие**:
- *дидактическая информация*
- *учебный вопрос*
- *учебная задача*

⇒ *библиографическая ссылка**:
- *Баклавски К..ОнтолС2020-2020ст*
- *Белкин Е.Л.ДидакОУПДвУ-1982кн*
- *Гаврилова Т.А..ВизуаМРсЗП-2008ст*

- *Gavrilova T..VisuaAaaRF-2009art*
- *Голенков В.В..ИнтелОСиВУ-2001кн*
- *Гракова Н.В..МоделСОКиД-2015ст*
- *Давыденко И.Т.РазраИОСнОТ-2013ст*
- *Давыденко И.Т..СеманСБЗИ-2015ст*
- *Заливако С.С..СеманТКПИРЗ-2011ст*
- *Зимин И.В.Реком пПСиСЭОР-2016кн*
- *Колб Д.Г.WebОРСМ-2011ст*
- *Колб Д.Г.ПроблОМПнО-2013ст*
- *Ли В..ОнтолМГВдИО-2019ст*
- *Муромцев Д.И.Модел иМИЭОвК-2020ст*
- *Осин А.В.Мульт вОКИ-2004кн*
- *Русецкий К.В.ЛингвБЗИО-2013ст*
- *Соловов А.В.ЭлектОПДТ-2006кн*
- *Соловов А.В.МоделСЭОР-2007ст*
- *Соловов А.В..ДискрММвИПА..-2021ст*
- *Таранчук В.Б.Метод иТРПС-2019ст*
- *Шункевич Д.В.БазовПТКПМ-2013ст*

⇒ *подраздел**:

- *Пункт 7.5.4.1. Средства спецификации описываемых объектов*
- *Пункт 7.5.4.2. Средства логико-семантической систематизации используемых информационных конструкций*
- *Пункт 7.5.4.3. Средства указания и описания сходств, отличий, типовых экземпляров*
- *Пункт 7.5.4.4. Правила построения сущностей различных классов*
- *Пункт 7.5.4.5. Методологическая спецификация тенденций эволюции знаний*
- *Пункт 7.5.4.6. Средства представления учебных заданий и учебно-методической структуризации баз знаний ostis-систем*

Введение в § 7.5.4.

Важнейшим критерием качества создаваемых *ostis-систем* любого назначения является создание условий для того, чтобы недостаточно квалифицированные пользователи каждой *ostis-системы* (как конечные пользователи, так и ответственные за ее эффективную эксплуатацию и модернизацию) могли достаточно быстро с помощью этой же *ostis-системы* приобрести требуемую квалификацию. Это означает, что каждая *ostis-система* независимо от ее прямого назначения (автоматизации конкретных видов человеческой деятельности в конкретной области) должна быть также *обучающей системой*, то есть должна уметь обучать своих пользователей в направлении повышения их квалификации. Квалифицированный пользователь любой категории должен понимать возможности *ostis-системы*, с которой он взаимодействует, должен понимать то, что знает и что умеет система, а также то, как можно управлять ее деятельностью. Отсутствие взаимопонимания между *ostis-системами* и их пользователями — это нарушение требования *интероперабельности*, предъявляемого как к *ostis-системам*, так и к их пользователям.

К сожалению, современные традиции представления различного вида документации, стандартов, отчетов, научно-технических статей и монографий не только не ориентированы на их адекватное понимание *интеллектуальными компьютерными системами*, но также и не способствуют быстрому их пониманию со стороны тех людей, которым эти тексты адресованы. Последнее обстоятельство требует разработки (написания) учебников и учебных пособий, специально предназначенных для тех людей, которые начинают усваивать соответствующую область знаний, которые еще не приобрели необходимую им квалификацию в этой области. Но очевидно, что это предполагает существенное дублирование представляемой информации.

В идеале *база знаний* каждой *ostis-системы* должна содержать достаточно полную собственную документацию (руководство конечного пользователя, руководство по эксплуатации, руководство по модернизации). Кроме этого, *ostis-система* должна уметь:

- отвечать на самые разные (в том числе, глупые) вопросы о себе;
- анализировать действия своих пользователей и давать им полезные советы, помогающие им быстрее приобрести требуемую квалификацию.

Для того, чтобы *база знаний ostis-системы* содержала всю информацию, помогающую ей выполнить свою обучающую функцию по отношению к своим пользователям и, соответственно, помогающую пользователям быстрее приобрести необходимую пользовательскую квалификацию, *база знаний* должна содержать дополнительную информацию дидактического и, в частности, учебно-методического характера. Рассмотрим подробнее, что можно

считать такой *дидактической информацией* и каковы языковые средства ее представления в *базе знаний ostis-системы*.

дидактическая информация
:= [информация, хранимая в *базе знания ostis-системы*, предназначенная для ее пользователей и помогающая им быстрее и адекватнее усвоить денотационную семантику знаний, хранимых в памяти этой системы]
:= [*знание ostis-системы*, предназначенное для ее пользователей и помогающее им быстрее и адекватнее усвоить денотационную семантику знаний хранимых в памяти этой системы]
:= [*знание*, входящее в состав *базы знаний* и помогающее пользователю быстрее и качественнее (адекватнее) понять смысл (*денотационную семантику*) *базы знаний*]
:= [информация, способствующая более глубокому пониманию и усвоению смысла различного рода сущностей (в том числе, и различных знаний)]
:= *пояснение**:
[информация, которая при ее представлении в памяти *ostis-систем* способствует установлению взаимопонимания между ostis-системами и их пользователями, которая ускоряет процесс формирования у пользователей требуемой квалификации в различных областях]
⇒ *примечание**:
["дидактический" эффект дидактической информации обеспечивается:
- достаточной детализацией изучаемой сущности (полнотой семантической окрестности, описывающей связи этой сущности с другими сущностями)
  - декомпозицией рассматриваемых сущностей;
  - указанием аналогов (сходных сущностей в различных смыслах);
  - указанием метафор (эпиграфов);
  - указанием антиподов (сущностей, отличающихся в различных смыслах);
- упражнениями — решением различных задач с использованием изучаемых сущностей;
- ссылками на знания, хранимые в рамках той же *базы знаний*;
- ссылками на библиографические источники.

]
⇒ *примечание**:
[Типология описываемых (рассматриваемых, исследуемых, изучаемых) сущностей (объектов) ничем не ограничивается и включает в себя как конкретные материальные и абстрактные объекты и процессы, конкретные связи и структуры, конкретные информационные конструкции и, в частности, различного вида знания, так и различные классы указанных объектов]

Пункт 7.5.4.1. Средства спецификации описываемых объектов

⇒ *библиографическая ссылка**:
§ 2.5.3. Формализация понятия семантической окрестности

Для представления *дидактической информации* данного вида используются рассматриваемые ниже понятия:

спецификация
:= *часто используемый sc-идентификатор**:
[*семантическая окрестность*]
⊃ *однозначная спецификация*
:= [спецификация, которая однозначно соответствует специфицируемой сущности]
⊃ *высказывание необходимости и достаточности*
:= [высказывание о необходимых и достаточных условиях принадлежности сущностей соответствующему специфицируемому *множеству*]
⊃ *определение*
⇒ *примечание**:
[Большинство определений являются высказываниями о необходимости и достаточности]
⊃ *формальное определение*
:= [определение, представленное на формальном языке (в частности, в SC-коде)]
⊃ *строгое естественно-языковое определение*
:= [естественно-языковое определение, которое достаточно легко и адекватно перевести на формальный язык (в том числе, на SC-код)]
⊃ *нестрогое естественно-языковое определение*
⊃ *пояснение*

:= [нестрогая спецификация, которая например, используется для спецификаций неопределяемых понятий и обычно представляется на естественном языке]

⊃ *представление множества*

:= [перечисление всех или многих (желательно разнообразных) элементов специфицируемого множества]

⊃ *представление иерархии множеств*

⊃ *представление разбиения*

⊃ *представление иерархии разбиений*

:= [представление иерархического разбиения специфицируемого множества по различным признакам разбиения]

:= [текст разбиения множества]

⊃ *классификация*

:= [представление иерархической классификации специфицируемого класса по различным признакам]

:= [текст классификации]

⊃ *представление декомпозиции*

⊃ *представление пространственной декомпозиции*

⊃ *представление темпоральной декомпозиции*

⊃ *представление иерархии декомпозиции*

:= [представление иерархической декомпозиции специфицируемого объекта на компоненты (части) по различным критериям]

:= [иерархическое описание структуры специфицируемого объекта]

⊃ *представление логической формулы*

:= [полное представление логической формулы до уровня входящих в нее атомарных логических формул]

⊃ *логическая структура понятия*

⊃ *иерархия логической структуры понятия*

:= [логическая структура специфицируемого понятия, соответствующая его определению и иерархии определений всех используемых понятий (вплоть до неопределяемых понятий)]

⊃ *иерархическая структура доказательства*

:= [иерархическая структура доказательства специфицируемого высказывания (теоремы), в каждом шаге которого указывается (1) используемое правило вывода, (2) высказывания, на основе которых осуществляется шаг логического вывода, (3) высказывание, являющееся результатом (следствием) этого шага вывода]

Перечисленные виды спецификаций (1) широко используются в "дидактических" целях, (2) соответствуют различным классам специфицируемых сущностей (любым множествам, понятиям, высказываниям, сущностям, которые множествами не являются и другим классам), (3) предполагают использование целого ряда других понятий — прежде всего, *отношений*, связывающих *спецификации* со специфицируемыми сущностями. К числу таких дополнительно используемых понятий относятся следующие понятия.

однозначная спецификация*

:= [быть однозначной спецификацией заданной сущности*]

высказывание о необходимости и достаточности*

:= [быть высказыванием о необходимости и достаточности принадлежности заданному множеству*]

определение*

:= [быть определением заданного класса*]

:= [быть определением заданного понятия*]

пояснение*

:= [быть пояснением заданной сущности*]

принадлежность*

:= [быть элементом заданного множества*]

:= [элемент*]

подмножество*

:= [*включение**]

разбиение*

:= [разбиение множества*]

покрытие*

:= [покрытие множества*]

представление множества*

представление иерархии множеств*

представление разбиения*

представление иерархии разбиения*
:= [быть представлением разбиения заданного множества*]
⊃ *классификация**

часть*
⊃ *пространственная часть**
⊃ *темпоральная часть**

декомпозиция*
⊃ *пространственная декомпозиция**
⊃ *темпоральная декомпозиция**

представление декомпозиции*
:= [быть представлением декомпозиции заданной сущности*]

представление иерархии декомпозиций*

определение*
:= [быть определением заданного понятия*]

логическая структура понятия*
:= [быть логической структурой заданного понятия*]
:= [быть семейством понятий, используемых в определении заданного понятия*]

иерархия логической структуры понятия*

шаг логического вывода*
:= [быть семейством высказываний, из которых логически следует заданное высказывание*]

иерархическая структура доказательства*
:= [быть иерархической структурой доказательства заданного высказывания*]

Подчеркнем, что структуризация и систематизация как самих описываемых сущностей (объектов, не являющихся множествами, понятий различного вида, определений, высказываний, не являющихся определениями — аксиом и теорем), так и знаний, входящих в состав *баз знаний ostis-систем*, является важнейшим видом *дидактической информации*. Ключевым видом структуризации *баз знаний ostis-систем* является их онтологическая стратификация путем выделения различных *предметных областей* и соответствующих им *онтологий*, специфицирующих эти *предметные области*. Подробнее об этом см. в *§ 2.5.4. Формализация понятия предметной области*, а также в *§ 2.5.5. Формализация понятия онтологии*.

Важным видом *дидактической информации* являются *спецификации* не только таких видов *знаний*, как *высказывания* и *предметные области*, но также и различных используемых *библиографических источников* и *информационных ресурсов* (в том числе и различных разделов этих источников и информационных ресурсов). Для представления таких спецификаций в *базах знаний ostis-систем* используются следующие понятия.

официальный библиографический идентификатор*
:= [библиографическая запись, соответствующая заданному информационному ресурсу*]

автор*
:= [быть одним из авторов данного знания*]

редактор*

подраздел*

цитата*

библиографическая ссылка*
:= [семантически близкий библиографический источник*]

рассматриваемый вопрос*

аннотация*

эпиграф*

ключевой знак*
⊃ *ключевая сущность, не являющаяся понятием**

⊃ *ключевое понятие**
- ⊃ *ключевое понятие, не являющееся ни отношением, ни параметром**
- ⊃ *ключевое отношение**
- ⊃ *ключевой параметр**

⊃ ***ключевое знание****
- ≔ [основной тезис*]
- ⊃ *основное положение**
  - ≔ [основной вывод (результат)*]

Пункт 7.5.4.2. Средства логико-семантической систематизации используемых информационных конструкций

В состав данных средств входят:
- Средства явного описания синонимии и омонимии идентификаторов (имен, терминов), соответствующих рассматриваемым объектам, которые описаны в *§ 2.3.1. Внешние идентификаторы sc-элементов — знаков, входящих в конструкции SC-кода*;
- Средства явного описания семантической эквивалентности, семантического включения, семантической смежности и других семантических связей между используемыми *информационными конструкциями*. При этом описываемыми *информационными конструкциями* могут быть как *sc-конструкции*, хранимые в составе базы знаний *ostis-системы*, так и различные внешние для *sc-памяти* ostis-системы информационные конструкции (различные файлы, различные "неоцифрованные" документы);
- Средства систематизации информационных ресурсов, представленных в дидактически эффективных (наглядных, мультимедийных) формах в виде графиков, таблиц, диаграмм, фотографий, рисунков, 3D–изображений, аудио-записей, видео-записей лекций, семинаров, конференций, бесед, средств виртуальной реальности.

Пункт 7.5.4.3. Средства указания и описания сходств, отличий, типовых экземпляров

⇒ *эпиграф**:
- [Все познается в сравнении]
- [Важнейшим проявлением интеллекта является умение "видеть" сходства в различных объектах и различия в сходных]

Для представления *дидактической информации* данного вида используются следующие понятия:

аналог*
≔ [быть аналогичной сущностью*]
≔ [быть похожей сущностью*]
≔ [быть аналогом*]
≔ [быть аналогом заданной сущности*]
≔ [бинарное неориентированное отношение, каждая пара которого связывает знаки двух аналогичных (в том или ином смысле) сущностей*]
≔ [пара аналогичных сущностей*]

аналоги*
≔ [множество аналогичных сущностей*]
⊃ *аналог**
  ≔ [быть аналогом заданной сущности*]

близкий аналог*
≔ [сущность-близнец*]
≔ [быть очень похожей сущностью*]

следует отличать*
≔ [быть семейством похожих, но отличающихся сущностей, которые не следует путать*]
⊂ *аналоги**

аналогичность*
∈ *параметр*
⇒ *примечание**:
[степень аналогичности может быть разной]

:= [степень аналогичности*]
:= [Параметр, заданный на множестве пар Отношения "быть аналогом"*]
⇔ *следует отличать**:
- *сходство**
  := [описание того, в чем конкретно заключается аналогичность (сходства)]
- *следует отличать**
  := [указание факта наличие отличий между перечисленными сущностями*]

антипод*
:= [пара принципиально отличающихся сущностей*]

сходство*
:= [сходство сущностей, принадлежащих заданному семейству*]
:= [уточнение того, в чем заключается аналогичность заданного семейства сущностей*]
:= [аналогия*]

отличие*
:= [описание отличий двух заданных сущностей*]

сравнение*
:= [текст, описывающий сходства и отличия сущностей, принадлежащих заданному семейству]

сравнительный анализ*
:= [сравнительный анализ заданной сущности*]

пример′
⊂ *(включение* ⋃ принадлежность*)*

типичный экземпляр′
:= [экземпляр (элемент) заданного класса, аналогичный большинству экземпляров этого класса ′]
:= [типичный представитель заданного класса′]
:= [типичный пример′]
⊂ *пример′*

примечание*
:= [неформальное описание отличительной особенности заданной сущности*]

афоризм
:= [краткое высказывание отражающее существующего свойства описываемых объектов]

метафора
:= [иносказание, указывающее аналогию некоторых объектов]

эпиграф*
:= [афоризм, соответствующий заданной сущности (чаще всего, тексту)*]

В качестве примера приведем указание факта аналогичности групп понятий, используемых для представления дидактической информации вида структуризации и систематизации описываемых объектов (см. *Пункт 7.5.4.1. Средства спецификации описываемых объектов*).

аналоги*
∋ {•
- ⟨•
  - *представление множества**
  - *представление множества*
  - *принадлежность′*
  - *множество*
  ⟩
- ⟨•
  - *представление разбиения**
  - *представление разбиения*
  - *разбиение**
  - *множество*
  ⟩
- ⟨•
  - *классификация**
  - *классификация*
    := [представление классификации]
    := [текст классификации]
  - *разбиение**
  - *класс*

⊂ *множество*

⟩

- ⟨• *представление декомпозиции**
  - *представление декомпозиции*
  - *декомпозиция**
  - *сущность*

  ⟩
- ⟨• *представление логической формулы**
  - *представление логической формулы*
  - *принадлежность′*
  - *логическая формула*

    ⊂ *множество*

  ⟩
- ⟨• *иерархическая структура определения**
  - *иерархическая структура определения*
  - *определяющее понятие**

    ≔ [понятие, используемое в определении заданного понятия*]
  - *определение*

  ⟩
- ⟨• *иерархическая структура доказательства**
  - *иерархическая структура доказательства*
  - *посылка шага вывода**
  - *шаг вывода*

  ⟩

}

Пункт 7.5.4.4. Правила построения сущностей различных классов

Для представления *дидактической информации* данного вида используются следующие понятия:

правила построения*

≔ [принципы, лежащие в основе заданного продукта некоторой деятельности или продуктов, принадлежащих заданному классу*]

≔ [требования, предъявляемые к заданному продукту или к продуктам заданного класса*]

≔ [свойства и характеристики правильно построенного (хорошо построенного) заданного продукта или правильно построенных продуктов заданного класса*]

≔ [быть правилом построения сущностей заданного класса*]

⊃ *правила оформления**

≔ [правила придания окончательной формы (вида), правила "упаковки" заданного создаваемого продукта или создаваемых продуктов заданного класса*]

⇒ *пояснение**:

[С формальной точки зрения правила построения сущностей заданного класса — это ядро (аксиоматическая система) *логической онтологии*, которая специфицирует *предметную область*, классом объектов исследования которой является указанный выше (заданный) класс сущностей]

⇒ *примечание**:

[О правилах построения можно говорить по отношению к таким классам сущностей, как:

- различного вида *sc-знания* (см. *Предметная область и онтология знаний и баз знаний*);
- различного вида *файлы*, например, *ея-файлы ostis-систем* (см. *§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*);
- *sc-идентификаторы* различных классов *sc-элементов*.

]

⊃ *правила идентификации**

≔ [правила построения *sc-идентификаторов* для заданного класса *sc-элементов*]

⇐ *ключевое отношение**:

§ 2.3.1. Внешние идентификаторы sc-элементов — знаков, входящих в конструкции SC-кода

⊃ *правила спецификации**

≔ [правила построения *sc-спецификаций* для заданного класса *sc-элементов*]

⇐ *ключевое отношение**:

§ 2.5.3. Формализация понятия семантической окрестности

⇒ *примечание**:
[Специфицируемыми *sc-элементами* и, соответственно, обозначаемыми ими сущностями могут быть *персоны*, *библиографические источники*, разделы и сегменты *баз знаний*, *файлы* ostis-систем и многое другое]

Пункт 7.5.4.5. Методологическая спецификация тенденций эволюции знаний

Для представления *дидактической информации* данного вида используются следующие понятия:

результат анализа*
:= [результат анализа заданного объекта или класса объектов*]
⊃ *оценка качества**
:= [анализ качества заданного объекта]

принципы, лежащие в основе*
:= [принципы, лежащие в основе заданной сущности или всех сущностей, принадлежащих заданному классу*]
:= [основные свойства и характеристики заданной сущности или сущностей заданного класса*]
:= [основные положения (свойства, закономерности), присущие заданной (описываемой) сущности*]
:= [принципы, лежащие в основе заданной сущности (объекта системы, информационной конструкции) или сущностей заданного класса*]
:= [ключевые особенности*]

достоинства*
:= [преимущества*]

недостатки*

назначение*
:= [предъявляемые требования*]
:= [требования, которым должны удовлетворять сущности заданного класса]
:= [требования, предъявляемые к заданным сущностям*]

проблемы*
:= [проблемы заданного объекта или класса объектов*]
:= [в чем заключается проблема*]
:= [проблема, ассоциируемая с заданной сущностью*]

актуальные задачи*
:= [актуальные задачи совершенствования заданного объекта или класса объектов*]

известные варианты решения заданных проблем*

предлагаемый подход к решению заданных проблем*
:= [принципы, лежащие в основе предлагаемого подхода к решению заданных проблем*]

новизна предлагаемого подхода*
:= *пояснение**:
[бинарное ориентированное отношение, каждая пара которого связывает некоторую сущность с описанием того, чем она принципиально отличается от предшествующих ей аналогов. Такой сущностью может быть новая техническая система, новый метод решения некоторого класса задач и другое]
⊃ *научная новизна предлагаемого подхода**
:= ["изюминка" предполагаемой системы, метода, принципов, подхода к решению]

реализация предлагаемого подхода*

оценка качества реализации предлагаемого подхода*

направления дальнейшего развития*
:= [направления дальнейшего развития заданного объекта*]
⇒ *разбиение**:
{• *перманентные направления развития**
• *тактические направления развития текущего состояния*
:= [план ближайших работ по развитию данного объекта*]
• *стратегические направления развития текущего состояния**
:= [перспективный план развития данного объекта*]

}

Пункт 7.5.4.6. Средства представления учебных заданий и учебно-методической структуризации баз знаний ostis-систем

Для представления *дидактической информации* данного вида используются следующие понятия:

учебный вопрос
:= **[**вопрос для самопроверки или проверки качества усвоения знаний**]**
⊃ *аттестационный вопрос*

учебный вопрос*
:= **[**учебный вопрос для заданного раздела (контекста) учебного материала*]**
⊃ *аттестационный вопрос**

учебная задача
:= **[**упражнение**]**
⇒ *разбиение**:
{• *учебная задача не требующая дополнительных средств для ее выполнения*
• *учебная лабораторная работа*
}
⊃ ***аттестационная задача***

учебная задача*
:= **[**бинарное ориентированное отношение, любая пара которого связывает предметную область и онтологию с соответствующими упражнениями**]**
⊃ ***аттестационная задача****
⇒ *примечание**:
[Необходимо разбиение множества упражнений по уровню сложности, по тематике, по используемым методам, а также задание порядка упражнений.**]**

аттестационный вопрос
:= **[**тестовый вопрос контроля знаний**]**
:= **[**вопрос, который рекомендуется задавать при проверке усвоения знаний**]**

аттестационный вопрос*
:= **[**быть аттестационным вопросом для заданного контекста (например, для заданной предметной области и онтологии)**]**

Учебно-методическая структуризация *баз знаний ostis-систем* требует не только установления соответствия *учебных вопросов* и *учебных задач* с соответствующими разделами *баз знаний*, но также и включения в состав *баз знаний* рекомендаций о последовательности изучения разделов учебного материала и о последовательности решения *учебных задач*.

Глава 7.6.

Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов

⇒ *автор**:
- *Андрушевич А. А.*
- *Войтешенко И. С.*

⇒ *аннотация**:
[В данной главе описан оригинальный подход к построению межотраслевой экосистемы интернета вещей и основывающихся на ней приложениях умного дома через семантическое представление экосистемы на базе *Технологии OSTIS*. Полученные результаты в будущем позволят повысить эффективность компонентного подхода к разработке приложений в интернете вещей (на примере умного дома), а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая их совместимость и согласованность.]

⇒ *подраздел**:
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*
- *§ 7.6.2. Предлагаемый подход к созданию умных домов*
- *§ 7.6.3. Многокомпонентная модель умных домов*
- *§ 7.6.4. Подсистемы умного дома*
- *§ 7.6.5. Элементы технической реализации умных домов*

⇒ *ключевой знак**:
- *REST API*
- *MQTT*
- *Node-RED*
- *Yandex IoT Core*
- *Yandex Cloud*

⇒ *ключевое понятие**:
- *интернет вещей*
- *веб вещей*
- *умный дом*
- *контекстно-зависимая система*
- *ambient assisted living*
- *приложение умного дома*

⇒ *библиографическая ссылка**:
- *Andrushevich A..TowarSBGDA-2010art*
- *Paola A..ConteAfMSDF-2016art*
- *Kamilaris A..GeospAatIoT-2018art*
- *Dobrescu R..ConteACaMSwI-2019art*
- *World-2018el*
- *Almusaylim Z..aRevie oSHPSaC-2019art*
- *Rosen A.MalveTAaaNT-2021art*
- *Chakraborty A..SmartHSaCR-2023art*
- *Sapasgozar S..aSysteCRoAI-2020art*
- *Гузаревич Я.В..ВнедрСУДнПМ-2023эл*
- *Казарновский В.А..РазвиПкВСУД-2019ст*
- *Голенков В.В..СтандОТОП-2021кн*
- *Stojkoska B.R..aRevie oIoTfSH-2017art*
- *Andrushevich A..ZigBe814T-2009art*
- *Biallas M..LivinSaAiaAtHF-2017art*
- *Zhou B..SmartHEMSCCaSS-2016art*
- *Beaudin M..HomeEMSaRoM-2015art*
- *ПрогрУОпДДиО-2022эл*

Введение в Главу 7.6.

Многоагентная и ситуационная (контекстная) обработка нашла широкое применение в *приложениях интернета вещей*, например в *умном доме* (см. *Andrushevich A..TowarSBGDA-2010art*). Однако, несмотря на значительный прогресс последних лет в развитии *отраслевых коммуникационных систем* (*о.к.с.*) и *автоматизированных систем управления* (*а.с.у.*), по-прежнему актуальными остаются следующие проблемы:

- разнородность стандартов и технологий получения, хранения, обработки, передачи данных в приложениях *интернета вещей*;
- ограниченная отраслью функциональность *о.к.с.*;
- низкая адаптивность, совместимость и масштабируемость приложений.

Таким образом, приходится констатировать отсутствие единого комплексного подхода к проектированию экосистемы межотраслевого универсального *интернета вещей*.

Вышеуказанные проблемы относятся и к проектированию систем *умных домов*.

Применение к проектированию систем умных домов технологий семантического проектирования позволит устанавливать семантическую совместимость (взаимопонимание) как компьютерных систем, входящих в интеллектуальную структуру умного дома, так и с внешними по отношению к умному дому компьютерными системами электро- и теплоснабжения, водопотребления и водоотведения, управления транспортной инфраструктурой, умного города и других; поддерживать эту семантическую совместимость в процессе собственной и их эволюции, координировать свою деятельность с пользователями и другими компьютерными системами при коллективном решении задач (см. *Пункт 7.1.1.7. Текущее состояние и современные проблемы прикладной инженерной деятельности в области Искусственного интеллекта*).

Для поддержки *жизненного цикла компьютерных систем умного дома*, спроектированного с использованием *Технологии OSTIS*, предполагается использовать *Метасистему OSTIS* (см. *Пункт 7.1.1.11. Комплексная автоматизация человеческой деятельности в области Искусственного интеллекта с помощью интеллектуальных компьютерных систем нового поколения*).

Применение указанных подходов обеспечит включение *самостоятельных ostis-систем умного дома* в *коллективы самостоятельных ostis-систем*, то есть, в *Экосистему OSTIS* (см. *Пункт 7.3.1.1. Поддержка совместимости между ostis-системами, входящими в состав Экосистемы OSTIS*).

§ 7.6.1. Анализ существующих подходов к созданию умных домов

Концепция повсеместного межотраслевого контекстно-зависимого безопасного интернета вещей быстро набирает популярность благодаря активно развивающейся *конвергенции* пакетов технологий, системных функций, платформ и услуг. Яркими примерами механизмов такой *конвергенции* являются становление и развитие таких общих системных функций в *интернете вещей*, как базовые службы и сервисы контекстной осведомленности, геолокации и информационной безопасности. Различные отрасли приложений интернета вещей также зачастую реализуют и используют схожие функциональные блоки, включая аутентификацию и авторизацию пользователей, гео-временную локализацию, обработку событий, контекстную осведомленность и многие другие. Например, научным сообществом уже опубликованы (см. *Paola A..ConteAfMSDF-2016art*, *Kamilaris A..GeospAatIoT-2018art*, *Dobrescu R..ConteACaMSwI-2019art*), результаты разработки базовых технологий и алгоритмов для получения и предоставления информации о гео-временном местоположении и контексте. На самом деле, видение повсеместного всепроникающего интернета вещей включает в себя не только множество технологий, но и динамические изменения окружающей среды и виртуальное информационное окружение (пространство) пользователей. Сбор и использование контекстной информации выходит за рамки обычных приложений с замкнутым циклом, которые используют только простую информацию о местоположении. Например, универсальная геовременная информация успешно модернизирует приложения интернета вещей в таких отраслях, как дом, офис, промышленность, торговля, транспорт, здравоохранение, города.

умный дом - важная и быстро развивающаяся область применения *интернета вещей*. Всемирный экономический форум предполагает, что стоимость этой отрасли к 2030 году достигнет 13 триллионов долларов США (см. *World-2018el*). *умный дом* - это взаимосвязанный дом, где все типы вещей взаимодействуют друг с другом через *интернет*. Однако, в то же время, это вызывает большую озабоченность в отношении конфиденциальности и безопасности пользователей из-за возможности злонамереннного удаленного вмешательства в управление домом (см. *Almusaylim Z..aRevie oSHPSaC-2019art*). Так, например, сообщается о недавнем обнаружении кибератаки на домашние подключенные через интернет устройства с использованием вредоносной рекламы (см. *Rosen A.MalveTAaaNT-2021art*). Таким образом, быстрое развитие технологий умного дома порождает новые проблемы, такие как обеспечение пользователям безопасных и надежных услуг, сохранение конфиденциальности, предотвращение несанкциони-

рованного вмешательства в управление устройствами *умного дома* (см. *Almusaylim Z..aRevie oSHPSaC-2019art*).

При проектировании и разработке *умных домов* применяются разнообразные технологические подходы. Так, авторы обзора (см. *Chakraborty A..SmartHSaCR-2023art*) на основании анализа большого количества исследований констатируют использование беспроводных сенсорных сетей; мультиагентных систем; нейронных сетей; Машинного обучения; нечеткой логики; глобальных систем мобильной связи; Bluetooth и другие.

Проектирование *умных домов* становится важной сферой применения *Искусственного интеллекта* (см. *Sapasgozar S..aSysteCRoAI-2020art*). Практические разработки по внедрению проектных решений по умному дому ведутся в Республике Беларусь (см. *Гузаревич Я.В..ВнедрСУДнПМ-2023эл*), в России (см. *Казарновский В.А..РазвиПкВСУД-2019ст*) и в странах дальнего зарубежья.

§ 7.6.2. Предлагаемый подход к созданию умных домов

В рамках данной главы предлагается взять за основу общеизвестные компоненты *Библиотеки OSTIS* (см. *Голенков В.В..СтандОТОП-2021кн*) для формализации описания предметной области экосистемы *интернета вещей* и основанного на ней приложения *умного дома*. Компоненты *Экосистемы OSTIS* могут разрабатываться не только сверху-вниз, но и снизу-вверх, включая в экосистему наиболее подготовленные к семантическому представлению системы. Таким образом может быть достигнута лучшая конвергенция разнородных стандартов и технологий *интернета вещей*, которая поспособствует межотраслевой интеграции *о.к.с.* и *а.с.у.* в единую экосистему.

Итак, дадим следующие определения рассматриваемой предметной области интернета вещей в *SC-коде*:

интернет вещей
:= [множество физических объектов, подключенных к интернету и обменивающихся данными. Термин *"интернет вещей"* был впервые употреблен в 1999 году Кевином Эштоном, предпринимателем и соучредителем центра Auto-ID Labs (распределенная исследовательская группа в области радиочастотной идентификации и новых сенсорных технологий) при Массачусетском технологическом институте MIT]
:= [концепция сети передачи данных между физическими объектами ("вещами"), оснащенными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой]
:= [концепция сети передачи данных между физическими объектами ("вещами"), оснащенными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой]
⇒ *пояснение**:
[*интернет вещей* — множество физических объектов, подключенных к интернету и обменивающихся данными. Термин "интернет вещей" был впервые употреблен в 1999 году Кевином Эштоном, предпринимателем и соучредителем центра Auto-ID Labs (распределенная исследовательская группа в области радиочастотной идентификации и новых сенсорных технологий) при Массачусетском технологическом институте MIT]

веб вещей
:= [глобальная сеть веб-страниц, автоматически сгенерированных и автоматически считываемых встроенными в физические объекты (парковка, тротуарная плитка, окна, двери, детская игрушка, посуда, одежда, почва и так далее) вычислительными устройствами. Для веба вещей характерна конвергенция и интеграция с технологиями *Искусственного интеллекта*]
:= [подход от W3C по использованию веб-технологий в интернете вещей с целью устранения фрагментации в стандартах разработки *интернета вещей*]
:= [глобальная сеть веб-страниц, автоматически сгенерированных и автоматически считываемых встроенными в физические объекты (парковка, тротуарная плитка, окна, двери, детская игрушка, посуда, одежда, почва и так далее) вычислительными устройствами, для которой характерна конвергенция и интеграция с технологиями *Искусственного интеллекта*]
:= [подход от W3C по использованию веб-технологий в интернете вещей с целью устранения фрагментации в стандартах разработки интернета вещей]

контекстно-зависимая информационная система
:= [информационная система, использующая понятие контекста для предоставления имеющей отношение персонифицированной динамической информации и/или услуг пользователю, где степень отношения зависит от текущей среды окружения, интересов, задач, намерений и действий пользователя]

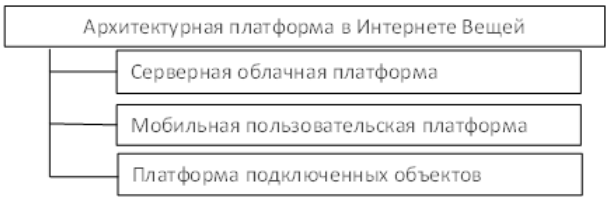
прикладной программный интерфейс с передачей репрезентативного состояния

:= [концепция построения распределенного приложения по типу клиент-сервер, в которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе (репрезентативном состоянии) сервера без необходимости хранения информации о состоянии клиента (клиентской сессии)]

На основе многолетнего профессионального опыта авторов можно утверждать, что организация универсального типового межотраслевого (или "горизонтального") способа технологической реализации системной архитектуры и приложений в интернете вещей может быть представлена в виде трех потоков разработки, каждый из которых посвящен реализации одной системной платформы интернета вещей. Эти 3 фундаментальные независимые от приложений платформы вместе образуют типовую архитектурную платформу интернета вещей, которая интегрируется во взаимосвязанную гетерогенную систему подсистем интернета вещей, предлагая разработчикам приложений общие функциональные возможности и оставляя разработчикам больше времени для концентрации на бизнес-логике их приложений.

Рисунок. Архитектура платформы IoT

=



серверная облачная платформа, обладающая высокими вычислительными ресурсами и большими объемами памяти, в наибольшей степени управляет всей системой интернета вещей. *платформа для мобильных устройств* отвечает за представление результирующей и полезной информации пользователям. *платформа подключенных объектов* объединяет сенсорную информацию об окружающей среде и способна выполнять простые действия через подключенные объекты.

Опишем указанные выше платформы подробнее:

- *Платформа Smart Server*: совокупность различных серверов для обеспечения *программного интерфейса* строительных блоков в облаке для интеграции приложений и сервисов. Данная платформа обладает высокими вычислительными возможностями, большим количеством ресурсов памяти и может взаимодействовать с любыми стандартами. Можно считать вычислительные ресурсы и возможности неограниченными. Ее роль заключается в сборе, агрегировании и/или интерпретации контекстно-зависимой информации, поступающей от подключенных узлов *приложений интернета вещей*, чтобы сделать ее эффективной и имеющей смысл (потребительскую ценность) для пользователей мобильных устройств. Она также предоставляет разработчикам приложений программных интерфейсой локальных или удаленных универсальных базовых компонент, реализующих функциональность серверной стороны всех приложений, например: обработку и аналитику больших данных, сложную обработку событий, механизм локализации, фреймворк контекстного управления, профиль пользователя и модель поведения пользователей, политики безопасности, контроль доступа (включая регистрацию пользователей, аутентификацию и схему доверия). Наиболее распространенными примерами платформы Smart Server являются Yandex.Cloud, Amazon Web Services (AWS), Microsoft Azure, Google Cloud, Cisco IoT Cloud Connect, SAP, Oracle IoT, ThingsBoard, SiteWhere, Predix IoT, Thinger.io, Ubidots;
- *Платформа Smart Mobile*: эталонная платформа для мобильных приложений, обеспечивающая функциональность на мобильных устройствах для всех приложений и.в. Мобильные устройства наиболее часто принадлежат конечным пользователям-людям. Общими функциями мобильных клиентов являются: общие компоненты пользовательских интерфейсов, профилирование пользователей, аутентификация и авторизация, информационная безопасность (конфиденциальность), поиск и обнаружение интеллектуальных серверов, связь с серверными компонентами, получение уведомлений от серверов, "локальные" алгоритмы добычи данных (толстый клиент). Данная платформа предоставляет приложениям и.в. возможность использовать необходимую им контекстную информацию, не беспокоясь о том, каким образом эта контекстная информация добывается. Мобильная платформа может взаимодействовать как с платформой Smart Server, так и непосредственно с некоторыми подключенными узлами и.в. В заключение *мобильная платформа интернета вещей* отвечает за последний этап формирования потребительской ценности через мультимедийное представление / вывод последовательной и интересующей пользователей информации. Наиболее распространенными примерами платформы Smart Mobile являются Zetta, HP Enterprise Universal, Carriots, ThingsBoard, ThingWorx, Xively. Для сокращения времени разработки мобильных приложений часто используются так называемые гибридные приложения, код которых адаптируется при помощи фреймворков сразу под несколько мобильных аппарат-

ных платформ. Таикими наиболее популярными фреймворками являются: PhoneGap, Rhodes, Appcelerator, Xamarin, Ionic, Appy Pie, Native Script.

- *Платформа Smart Object*: интеграция различных шлюзов, сетевых концентраторов и связанных с ними технологий подключенных объектов. Подключенные объекты могут воспринимать физические данные окружающей среды (температура, давление, освещенность, влажность, вибрация и так далее) через датчики или быть исполнительными механизмами (выключатель, электропривод, электромагнит и так далее). Передача данных как правило происходит через маломощные коммуникационные стандарты связи. Такие устройства также могут быть оснащены "легковесной" операционной системой. Такие устройства очень разнородны и могут работать очень по-разному. Однако, все особенности реализации подключенных объектов (вещей) должны быть скрыты от других устройств, чтобы обеспечить однородный способ общения с другими компонентами системы, независимо от того, какие данные они воспринимают или как они устроены. Следовательно, единый объектный программный интерфейс является ключевым моментом для того, чтобы иметь возможность легко интегрировать различные шлюзы, обеспечивающие доступ к различным подключенным объектам через одни и те же стандарты, не требуя от разработчиков приложения и.в. знания сложности взаимосвязи объектов со шлюзом и того, какие функциональные возможности (сбор данных или приведение в действие) выполняются на этих объектах. Данная платформа обладает общими функциями, такими как: обнаружение объектов, безопасность и управление. Наиболее распространенными примерами платформы Smart Object являются open source IoT Kaa, Arduino, Flutter, Qualcomm's IoT Development Kit, Particle.io, ESP8266, Intel Edison, Raspberry Pi, Beagle Bone.

§ 7.6.3. Многокомпонентная модель умных домов

Данный раздел посвящен теоретическому определению универсальной многокомпонентной модели типового приложения *интернета вещей* для платформ из предыдущего подраздела. Описание для "канонической" или "классической" архитектуры приложения *интернета вещей*, состоящей из облака или серверных мощностей, различных протоколов передачи данных, пользовательских устройств (персональные компьютеры, ноутбуки, планшеты, телефоны, встраиваемые пользовательские интерфейсы в любые устройства), любого рода датчики по сбору данных и любого "устройства-исполнители-руки". Учитывая особенности и свойства трех частей горизонтальной платформы (Smart Server, Smart Mobile и Smart Object), а также характеристики приложений *интернета вещей*, предлагается следующее универсальное древовидное представление многокомпонентной модели архитектуры приложения в *интернете вещей*.

(0) Приложение Интернета Вещей S = A * B * C

(1) Интеллектуальная Серверная Платформа A = D * E

(1.1) Контроль Доступа D = G * H * I

(1.1.1) Регистрация G: G1 (Пользователи), G2 (Машины)

(1.1.2) Аутентификация H: H1 (Периодическая), H2 (По Требованию)

(1.1.3) Схема доверия I: I1 (Управляемая приложениями), I2 (Управляемая политикой), I3 (Управляемая профилем), I4 (Управляемая рисками)

(1.2) Обработка Данных E = J * K * L

(1.2.1) Контекстное управление J: J1 (Управляемое событиями), J2 (Опрос)

(1.2.2) Обработка потока K: K1 (Оффлайн), K2 (Полу-онлайн), K3 (Онлайн)

(1.2.3) Хранение Данных L: L1 (Простое), L2 (Иерархическое), L3 (Структурированное), L4 (Динамическое)

(2) Интеллектуальная Мобильная Платформа B = M * Phi

(2.1) Общие Характеристики Мобильности M = O * P

(2.1.1) Основной тип траффика данных O: O1 (Мультимедиа), O2 (Восприятие/считывание), O3 (Управление), O4 (Хорошо сбалансированный)

(2.1.2) Информация о местоположении P: P1 (Режим включения/выключения), P2 (На основе точности)

(2.2) Пользовательский Интерфейс Phi = R * F

(2.2.1) Пользовательский интерфейс R: R1 (Один носитель), R2 (Мультимедиа), R3 (Адаптивный)

(2.2.2) Доставка контента F: F1 (На основе услуг), F2 (На основе устройств)

(3) Платформа Интеллектуального Объекта C = Q * T

(3.1) Общие Характеристики Узла Q = W * V * U

(3.1.1) Источник Питания W: W1 (Пассивный), W2 (Активный)

(3.1.2) Физическое взаимодействие V: V1 (Считывание), V2 (Действие), V3 (Комбинированное)

(3.1.3) Шифрование U: U1 (Да), U2 (Нет)

(3.2) Связность T = X * Y * Z

(3.2.1) Носитель X: X1 (Беспроводной), X2 (Проводной)

(3.2.2) Тип сети Y: Y1 (Сетка), Y2 (Точка-точка)

(3.2.3) Конфигурация Z: Z1 (Адаптивная), Z2 (Статическая)

Все компоненты и параметры данной универсальной древовидной иерархической модели приведены в качестве примера и могут изменяться в конкретных *приложениях интернета вещей*.

На основе предложенной древовидной многокомпонентной модели экосистемы *интернета вещей* могут быть реализованы такие *приложения интернета вещей*, как *умный дом*, *офис*, *промышленность*, *торговля*, *транспорт*, *здравоохранение*, *умные города* и другие.

В качестве примера приложения рассмотрим реализацию и адаптацию ранее предложенной универсальной древовидной модели интернета вещей для предметной области *умного дома*. Как и в универсальной модели, все компоненты и параметры древовидной иерархической модели *умного дома* приведены в качестве примера и могут изменяться в конкретных реализации *умного дома*. Умный дом S = A * B * C

(1) Подсистема обеспечения безопасности A = D * E

(1.1) Контроль доступа D = G * H * I

(1.1.1) Оконные ставни G: G1 (Ручные), G2 (С электроприводом)

(1.1.2) Дверные замки H: H1 (Стандартные), H2 (Электрические)

(1.1.3) Способ аутентификации I: I1 (Физический ключ), I2 (PIN), I3 (RFID), I4 (Биометрия)

(1.2) Управление тревожной сигнализацией E = J * K * L

(1.2.1) Сигнал тревоги J: J1 (Зуммер), J2 (Световой)

(1.2.2) Детектор присутствия K: K1 (Инфракрасный), K2 (Ультразвуковой), K3 (Датчик движения)

(1.2.3) Подключение оповещения L: L1 (Наземная линия связи), L2 (Радио), L3 (Интернет), L4 (GSM/SMS)

(2) Подсистема обеспечения комфорта B = M * N

(2.1) На основе температурного режима M = O * P

(2.1.1) Отопление O: O1 (Теплый пол), O2 (Радиаторы), O3 (Обогреваемая кровля), O4 (Термостенка)

(2.1.2) Кондиционирование воздуха P: P1 (Внешнее), P2 (Внутреннее)

(2.2) На основе качества N = R * F

(2.2.1) Вентилятор R: R1 (Потолочный), R2 (Рабочие помещения), R3 (Центральный)

(2.2.2) Воздушный фильтр F: F1 (С теплообменом), F2 (Централизованный)

(3) Интеллектуальная подсистема C = Q * T

(3.1) Мультимедиа Q = W * V * U

(3.1.1) Видеосистема W: W1 (Монитор), W2 (Проектор)

(3.1.2) Аудиосистема V: V1 (2:1), V2 (5:1), V3 (Долби)

(3.1.3) Домашний сервер / PC U: U1 (Отсоединенный), U2 (Интегрированный)

(3.2) Бытовая техника T = X * Y * Z

(3.2.1) Печь X: X1 (Газ), X2 (Электричество)

(3.2.2) Холодильник Y: Y1 (С морозильной камерой), Y2 (С веб-интерфейсом)

(3.2.3) Пылесос Z: Z1 (Центральный), Z2 (с поддержкой технологии iLoc)

Далее более подробно опишем основные типовые подсистемы и функции умного дома.

§ 7.6.4. Подсистемы умного дома

⇒ *подраздел**:
- *Пункт 7.6.4.1. Подсистема доступа в жилое помещение*
- *Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми*
- *Пункт 7.6.4.3. Подсистема для управления освещенностью жилища*
- *Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью*

Среди функциональных задач, реализуемых в виде компонентов/приложений при проектировании и разработке программно-аппаратного обеспечения *"умного дома"* (см. *Stojkoska B.R..aRevie oIoTfSH-2017art*), типичными являются:
- задача доступа к жилое помещение;
- задача наблюдения за одинокими пожилыми людьми (см. *Andrushevich A..ZigBe814T-2009art* , *Biallas M..LivinSaAiaAtHF-2017art*);
- задача управления освещенностью жилища;
- задача управления энергопотреблением и энергоэффективностью (см. *Zhou B..SmartHEMSCCaSS-2016art*).

Таким образом, функциональная классификация приложений умного дома может быть определена с использованием SC-кода следующим образом:

приложение умного дома
:= [обособленный программно-аппаратный комплекс, работающий согласно функциональным требованиям]
⇒ *разбиение**:
Типология приложений умного дома по функциональности
= {• *приложение управления физическим доступом*
⊃ *функциональность контроля и управления состоянием всех традиционных и эвакуационных входов и выходов помещения*
• *приложение наблюдения за одинокими пожилыми людьми*
⊃ *функциональность по обеспечению бытовой безопасности одиноких пожилых людей*
• *приложение управления освещенностью жилища*
⊃ *функциональность контроля и управления состоянием источников естественного и искусственного освещения*
• *приложение управления энергопотреблением и энергоэффективностью*
⊃ *функциональность контроля и управления потребления всех видов ресурсов и энергии*
}

Далее более детально опишем функциональность вышеуказанных приложений умного дома.

Пункт 7.6.4.1. Подсистема доступа в жилое помещение

Ключевой задачей такой системы является идентификация людей, которые подходят к двери помещения, и корректная обработка ее результатов: если подошедший к двери человек должен иметь доступ к помещению, дверь открывается автоматически, в противоположном же случае система предлагает поговорить с жителями дома или квартиры. При этом желательно, чтобы система умела определять, находится ли кто-то из жителей дома. Если дома никого нет, то система должна сообщить посетителю, что зайти сейчас он не может.

С точки зрения устройств система оснащена камерой, а также лампами для освещения. Камера включается по датчику движения, причем также включается и освещение, если окружающая среда слишком темная для работы камеры. Система пытается распознать посетителя по снимкам с камеры.

В дополнение к данным камеры, система также имеет в своем распоряжении данные о местоположении жителей. Если по данным геолокации никого нет дома, то система отклоняет посетителей. Также для пользователя предусмотрена возможность включить такой режим вручную. Это может быть полезно, например, если в какой-то период времени жителей не должны беспокоить.

Система также должна содержать графический пользовательский интерфейс, через который можно как наблюдать за выбранными показателями и состоянием устройств, так и управлять поведением системы. Кроме того, через пользовательский интерфейс жители могут получать уведомления о том, что кто-то пытается войти в помещение.

Таким образом, можно выделить следующие функциональные требования к системе:
- Система позволяет задать несколько профилей жителей таким образом, чтобы их можно быть идентифицировать на входе;
- При положительной идентификации система открывает дверь автоматически;

- В состоянии по умолчанию камера и освещение выключены, однако они должны включаться по необходимости;
- Система может определить, находится ли кто-либо из жителей дома или нет. В случае, если никого нет дома, система должна сообщить посетителю об этом. В противном случае система должна предложить поговорить с людьми внутри помещения;
- Система также должна поддерживать режим “Не беспокоить”. Поведение системы при этом соответствует случаю, когда в помещении никого нет;
- Система должна уведомлять жителей о посетителях.

В дополнение к функциональным требованиям к системе также были разработаны следующие нефункциональные требования:

- Система может распознавать до 10 различных профилей жителей;
- Система должна корректно реагировать на сигналы устройств, даже если произошел разрыв соединения;
- Допустимо также ручное управление, в частности, использование ключа для того, чтобы открыть дверь;
- Поддерживается расширение системы большим количеством устройств.

Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми

В условиях снижающейся рождаемости доля пожилых людей в обществе растет, в то время как доля людей трудоспособного возраста снижается. Пожилые люди часто нуждаются в помощи с заботой о здоровье, свободном перемещении и в случае нежелательных происшествий. Благодаря использованию технологий интернета вещей, такие люди могли бы жить в большей безопасности и комфорте. Сфера интернета вещей, связанная с наблюдением за пожилыми людьми, также называется *a.a.l.* (ambient assisted living, проживание с фоновым сопровождением).

В *Andrushevich A..ZigBe814T-2009art* были выделены следующие сферы применения технологий *интернета вещей*:

- информационная помощь (легкодоступность всей необходимой информации);
- умное ситуационное поведение (среда должна узнавать типичные шаблоны поведения и предлагать соответствующую помощь);
- предсказание нежелательных событий (распознавание таких ситуаций на основании поведенческих и физиологических показателей, а также применение превентивных мер);
- распознавание нежелательных ситуаций и реакция на них;
- безопасность (защита от вторжений с использованием авторизационных и аутентификационных механизмов);
- конфиденциальность (минимизация вмешательства в личную жизнь).

Для удовлетворения указанных потребностей система может полагаться как на исторические, так и получаемые в реальном времени данные. Выделяют две группы датчиков. Данные об окружающей среде исходят от датчиков окружения, например, температуры, движения. Данные о поведении и состоянии человека исходят от носимых датчиков. В системе также могут присутствовать устройства обратной связи для визуального и голосового уведомления о различных событиях. Современные технологии беспроводной связи позволяют создавать надежную, легкую в установке и недорогую инфраструктуру для передачи данных.

Центральным для систем *a.a.l.* являются сбор и хранение данных о поведении пользователя, выделение шаблонов и определение нежелательных ситуаций на основании отклонений от них. Состояние пользователя может быть определено на основе данных от нескольких датчиков, расположенных в жилище (см. *Biallas M..LivinSaAiaAtHF-2017art*).

Среди нежелательных ситуаций особо отметим падения. Одним из способов распознавания падений является совместное использование носимого датчика ускорения и атмосферного давления. В случае неожиданного ускорения система дважды считывает данные о давлении, на основании которых делает вывод о положении тела человека (см. *Andrushevich A..ZigBe814T-2009art*).

С точки зрения моделирования таких систем можно выделить три категории показателей. К физическим аспектам относятся как условия внешней среды, так и параметры здоровья человека. В связи с непрерывной природой этой категории наиболее естественным представляется метод системной динамики в сочетании со стохастическим моделированием для учета роли случайности. С другой стороны, дискретно-событийное моделирование позволяет учесть появляющиеся события, например, падения. Наконец, спонтанное поведение пользователя наилучшим образом моделируется с помощью агентного метода.

Пункт 7.6.4.3. Подсистема для управления освещенностью жилища

Примерные функциональные требования к приложению: при входе в помещение свет загорается, среагировав на движение, а через 10 секунд после выхода из помещения свет отключается. Яркость освещения должна корректи-

роваться с учетом уровня уличного освещения, проникающего в жилище. Кроме того, освещение с 23-00 до 6-00 утра должно работать в режиме ночника, с пониженной яркостью.

Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью

В связи с нарастающим энергетическим кризисом и неадекватностью традиционных централизованных энергосистем новым вызовам появилась новая модель: гибридное распределенное производство энергии со значительным вкладом возобновляемых источников энергии. Такая модель также характеризуется двунаправленным потоком информации и электричества. Существуют решения на всех этапах производства энергии, однако на стороне потребителя основным является направление *интернета вещей*, в частности, домашней автоматизации. Такие системы получили название *h.e.m.s.* (home energy management system, система управления энергопотреблением для жилых домов) (см. *Stojkoska B.R..aRevie oIoTfSH-2017art*)

К системам управления энергией предъявляют следующие требования (см. *Zhou B..SmartHEMSCCaSS-2016art*):
- Система собирает в реальном времени данные о потреблении и получении энергии, а также о состоянии устройств;
- Система сохраняет и анализирует исторические данные;
- Система управляет устройствами в своих рамках таким образом, чтобы обеспечить оптимальное энергопотребление;
- Пользователь имеет возможность управления устройствами напрямую и удаленно;
- В случае возникновения нежелательных ситуаций система предупреждает пользователя.

Рассмотрим выбор метрик для оценки работы систем управления энергопотреблением.

Среда работы систем управления энергопотреблением — жилое помещение, домохозяйство, даже офисное или производственное здание — приводит к многоцелевому характеру таких систем (см. *Beaudin M..HomeEMSaRoM-2015art*), то есть к необходимости нахождения компромисса между несколькими задачами. В то время как глобальной целью всегда является повышение эффективности использования энергии, на ее достижения налагаются ограничения, в частности, по комфорту пользователей. Кроме того, формулировка конкретной задачи может разниться в зависимости от контекста: снижение выбросов, денежная экономия, балансировка нагрузки на энергосистему — вот некоторые возможные направления.

Цели управления потреблением энергии могут быть выражены через стоимость. Хотя наиболее очевидным и преобладающим фактором является цена потребляемой энергии, в общую формулу расчета также могут включаться такие факторы, как затраты на начальную установку системы, штраф за вклад в общую нагрузку, прогнозируемый износ оборудования, налог на выбросы парниковых газов (см. *Beaudin M..HomeEMSaRoM-2015art*). Таким образом, система может следовать единой цели — снижение получаемых по этой формуле денежных затрат. Выбор корректного соотношения может представлять сложность, если нет устоявшихся денежных значений для некоторых целей.

Основная задача из тех, которые сложно представить в денежном эквиваленте, — это комфорт пользователей, то есть их неудобства, связанные с качеством услуг, предоставляемых при доставке энергии. Система не должна приводить к значительному изменению их образа жизни. Для оценки влияния системы на комфорт пользователей могут использоваться различные штрафные функции: ограничения по значению, отклонение, штраф за отсутствие сервиса (см. *Beaudin M..HomeEMSaRoM-2015art*). Иначе говоря, если стоимость представляет собой некоторое соотношение, которое требуется минимизировать, то требования по комфорту налагают ограничения на возможные стратегии.

Описанные выше соображения могут применяться в тестировании общих подходов к оптимизации потребления энергии. Например, может потребоваться оценить выгоды от добавления того или иного устройства и сравнить их с ценой приобретения и установки, или же выбрать из нескольких разрабатываемых алгоритмов наилучший.

С точки зрения конечного пользователя к системе могут предъявляться такие требования, как доступ к историческим данным и тенденциям, возможность удаленного управления устройствами, предупреждение о нежелательных ситуациях (см. *Zhou B..SmartHEMSCCaSS-2016art*). Конфиденциальность данных, надежность и эффективность системы также играют роль в оценке ее качества. Подобные требования скорее характерны для продуктов практической направленности, предназначенных для прямого использования по назначению, а не исследовательских прототипов.

§ 7.6.5. Элементы технической реализации умных домов

⇒ *подраздел**:
- *Пункт 7.6.5.1. Запуск Node-RED на виртуальной машине в Yandex Cloud*
- *Пункт 7.6.5.2. Создание объектов сервиса Yandex IoT Core. Интеграция с Node-RED*

Проектирование и программная реализация указанных компонентов/приложений осуществлялось с помощью инструмента визуального программирования *Node-RED* в сочетании с использованием облачных технологий, в частности, *Yandex IoT Core* или *AWS IoT Core*. *Node-RED* — это инструмент потокового программирования для соединения аппаратных устройств, *программных интерфейсов* и онлайн сервисов. Их совместное позволяет создавать прототипы систем интернета вещей без использования реальных устройств, что позволяет проработать архитектуру системы до ее физической реализации. Одним из немаловажных преимуществ *Node-RED* является то, что с помощью этого инструмента возможно создать простой прототип системы в той же среде, в которой ведется или будет вестись основная разработка. Такой подход значительно упрощает разработку. Кроме того, *Node-RED* также предоставляет средства для простой визуализации получившейся системы, что позволяет создать прототип *графического интерфейса* в рамках этой же среды.

Использование приложениями облачных сервисов, к которым относятся *Yandex IoT Core* и *AWS IoT Core*, позволяет значительно уменьшить затраты на инфраструктуру системы, при этом обеспечивая лучшую масштабируемость и отказоустойчивость. Облачные технологии предлагают вычислительные ресурсы, ресурсы для хранения данных и ресурсы коммуникации.

Для передачи сообщений внутри системы в этом случае используется *MQTT* — сетевой протокол, использующий архитектуру издатель-подписчик и работающий поверх *TCP/IP* с использованием очередей (Message Queuing Telemetry Transport). Он удобен при использовании при невысокой пропускной способности сети.

Реализация прототипа системы для управления освещенностью жилища была проведена согласно описанию (см. *ПрогрУОпДДиО-2022эл*).

Пункт 7.6.5.1. Запуск Node-RED на виртуальной машине в Yandex Cloud

Для исполнения среды *Node-RED* создается виртуальная машина в облачном сервисе *Yandex Compute Cloud*. Этот сервис является частью Yandex Cloud и предоставляет масштабируемые вычислительные мощности для создания виртуальных машин и управления ими. Данный сервис предлагает широкий выбор настроек виртуальных машин, от различных операционных систем до тонкой настройки используемых ресурсов.

Нами была использована виртуальная машина на базе операционной системы *CentOS 8*. Поскольку для работы с *Node-RED* и для исполнения прототипа приложения не требуется больших вычислительных возможностей, ресурсы виртуальной машине были выделены минимальные.

Для доступа к виртуальной машине использовался *SSH*. Для этого в настройках облачных сервисов Yandex был выделен публичный ip-адрес. Доступ к виртуальной машине по *SSH* необходим для установки *Node-RED*, а разработка может вестись в обычном веб-браузере. После генерации пары ключей для SSH и задания соответствующего публичного ключа в настройках доступа к виртуальной машине к ней можно подключиться с использованием интерфейса командной строки. Для установки *Node-RED* был использован официальный ресурс “Linux Installers for Node-RED”. Также был включен автоматический запуск *Node-RED* при запуске виртуальной машины. Доступ к *Node-RED* был настроен на порт 1880.

Пункт 7.6.5.2. Создание объектов сервиса Yandex IoT Core. Интеграция с Node-RED

Основными элементами сервиса *Yandex IoT Core* являются устройство и реестр. Эти объекты могут обмениваться данными и командами по протоколу *MQTT*. Устройство в данном сервисе представляет собой абстракцию физического устройство, а реестр является группой устройств, которые логически связаны друг с другом. Данные могут передаваться между устройством и реестром.

Чтобы добавить устройства, необходимо сначала создать реестр. Для доступа к устройствам и реестрам *IoT Core* предлагает задать пароль, но можно также добавить сертификат. Для данной системы создадим один реестр и три устройства: освещение, камеру и модуль геолокации. Последний нужен только для целей прототипирования и является абстракцией для реальных устройств. Датчики создавать в прототипе не нужно, поскольку эта часть реализуется через *Node-RED*.

Для обмена *MQTT*-сообщениями между устройствами и реестрами в *IoT Core* предусмотрен MQTT-брокер, который отвечает за получение, обработку и доставку сообщений. *Node-RED* поддерживает соединение с MQTT-брокером через специальные узлы *MQTT Input* и *MQTT Output*. Обычно реестрам соответствуют узлы *MQTT Input*, поскольку этот узел позволяет подписаться на сообщения какой-либо темы ("топики") и таким образом его данные можно передать на обработку. Устройствам же чаще соответствуют узлы *MQTT Output*. Чтобы подключить эти узлы к объектам Yandex IoT Core, требуется указать идентификаторы, а также созданные пароли. Наконец, MQTT-узлы необходимо подписать на нужные топики, чтобы верно передавать сообщения.

Заключение к Главе 7.6.

В главе рассмотрен подход к описанию предметной области в приложениях интернета вещей на примере умного дома на основе Технологии OSTIS.

Полученные результаты в будущем позволят повысить эффективность компонентного подхода к разработке приложений в интернете вещей, а также обеспечить возможность автоматической синхронизации различных версий компонентов, повышая совместимость и согласованность.

Глава 7.7.

Автоматизация производственной деятельности в рамках Экосистемы OSTIS

⇒ *автор**:
- *Иванюк Д. С.*
- *Таберко В. В.*
- *Прохоренко В. А.*
- *Смородин В. С.*

⇒ *аннотация**:

[Современные направления научно-технической деятельности в области оптимизации функционирования производств требуют разработки актуальных подходов в реализации адаптивного управления производственной деятельностью с использованием элементов искусственного интеллекта, нейросетевого моделирования и разработки интеллектуальных компьютерных систем на основе использования новых технологий.

В настоящем разделе предлагается подход к построению интеллектуальной компьютерной системы адаптации управления нового поколения в рамках Экосистемы OSTIS. Формализация компьютерной системы адаптации управления реализуется на основе онтологии предметной области “технологические процессы производства с вероятностными характеристиками”, реализованной посредством применения ostis-систем в рамках Технологии OSTIS.

В основе создания гибридной интеллектуальной системы адаптации управления лежит идея разработки математических моделей нейросетевых контроллеров, решающих задачи реализации методов и алгоритмов синтеза обратных связей по управлению технологическим циклом в зависимости от изменения параметров функционирования объекта управления в режиме реального времени, на основе использования средств программно-аппаратного сопряжения с технологическим циклом производства.]

⇒ *подраздел**:
- *§ 7.7.1. Адаптивное управление технологическим циклом производства на основе Технологии OSTIS*
- *§ 7.7.2. Построение умных предприятий рецептурного производства с помощью ostis-систем*

⇒ *ключевой знак**:
- *Industry 4.0*
- *ISA-88*
- *ISA-95*

⇒ *ключевое понятие**:
- *технологический процесс*
- *вероятностный технологический процесс*
- *технологическая операция*
- *микротехнологическая операция*
- *адаптивное управление*
- *автоматизированная система управления технологическим процессом*
- *рецептурное производство*
- *цифровой двойник*

⇒ *библиографическая ссылка**:
- *Голенков В.В..оОбучеИСкОС-2018ст*
- *Taberko V.V..Desig oBMEitC-2018art*
- *Смородин В.С..Метод иСИМТ-2007кн*
- *Таберко В.В..ПроекПРПвК-2018ст*
- *Taberko V..Princ fEtDaUoS-2020art*
- *Omidvar O..NeuraSfC-1997bk*
- *Smorodin V..Appli oNMfAC-2019art*
- *White D.A..Handb oICNF-1992bk*
- *Ivaniuk D.NeuroPIDCfaP-2013art*
- *Иванюк Д.С.НейроПОУ-2014ст*
- *Hagan M..NeuraNfC-1999art*
- *Smorodin V..Contr oaTCoPPB-2019art*
- *Smorodin V..AdaptCoRPS-2019art*

- *Smorodin V..SoftwCfACoaP-2022art*
- *Stanley K..EvolvNNtAT-2002art*
- *Watkins C..QL-1992art*
- *Mnih V..HumanLCtDRL-2015art*
- *Bakker B.ReinfLwLSTM-2001art*
- *Hochreiter D.LongSTM-1997art*
- *Sutton R.S..ReinfLaI-1998bk*
- *Golenkov V.V..OntolDoBME-2017art*
- *Голенков В.В..ПроекПРПнОО-2017cm*
- *ISA88BC-2022el*
- *Taberko V..OntolAtBEwI-2022art*
- *ISA95-2022el*
- *Lutska N..OntolMoDTiM-2022art*
- *ISA51-2022el*

Введение в Главу 7.7.

Современное направление конвергенции работ в области создания интеллектуальных систем (см. *Голенков В.В..оОбучеИСкОС-2018cm*) требует разработки соответствующего программного обеспечения с элементами когнитивных способностей на основе семантически совместимых технологий искусственного интеллекта. Концепция Industry 4.0 предполагает построение единой онтологической модели предприятия (см. *Taberko V.V..Desig oBMEitC-2018art*), включающей в себя описание оборудования и технологических процессов производства. Технология OSTIS предоставляет средства для построения такой модели, обеспечивает возможность построения "цифрового двойника" предприятия на основе формализованного описания соответствующей предметной области. Полученная онтологическая модель, таким образом, может выступать основой интеграции востребованных интеллектуальных решений по автоматизации и информационному обеспечению производственной деятельности.

В первой части главы рассматривается подход к построению системы адаптивного управления технологическим процессом производства в виде решателя соответствующей ostis-системы на основе онтологии предметной области "технологические процессы производства с вероятностными характеристиками". В основу функционирования предлагаемой системы положено применение нейросетевых контроллеров. В основу формализации контура управления и математических моделей объекта исследования положены результаты научных разработок авторов в области имитационного моделирования сложных технических систем (см. *Смородин В.С..Метод иСИМТ-2007кн*). Такая реализация позволяет обеспечить возможность интеграции предлагаемого решения с другими разработками, программными средствами предприятия для обеспечения построения интеллектуальных систем автоматизированного управления, рекомендательных систем и систем поддержки принятия решений, систем информационного обеспечения персонала предприятия.

Во второй части главы рассматриваются вопросы построения онтологической модели предприятия на примере предметной области "рецептурные производства" с применением общепринятых международных стандартов описания содержания производственной деятельности предприятия. Формализация стандартов является основой подхода к проектированию предприятия, в ее процессе требуется принять во внимание сложную специфику предметной области, возможность неоднозначной трактовки положений и необходимость обеспечения актуализации используемых стандартов. Онтологический подход к построению умных предприятий в рамках Экосистемы OSTIS показан на примере формализованного описания физической модели предприятия ОАО "Савушкин продукт" и построения системы автоматизации деятельности инженера-технолога.

§ 7.7.1. Адаптивное управление технологическим циклом производства на основе Технологии OSTIS

⇒ *подраздел**:
- *Пункт 7.7.1.1. Онтология предметной области "технологические процессы с вероятностными характеристиками"*
- *Пункт 7.7.1.2. Решатели задач ostis-системы адаптивного управления вероятностным технологическим процессом производства*

В основу формализации контура управления и математических моделей объекта исследования положены результаты научных разработок авторов в области имитационного моделирования сложных технических систем (см. *Смородин В.С..Метод иСИМТ-2007кн*).

Пункт 7.7.1.1. Онтология предметной области “технологические процессы с вероятностными характеристиками”

⇒ *подраздел**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*
- *Подпункт 7.7.1.1.2 Надежностные характеристики функционирования оборудования*
- *Подпункт 7.7.1.1.3 Параметры (переменные) управления технологическим циклом*
- *Подпункт 7.7.1.1.4 Математическая модель микротехнологической операции*
- *Подпункт 7.7.1.1.5 Модель функционирования устройств оборудования*
- *Подпункт 7.7.1.1.6 Математическая модель вероятностного технологического процесса*
- *Подпункт 7.7.1.1.7 Имитационное моделирование технологических процессов*
- *Подпункт 7.7.1.1.8 Формализация систем управления на основе агрегатного способа имитации*
- *Подпункт 7.7.1.1.9 Пример построения технологического процесса производства в рамках Industry 4.0*

Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов

Центральными понятиями в рассматриваемой предметной области являются технологический процесс (цикл) и вероятностный технологический процесс (*Смородин В.С..Метод иСИМТ-2007кн*).

технологический процесс
:= [т.п.]
:= [технологический цикл]
:= [т.ц.]
:= [установленная технологическими документами последовательность взаимосвязанных действий, направленных на объект процесса с целью получения требуемого конечного результата]
:= [множество технологических операций $\{TCO_{ij}\}$, где $i, j = \overline{1, N}$, в совокупности с используемыми ими ресурсами]

В рамках стандарта ISA-88 в модели процедурного управления (procedural control module) это является процедурой (procedure), в результате выполнения которой появляется конечной партия продукта (возможно полуфабриката, который будет использоваться в дальнейшем для производства других партий продукции).

вероятностный технологический процесс
:= [в.т.п.]
⊂ *технологический процесс*
:= [технологический процесс с вероятностными параметрами функционирования]
:= [технологический процесс с изменяющейся в ходе его реализации структурой технологического цикла]

технологическая операция
:= [ТСО]
⊂ *технологический процесс*
:= [часть технологического процесса, выполняемая непрерывно на одном рабочем месте над одним или несколькими одновременно обрабатываемыми или собираемыми изделиями, одним или несколькими исполнителями]

В рамках стандарта ISA-88 это является фазой (phase), в результате выполнения которой появляется несущественно изменяются свойства продукта. Шагами управлять оператор не может (стартовать/ставить на паузу/продолжать). Они могут выполняться последовательно, параллельно или в комбинации данных вариантов.

микротехнологическая операция
:= [МТСО]
⊂ *технологическая операция*
:= [конечная последовательность элементарных операций, составляющих в совокупности содержание технологической операции, выполняемой непрерывно на одном рабочем месте]

Традиционно рассматривают две группы технологических процессов:
- непрерывного типа
- дискретного типа

Первая группа технологических процессов обычно реализуется на производстве в режиме реального времени. Данные технологические процессы являются объектом а.с.у.т.п. Примером применения а.с.у.т.п. является контроль за

процессом выплавки стали, контроль за поступлением сырья в мартеновские печи и автоматической разливкой металла по формам. Вторая группа технологических процессов характеризуется графовой структурой организации технологического цикла, который реализуется в результате взаимодействия множества технологических операций $\{TCO_i\}$. Некоторые TCO_i могут состоять, в свою очередь, из множества микротехнологических операций $\{MTCO_{ij}\}$. В зависимости от способа вхождения $\{MTCO_{ij}\}$ в состав $\{TCO_i\}$ различают следующие типы технологических процессов:

- одноуровневые, в которых $\{MTCO_{ij}\}$ могут выполняться параллельно-последовательно (в соответствии с графовой структурой связей между $MTCO_{ij}$);
- иерархические, в которых основная технологическая ветвь разделяется на несколько дочерних, после чего происходит обратное слияние; при этом в составе $MTCO_{ij}$ любого уровня иерархии имеются операции "расщепления" технологических линий и операция "сбора" технологических линий;
- итеративные, в которых дочерние операции вложены в основные операции; при этом результаты выполнения операций на одном уровне вложенности используются для выполнения дочерних технологических ветвей.

Моделирование всех типов технологических процессов осуществляется либо на основе критических, либо на основе средних значений расхода ресурсов. Наиболее сложными для моделирования считаются вероятностные технологические процессы второго и третьего типов.

При выполнении $MTCO_{ij}$ осуществляется расход ресурсов технологического цикла. В зависимости от характера расхода этих ресурсов выделяют две группы процессов:

- детерминированные (д.т.п.), у которых расход ресурсов предприятия осуществляется по средним или максимальным значениям;
- вероятностные (в.т.п.), у которых одна часть ресурсов используется детерминированным способом, задаваемым списками ресурсов и их количеств, а другая часть ресурсов предприятия используется вероятностным способом, задаваемым с помощью функций вероятностей распределения значений ресурсов, необходимых для выполнения $MTCO_{ij}$.

Подпункт 7.7.1.1.2 Надежностные характеристики функционирования оборудования

Одноуровневые в.т.п. используют для своего выполнения устройства оборудования, которые могут иметь различные характеристики надежности функционирования. Надежностные характеристики оборудования являются вероятностными и характеризуются следующим образом: моменты отказа определяются функцией распределения вероятностей значений времени безотказной работы устройства оборудования $F(\tau_{NOw})$, по истечении которого происходит отказ выполнения функций устройством. Отказы могут быть простыми, и после интервала времени восстановления (τ_{ROw}) его функционирование возобновляется. Некоторые отказы могут с вероятностью (P_{f1}) приводить к простой аварии, которая требует для своей ликвидации дополнительного времени (τ_{EM1w}) и дополнительной стоимости (c_{EM1w}). В этом случае имеет место авария оборудования 1-го типа. С вероятностью (P_{f2}) после обычного отказа может происходить авария 2-го типа, которая требует для своей ликвидации некоторой последовательности специальных технологических операций. Каждая такая технологическая операция может требовать дополнительного времени ее ликвидации (τ_{liq}) и дополнительной стоимости ее выполнения (c_{liq}). В итоге ликвидация аварии 2-го типа может приводить к задержке выполнения $MTCO_{ij}$ из-за отказа оборудования на величину, равную сумме этих времен $\sum \tau_{liq}$, и росту стоимости выполнения $MTCO_{ij}$ также на величину суммы стоимости $\sum \tau_{liq}$. После ликвидации отказов аварий 1-го и 2-го типа происходит лишь задержка времени выполнения и увеличение стоимости ее выполнения, а технологический процесс продолжает функционировать с ухудшенными временными и стоимостными характеристиками его реализации.

Устройства оборудования обладают некоторым ресурсом выполнения своих функций, который постепенно уменьшается и зависит от времени активного использования устройства. Специалисты по надежности обычно оперируют понятием “время наработки устройства на отказ” (T_{TTF}). При достижении значения времени активного использования устройства этого порогового значения вероятность отказа резко возрастает, и поэтому на практике стремятся либо переключить устройство на резервное (с временем фактического использования устройства близком к нулю), либо выполнить профилактические работы с целью восстановления ресурса времени использования устройства.

Имеются также т.п., в которых появление аварии оборудования 3-го типа приводит к отказу функционирования всего т.п. Аварии третьего типа ликвидируются аналогично ликвидации аварии 2-го типа с использованием специального оборудования и специальных бригад исполнителей. Но нормальное функционирование т.п. приостанавливается.

Таким образом, проектное моделирование в.т.п. в силу вероятностного характера запросов ресурсов множеством $\{MTCO_{ij}\}$ и наличия отказов оборудования, природа которых также вероятностная, представляет собой сложную научную задачу.

Подпункт 7.7.1.1.3 Параметры (переменные) управления технологическим циклом

Некоторые $MTCO_{ij}$ в в.т.п. используют не только ресурсы, но могут выполнять ряд контрольных функций за изменением значений компонентов множества переменных управления в.т.п. U_s. При нормальном выполнении в.т.п. каждый компонент этого множества должен находиться в допустимых диапазонах между минимальным U_s^- и максимальным значением U_s^+ s-го компонента множества переменных управления U_s. Причем, изменения значения U_s может иметь вероятностную природу. При выполнении другой группы $MTCO_{ij}$ значения переменных управления могут корректироваться таким образом, чтобы U_s возвращалось в допустимые пределы ($U_s^- \leq U_s \leq U_s^+$) . Третья группа $\{MTCO_{ij}\}$ корректирует значения компонентов U_s . Некоторые $MTCO_{ij}$ используются только для индикации состояний оборудования и в.т.п.

По способу использования управляющих переменных можно разделить $\{MTCO_{ij}\}$ на следующие группы:
- индикации состояний т.п. (используют минимальное число ресурсов);
- исполнительные элементы (не контролируют и не меняют компоненты U_s);
- контролирующие выход U_s за допустимые диапазоны;
- восстанавливающие значения компонентов U_s в отведенные диапазоны их изменения.

В составе в.т.п. контролирующие $MTCO_{ij}$ могут изменяться вероятностным образом: выход за допустимые пределы (P_{thr}) на величину ($\triangle U_s$) , которая в ту или в другую сторону может изменяться также вероятностным образом.

Подпункт 7.7.1.1.4 Математическая модель микротехнологической операции

Состав ресурсов, которые может использовать каждая $MTCO_{ij}$, включает десять типов:
1. Устройство оборудования индивидуального пользования
2. Устройство оборудования общего пользования
3. Ресурс индивидуального пользования
4. Ресурс общего пользования
5. Индивидуальные исполнители
6. Бригады исполнителей
7. Материалы
8. Комплектующие
9. Стоимость выполнения операций
10. Время выполнения операций

Устройства оборудования индивидуального использования ($DEVBIN_r$) могут использовать какой-либо состав операций на время ее выполнения τ_{ij} Состав устройств общим количеством может задаваться либо персонального списком, либо $MTCO_{ij}$ требуется для ее выполнения только число таких устройств, которые свободны на момент запроса операций ресурсов первого типа ($r = 1$)

$MTCO_{ij}$ может затребовать для своего выполнения устройство оборудования общего пользования (n_2) специальным списком. На устройствах общего пользования номера r выделяет место для ij-й операции размером V_{r2} на время ее выполнения, которое после окончания операции свободно для выделения другой операции. V_{r2} является случайной величиной. Допускается конкуренция $\{MTCO_{ij}\}$ за устройства общего пользования и место на этих устройствах.

Первые два типа ресурсов, являясь устройствами оборудования, могут отказывать в функционировании при их использовании $MTCO_{ij}$.

$MTCO_{ij}$ может потребовать для своего выполнения ресурсы индивидуального пользования ($r = 3$) общим количеством n_3 и ресурсы общего пользования в количестве n_4 ($r = 4$). Ресурсы ($r = 3$) и ($r = 4$) могут задаваться либо списком, либо для случая, когда номер ресурса не имеет значения, используются первые свободные n_3 или n_4 ресурсов. От ресурсов ($r = 1$) и ($r = 2$) эти ресурсы отличаются только тем, что это абсолютно надежные устройства, которые выделяются в распоряжение $MTCO_{ij}$ на время ее выполнения, а затем снова перераспределяются между $\{MTCO_{ij}\}$, либо на принципах конкуренции их за эти ресурсы, либо специальным образом зарезервированными за операциями место на общем ресурсе ($r = 4$) также может выделяться нескольким операциям и величина запроса является случайной величиной, которая задается с помощью функции распределения $F_{6ij}(V_{r4})$.

Любая $MTCO_{ij}$ может требовать конкретное число исполнителей индивидуальных (n_5) и бригад исполнителей (n_6) . Состав бригад определяется списком размера (n_6). Эти запросы являются детерминированными величинами.

Ресурсы 7-10 имеют вероятностную природу. В общем случае для выполнения операции $MTCO_{ij}$ расходуется k_1-го типа материалы (mt_{k1}) и комплектующие k_2-го типа (prt_{k2}). Количества таких вероятностных ресурсов равны соответственно n_{7k1} и n_{8k2} и представляют собой детерминированные величины. Расход количества материалов

определяется с помощью функций распределения $F_{4k2ij}(mt)$ и расход количества комплектующих также задается с помощью соответствующих функций распределения $F_{3k2ij}(prt)$.

Для нормального выполнения операции $MTCO_{ij}$ требуется расход ресурсов $r = 9$ и $r = 10$ финансовых затрат и времени выполнения операции. Обе эти характеристики являются случайными величинами и поэтому запросы этих ресурсов задаются соответственно с помощью функций вероятности распределения их значений $F_{2ij}(C)$ и $F_{1ij}(\tau)$.

Подпункт 7.7.1.1.5 Модель функционирования устройств оборудования

Устройства оборудования используются двух типов: индивидуального использования ($DEVIN_{r1}$) номера r_1 и общего пользования ($DEVSHR_{r2}$) номера r_2. Для устройства $DEVSHR_{r2}$ имеется ограничение места, которые разные $MTCO_{ij}$ могут одновременно использовать размером V_{0r2}, задаваемым в начале имитации. За каждый $MTCO_{ij}$ резервируется определенное место V_{r2ij}, при этом всегда проверяется, чтобы сумма используемых мест на $DEVSHR_{r2}$ была бы меньше V_{0r2}. В противном случае формируется отказ $MTCO_{ij}$ в использовании $DEVSHR_{r2}$, что приводит к ожиданию появления места в случае завершения предыдущей $MTCO_{ij}$.

Устройства оборудования обоих типов могут отказывать в функционировании. При этом они имеют надежностные характеристики вероятностной природы. Для каждого времени нахождения устройства r в состояниях задаются функции вероятностей распределения значений времени:
- безотказного функционирования $\Phi_{1r}(\tau_{NOr})$
- восстановление работоспособности после простого отказа $\Phi_{2r}(\tau_{ROr})$
- ликвидации аварии 1-го типа $\Phi_{3r}(\tau_{em1})$;
- ликвидации аварии 2-го типа $\Phi_{5r}(\tau_{em2})$

При появлении аварий время использования оборудования (а следовательно и выполнение операции) возрастает и определяется с помощью функций вероятностей распределения значений стоимости:
- ликвидации аварии 1-го типа $\Phi_{4r}(Cem1)$;
- ликвидации аварии 2-го типа $\Phi_{6r}(Cem2)$

Задаются также при появлении отказов оборудования вероятности возникновения:
- ликвидации аварии 1-го типа (P_{em1});
- ликвидации аварии 2-го типа (P_{em2})

Таким образом, время выполнения на оборудовании r номера операции $MTCO_{ij}$ равно $1 - (P_{em1} + P_{em2}) = P_{5A}$. Каждое r-е устройство оборудования обладает пороговым значением времени наработки t_{optr}.

Подпункт 7.7.1.1.6 Математическая модель вероятностного технологического процесса

При реализации последовательных вероятностных технологических процессов (в.т.п.1) только одна из $MTCO_{ij}$ выполняется в текущий момент времени. При этом она может использовать все ресурсы предприятия или же эти ресурсы заранее распределены между операциями до начала реализации в.т.п.1. Поэтому в последовательных в.т.п. используются только устройства индивидуального пользования. Кроме того, в в.т.п.1 имеется несколько нестандартных $MTCO_{ij}$:
- одиночного оперативного резервирования устройств оборудования в тех случаях, когда фактическое время наработки на отказ t_{n1} приблизится к критическому значению времени наработки $T_{N_{crit}}$;
- групповое резервирование устройств оборудования, если время наработки на отказ достигает критической величины одновременно у группы устройств;
- проведение профилактического ремонта всех устройств оборудования, когда процент устройств, требующих резервирования очень высок и при этом допустима остановка выполнения в.т.п.1;
- ликвидации аварий на устройствах для тех случаев, когда режимы резервирования и профилактики не удается заблаговременно выполнить.

система управления
- ≔ [СУ]
- ≔ [строго определенный набор программно-аппаратных средств для управления подконтрольным объектом, обеспечивающий возможность сбора показаний о его состоянии и воздействий на его поведение для достижения заданных целей]

автоматизированная система управления технологическим процессом
- ≔ [а.с.у.т.п.]
- ⊂ *система управления*

:= [комплекс технических и программных средств, обеспечивающих работу технологического оборудования в автоматическом (или автоматизированном) режиме в соответствии с выбранным критерием управления]

Быстротекущий вероятностный технологический процесс (в.т.п.2) обычно выполняется с высокой скоростью. Поэтому имитационная модель не успевает вмешиваться в динамику его реализации. Для этой цели имеется система управления (СУ), которая с помощью специального оборудования управляет его функционированием. Здесь также используется только оборудование индивидуального использования. Но это оборудование может выходить из строя. Поэтому крайне важно обеспечить своевременное резервирование устройств оборудования для предотвращения отказов и аварий в в.т.п.2. Состав и структура СУ в.т.п.2 известны технологу до имитации, и обычно это логические схемы, имеющие графовую структуру.

Вероятностный технологический процесс с параллельно-последовательной организацией (в.т.п.3) обычно имеет довольно низкую скорость реализации. Кроме того, эксперту-технологу известна технологическая схема реализации в.т.п.3. Как правило, группы выполняются одновременно, однако сохраняется последовательный характер выполнения каждой $MTCO_{ij}$. Это означает, что внутри технологической схемы в.т.п.3 имеются устройства синхронизации типа “И”, которые срабатывают по последнему во времени приходу сигналов активизации, после завершения предыдущих $MTCO_{ij}$. Эти сигналы имеют сложную структуру за счет их информационной "подкраски" признаками π_{emij} (наличие аварии на оборудовании при выполнении $MTCO_{ij}$) и π_{uij} (наличие выхода компонентов вектора управления U_k за допустимые пределы). После прихода последнего во времени сигнала на схему совпадения “И” на выходе формируется множество сигналов, активизирующих другую группу $\{MTCO_{ij}\}$. Реализация в.т.п.3 начинается с первой $MTCO_{11}$ и завершается последней схемой совпадения “И”. Все схемы синхронизации пронумерованы от 1 до n, а $MTCO_{ij}$ имеет двойную нумерацию (i номер предыдущей схемы синхронизации, а j номер последующей схемы синхронизации). Число входов и выходов у схем синхронизации может быть любым. Структура в.т.п.3 определяется графом связи $MTCO_{ij}$, который задается с помощью таблицы коммутации операций в в.т.п.3.

В общем случае в.т.п.3 может использовать все ресурсы и оборудование в режиме конкуренции. Поэтому некоторые $MTCO_{ij}$ могут ожидать освобождения оборудования, а затем начинается имитация их выполнения. Связь между $MTCO_{ij}$ и собственно устройствами в.т.п.3 может осуществляться либо через специально выделенное оборудование индивидуального пользования, либо через любое из устройств оборудования, выделяемого $MTCO_{ij}$ на основе конкуренции. . Важной особенностью в.т.п.3 является возможность регулирования технологического процесса с помощью контроля за содержимым управляющих переменных и модификации значений компонентов вектора переменных управления $\{U_k\}$ с целью возврата их в отведенные для них диапазоны значений. Выполнение этой коррекции осуществляется специальными резервными $MTCO_{ij}$, регулирующими значения $\{U_k\}$. Кроме того, ведется контроль за надежностью выполнения оборудованием своих функций. После ликвидации аварий оборудования в технологической цепи предусмотрены резервные $MTCO_{ij}$, которые ликвидируют последствия аварий на оборудовании. Наличие аварий оборудования и задержка активизации $MTCO_{ij}$ из-за нехватки ресурсов приводит к общему увеличению времени и стоимости выполнения всего множества $MTCO_{ij}$. Поэтому важным откликом ИМ в.т.п.3 является время выполнения $MTCO_{ij}$ за один цикл имитации.

Параметрами моделирования являются начальное значение количества ресурсов, предоставленное в распоряжение всех $MTCO_{ij}$, и количество устройств оборудования, с помощью которых реализуется технологический цикл. Откликами ИМ в.т.п.3 являются суммарные расходы ресурсов вероятностного типа на один цикл имитации, а также коэффициенты использования ресурсов в.т.п.3 η_r. Статистиками имитации являются вектора значений удельных весов времени свершения событий в технологической схеме в.т.п.3.

Если в.т.п.3 является циклическим, то важными статистиками становятся среднее время цикла выполнения всего множества $MTCO_{ij}$ до завершения имитации функционирования в.т.п.3 и минимально допустимый состав ресурсов и оборудования, при котором возможна его реализация.

Подпункт 7.7.1.1.7 Имитационное моделирование технологических процессов

Состав и функции операторов алгоритма имитации $MTCO_{ij}$ включают в себя:

1. Начало активизации имитации операции - Интервал времени - A_0, Время - t_{aij}
2. Формирование заказа ресурсов - Интервал времени - A_1, Время - t_{1ij}
3. Захват свободных ресурсов на время выполнения операций - Интервал времени - A_2, Время - t_{2ij}
4. Ожидание выделения освободившегося ресурса - τ_{wij} - Интервал времени - A_3
5. Захват освободившегося ресурса - Интервал времени - A_4, Время - t_{3ij}
6. Ожидание следующего освободившегося ресурса - τ_{w2ij} - Интервал времени - A_5
7. Захват последнего из затребованных устройств на время выполнения - Интервал времени - A_6, Время - t_{4ij}
8. Выполнение имитации операции - τ_{r2ij} - Интервал времени - A_7
9. Конец выполнения операции, возврат захваченных ресурсов - Интервал времени - A_8, Время - t_{5ij}
10. Формирование информации о характере использования оборудования - Интервал времени - A_9, Время - t_{9ij}

11. Формирование информации о модификации значений компоненты вектора управления U_k - Интервал времени - A_10, Время - t_{7ij}
12. Формирование адресата продолжения операции - Интервал времени - A_11, Время - t_{8ij}

Надежностные характеристики функционирования устройств оборудования включают в себя:

1. Время безотказного функционирования - с распределением $\Phi_{1r}(\tau_{NOr})$
2. Время восстановления работоспособности - с распределением $\Phi_{2r}(\tau_{RO})$
3. Вероятность появления аварии 1-го типа - с вероятностью P_{em1}
4. Время ликвидации аварии 1-го типа - с распределением $\Phi_{3r}(\tau_{em1})$
5. Стоимость ликвидации аварии 1-го типа - с распределением $\Phi_{4r}(C_{em1})$
6. Вероятность появления аварии 2-го типа - с вероятностью P_{em2}
7. Время ликвидации аварии 2-го типа - с распределением $\Phi_{5r}(\tau_{em2})$
8. Стоимость ликвидации аварии 2-го типа - с распределением $\Phi_{6r}(C_{em2})$
9. Время наработки устройства на отказ - T_{optr}

Состав и структура операторов алгоритма имитации функционирования устройств оборудования индивидуального пользования включает в себя:

1. Начало активизации устройства r - оператор B_0 - Момент времени - t_{ar}
2. Розыгрыш по функции распределения τ_{NOK} - оператор B_1 - Момент времени -t_{ar}
3. Определение типа использования оборудования - оператор B_2 - Момент времени -t_{1r}
4. Имитация нормального использования устройства длительностью - Интервал времени - τ_{ij} - оператор B_3 τ_{ij} - Момент времени -t_{1r}
5. Формирование признака $\pi_{\Phi} := 0$ - оператор B_4 - Момент времени - t_{1r}
6. Активизация операции ij - оператор B_5 - Момент времени - t_{1r}
7. Розыгрыш по функции распределения τ_{ROK} - оператор B_6 - Момент времени -t_{1r}
8. Имитация восстановления работоспособности устройств - оператор B_7 - Интервал времени - τ_{ROr}
9. Розыгрыш по P_{em1} ситуации появления аварии 1-го типа - оператор B_8 - Момент времени -t_{2r}
10. Имитация повторения имитации - оператор B_9 - Интервал времени - τ_{ij}
11. Формирование признака $\pi_{emk} := 0$ - оператор B_{10} - Момент времени -t_{3r}
12. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор B_{11} - Момент времени -t_{2r}
13. Розыгрыш по функции распределения - Интервал времени - τ_{em1} - оператор B_{13}
14. Имитация ликвидации аварии 1-го типа - оператор B_{14} - Интервал времени - τ_{em1}
15. Имитация повторения операции - оператор B_{15} - Интервал времени - τ_{ij}
16. Формирование признака $\pi_{emk} := 1$ и активизация операции - оператор B_{16} - Момент времени -t_{4r}
17. Розыгрыш по функции P_{AV} ситуации появления аварии 2-го типа - оператор B_{17} - Момент времени -t_{2r}
18. Активизации последовательности процедур ликвидации сложной аварии и останов - оператор B_{18} - Момент времени -t_{2r}
19. Ожидание завершения ликвидации аварии 2-го типа - оператор B_{19} - Интервал времени - τ_{w2r}
20. Формирование признака $\pi_{emk} := 1$ и активизация операции - оператор B_{20} - Момент времени - t_{5r}

Состав и функции операторов алгоритма имитации функционирования устройств общего пользования включают в себя:

1. Активизация устройства - оператор - C_0 - Момент времени - t_{ar}
2. Определение типа использования устройства - оператор - C_1 - Момент времени - t_{ar}
3. Имитация нормального использования длительностью C_2 - Интервал времени - τ_{ij}
4. Формирование признака $\pi_{ak} := 0$ и переход - оператор - C_3 - Момент времени - t_{1r}
5. Розыгрыш по функции распределения τ_{ROk} - оператор - C_4 - Момент времени - t_{ar}
6. Имитация восстановления работоспособности устройств - оператор - C_5 - Интервал времени - τ_{ROk} - Момент времени - t_{2r}
7. Розыгрыш по P_{em1} ситуации появления аварии 1-го типа - оператор - C_6 - Момент времени - t_{2r}
8. Имитация появления аварии - оператор - C_7 - Интервал времени - τ_{ij}
9. Формирование признака $\pi_{ak} := 0$ и переход на C_{19} - оператор - C_8 - Момент времени - t_{3r}
10. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор - C_9 - Момент времени - t_{3r}
11. Розыгрыш по функции распределения τ_{em2} - оператор - C_{10} - Момент времени - t_{3r}
12. Имитация ликвидации аварии 2-го типа C_{11} - Интервал времени - τ_{em2}
13. Формирование признака $\pi_{emk} := 1$ и переход на C_{19} - оператор - C_{13} - Интервал времени - τ_{ij}
14. Розыгрыш по P_{em2} ситуации появления аварии 2-го типа - оператор - C_{15} - Момент времени - t_{2r}
15. Активизация последовательности процедур ликвидации сложной аварии и останов C_{16} - Момент времени - t_{2r}
16. Ожидание завершения ликвидации аварии 2-го типа - оператор - C_{17} - Интервал времени - τ_{w2k}
17. Формирование признака $\pi_{emr} := 1$ - оператор - C_{18} - Момент времени - t_{5r}
18. Розыгрыш по функции распределения τ_{NOk} нового значения интервала безотказного функционирования - оператор - C_{19} - Момент времени - t_{5r}

19. Активизация операции - оператор - C_{20} - Момент времени - t_{5r}

Таким образом описывается состав и функции операторов алгоритма имитации $MTCO_{ij}$. В момент активизации $MTCO_{ij}$ (t_{aij}) выполняется оператор "начало активизации имитации"операции (A_0). С помощью оператора A_1 формируется заказ ресурсов для имитации выполнения $MTCO_{ij}$. Для этой цели используются функции распределения вероятностей их значений. В итоге выполнения оператора формируется вектор детерминированных запросов ресурсов

$$(n_1, n_2, n_3, n_4, n_5, n_6, n_{7k1}, n_{8k2})$$

и подмножество конкретных значений вероятностных характеристик запроса ресурса

$$(\{V_{r2l}\}, \{V_{r4l}\}; \{mt_{k1}\}, \{kok2\}, c_{ijl}; \tau_{ijl})$$

Далее с помощью оператора захвата ресурсов A_2 на время выполнения операции осуществляется резервирование ресурсов, представленных множествами (1.1) и (1.2). В некоторых случаях из-за конкуренции $MTCO_{ij}$ за ресурсы может не оказаться свободных ресурсов или же места на ресурсах или устройствах. В этом случае с помощью оператора A_3 осуществляется ожидание освобождения ресурса длительностью τ_{w1ij}. По мере освобождения ресурсов осуществляется захват частично освободившихся ресурсов с помощью оператора A_4 . Далее с помощью оператора A_5 операция $MTCO_{ij}$ ожидает освобождения следующего ресурса длительностью τ_{w2ij}. С помощью оператора A_6 имитируется захват последнего из затребованных ресурсов. И в момент t_{4ij} начинается с помощью оператора A_7 имитация выполнения операции. Имитируется запуск всех устройств оборудования на заказанное время выполнения операции τ_{lij} сформированное с помощью функции распределения $F_{1ij}(\tau)$. Имитация выполнения операции продолжается до тех пор, пока все устройства оборудования не завершат имитацию выполнения операции. Из-за отказов устройств оборудования и ликвидации аварий оборудования фактическое время имитации операций может быть больше, чем заказанное время имитации $\tau_{zlij} \geq \tau_{ij}$.

С помощью оператора A_7 в момент времени t_aij по окончании имитации операции все захваченные $MTCO_{ij}$ ресурсы возвращаются системе. В этот момент возможно продолжение тех $MTCO_{ij}$, которые ждут освобождения ресурсов. По информации, поступившей от устройств оборудования о том, что имела место авария оборудования на хотя бы одном из устройств, формируется признак аварии ($\pi_{em} := 1$) оператором A_9.

С помощью оператора A_{10} формируется признак модификации значений компонентов вектора переменных управления за допустимые пределы (то есть $U_k \prec U_k^-$ или $U_k \succ U_k^+$) $\pi_u := 1$, в случае если $MTCO_{ij}$ является операцией изменения $\{U_k\}$. Для случая, когда $MTCO_{ij}$ является резервной операцией возврата переменных управления в допустимые пределы с помощью оператора A_{10} осуществляется обратная модификация $\{U_k\}$ таким образом, чтобы $U_k^- \leq U_k \leq U_k^+$ в момент времени t_7ij . Наконец, с помощью оператора A_{11} формируется адресат продолжения операции. При этом формируется сигнал сложного вида (информационно "подкрашенный" значениями признаков π_{aij} и π_{uij}), которые согласно таблице коммутации $MTCO_{ij}$ в составе технологической схемы в.т.п. посылается на одно из устройств синхронизации типа “ИЛИ” и “И”. Эти устройства имеют нумерацию входов и поэтому в адресате продолжения операции внутри сигнала указывается номер и номер входа устройства синхронизации.

Возможны четыре случая имитации операции на оборудовании:
- нормальное выполнение операции на всем оборудовании длительностью τ_{lij};
- имеет место восстановление функций оборудования хотя бы на одном из устройств $\tau_{2ij} > \tau_{lij}$;
- имела место авария 1-го типа на одном из устройств оборудования $\tau_{3ij} > \tau_{2ij} > \tau_{lij}$;
- имела место авария 2-го типа на любом из устройств оборудования $\tau_{4ij} > \tau_{3ij} > \tau_{2ij} > \tau_{lij}$.

Таким образом, наличие отказов и аварий устройств оборудования, а также ожиданий освобождения ресурсов в ряде случаев может существенно увеличить время выполнения операций по сравнению с разыгранным по функции распределения $F_{1ij}(\tau)$ с помощью жребия 3-го типа.

Это означает, что аварии оборудования приводят только к увеличению времени и стоимости выполнения $MTCO_{ij}$ и, как следствие, к росту общего времени и стоимости выполнения в.т.п.

В процессе имитации функционирования устройства оборудования фиксируется статистика фактического времени наработки устройства по формуле

$$t_{optr} := t_{optr} + \tau_{ij}$$

При достижении t_{optr} пороговой величины T_{opt} вероятностиь отказа устройства настолько возрастает, что при следующем использовании устройства r возможен отказ функционирования устройства, который может приведсти к появлению аварий на устройстве r во время выполнения очередной операции $MTCO_{ij}$.

Одним из способов предотвращения ситуации может служить либо переключение на резервные устройства, либо проведение профилактики. После проведения одной из этих операций фактическое время наработки устройства устанавливается в нуль $t_{optr} := 0$. Пороговое значение времени наработки указывается надежностных характеристиках для каждого r-го устройства оборудования.

В момент активации устройства номера r операцией $MTCO_{ij}$ (t_{ar}) с помощью оператора B_0 осуществляется запись в базу данных конкретного номера устройства оборудования индивидуального пользования. Используя таблицу 1.3, с помощью функции распределения $\Phi_{1r}(\tau_{NOr})$ разыгрывается конкретное значение интервала безотказного функционирования устройства τ_{lrNOK}.

Далее с помощью оператора B_2 в момент t_{2r} определяется тип использования оборудования. Если выполняется неравенство $\tau_{lij} \leq \tau_{lrNOK}$, то это означает нормальное использование устройства оборудования (без отказов). В подобном случае с помощью оператора B_4 формируется признак $\tau_{emr} := 0$, означающий отсутствие аварий.

С помощью оператора B_5 активизируется $MTCO_{ij}$, и имитация выполнения завершается с передачей информационного сигнала с признаком наличия аварии ($\pi_{emr} := 0$).

Если же $\tau_{lij} \geq \tau_{lrNOr}$, то это означает случай 2 (наличие отказа оборудования). В этом случае с помощью оператора B_6 в момент t_2 по функции распределения $\Phi_{2r}(\tau_{ROr})$ формируется значение интервала восстановления работоспособности τ_{RORl}. Затем осуществляется имитация восстановления работоспособности r-го устройства с помощью оператора B_7. По завершении этой имитации по вероятности P_{em1r} с помощью жребия 1-го типа моделируется появление аварии с помощью оператора B_8 в момент $(t_{2r} > \tau_{ROr})$. Далее операция повторяется с помощью оператора B_9 , который имитирует ее выполнение длительностью $\tau_l ij$. Эта ситуации означает отсутствие аварий, поэтому с помощью оператора B_{10} формируется признак $\pi_{em} := 0$. Далее с помощью оператора $B_1 1$ по вероятности P_{em1} разыгрывается ситуация появления аварии 1-го типа. Разыгрывается по функции распределения длительности ликвидации аварий с помощью оператора B_{13} и осуществляется имитация ликвидации аварий 1-го типа длительностью τ_{lem1} оператором $B_1 4$. Далее аналогично имитируется повторение операции длительностью τ_{lij} с помощью оператора B_{19}. По завершении этой имитации оператор B_{16} формирует признак “была авария” $\pi_{emr2} := 1$ и активизирует операцию $MTCO_{ij}$. В противном случае с помощью оператора B_{17} в момент t_{2r} по вероятности P_{a2} с помощью жребия 1-го типа формируется ситуация “наличие аварии” 2-го типа. С помощью оператора $B_1 8$ активизируется начальный элемент последовательности процедур ликвидации сложной аварии. Само же устройство r_1 останавливается и с помощью оператора $B_2 0$ имитируется ожидание завершения ликвидации аварии 2-го типа длительностью τ_{w2r}. Далее с помощью оператора B_{20} активизируется устройство оборудования типа 2. Устанавливается признак $\pi_{emr} := 1$, активизируется операция $MTCO_{ij}$ и устройство останавливается, завершая имитацию выполнения операции на устройстве в момент t_{5r}.

Может иметь место 4 случая:

- нормальное выполнение операции длительностью $(\tau_{\phi 1ij} = \tau_{ij})$;
- автоматическое восстановление отказа без аварии на устройстве оборудования r_1 длительностью $(\tau_{\phi 2ij} > \tau_{ij})$;
- ликвидация аварии 1-го типа длительностью $(\tau_{\phi 3ij} > \tau_{\phi 2ij} > \tau_{ij})$;
- наличие имитации аварии на оборудовании r_1 длительностью $(\tau_{\phi 4ij} \gg \tau_{ij})$.

Оборудование общего пользования имитируется по более сложному алгоритму. Надежностные характеристики устройств оборудования общего использования задаются аналогичным образом.

Основное отличие состоит в том, что в моменты завершения интервалы безотказного функционирования начинаются с начала имитации, поскольку это устройство функционирует непрерывно. Здесь также могут быть четыре случая:

- нормальное выполнение длительностью τ_{ij};
- наличие восстанавливаемого отказа длительностью $(\tau_{ROlr2} + 2\tau_{ij})$;
- ликвидация аварии 1-го типа длительностью $(\tau_{ROlr2} + \tau_{em1r2} + 2\tau_{ij})$;
- ликвидация аварии 2-го типа длительностью $(\tau_{ROlr2} + \tau_{O2r} + 2\tau_{ij})$.

Фактическое время выполнения операции возрастает с ростом сложности отказа оборудования. Сам же алгоритм имитации функционирования устройств общего пользования похож на алгоритм имитации функционирования устройств оборудования индивидуального пользования.

Вероятностный переход от одной к другой $MTCO_{ij}$ осуществляет элемент выбора операции. При этом он проверяет наличие критической ситуации устройств оборудования и своевременно активизирует нестандартные $MTCO_{ij}$. После ликвидации нестандартной ситуации у устройств оборудования осуществляется переход на выполнение стандартных $MTCO_{ij}$ согласно матрице вероятностей перехода $\parallel P_{ij} \parallel$, где i - номер предыдущей, а j - номер последующей $MTCO_{ij}$. Очевидно, что элемент выбора операции ресурсов не использует, поэтому это переключение осуществляется мгновенно в модельном времени. Поскольку в.т.п.1 может быть циклическим, то он может повторяться многократно, начиная с начальной $MTCO_{1j}$ и заканчивая последней $MTCO_{in}$.

В качестве исходной информации задаются надежностные характеристики функционирования устройств оборудования, а также начальное количество ресурсов, представляемые в распоряжение всех $MTCO_{ij}$. Для всех устройств оборудования указывается время наработки на отказ каждого устройства. Откликами ИМ в.т.п.1 являются среднее время цикла выполнения в.т.п.1 ($T_{\mu 1}$) и вектор коэффициентов использования ресурсов в.т.п.1 (η_k). Статистиками имитации являются вектора удельных весов времени и суммарной стоимости выполнения каждой ($z_r \tau_{ij}$ и $z_r c_{ij}$).

Подпункт 7.7.1.1.8 Формализация систем управления на основе агрегатного способа имитации

Состав и структура с.у. в.т.п.2 известны технологу до имитации, и обычно это логические схемы, имеющие графовую структуру.

На входе с.у. может быть несколько TCO_i, из которых в с.у. поступают сигналы активизации ее компонентов с интенсивностями λ_i.

Анализ особенностей с.у. в.т.п.п. позволяет установить что имитационное моделирование взаимодействия элементов систем управления и компонентов оборудования вероятностных процессов производства можно осуществить на основе блок-схем синхронизации их функционирования. Элементами такой синхронизации являются:

- синхронизатор первого типа $SLAST_k$ ($k = \overline{1,N}$) взаимодействия нескольких микротехнологических операций $MTCO_{ij}$ (i и j — номера синхронизаторов соответственно на входе и выходе $MTCO_{ij}$), функционирующий по логической схеме совпадения “И”;
- синхронизатор второго типа $SFIRST_k$ ($k = \overline{1,N}$) взаимодействия нескольких $MTCO_{ij}$, функционирующий по логической схеме совпадения “ИЛИ”;
- элементы $INDS_j$ с.у. в.т.п.п. являющиеся инициаторами цикла его функционирования с интенсивностью λ_j и формирования воздействий на вероятностный процесс через устройства оборудования (здесь j — номера элементов-синхронизаторов $MTCO_{ij}$, которые инициализируются данными элементами)
- элементы $INDF_i$ с.у., на которых завершаются цепочки управления выполнением функций в.т.п.п. (где i — номер индикатора предшественника данному элементу)
- исполнитель $ISPF_ij$ функциональных действий микротехнологической операции $MTCO_{ij}$, инициируемый синхронизатором номера i и посылающий сигнал синхронизатора на синхронизатор номера j;
- исполнитель $CORF_{ij}$ функциональных действий, корректирующий вектор глобальных переменных U_k при выходе за границы допустимых значений его компонентов; здесь i и j — номера элементов синхронизации соответственно на входе и выходе $CORF_{ij}$.
- исполнитель LIQ_{ij} функциональных действий, ликвидирующий последствия аварии оборудования в.т.п., имевшей место перед его выполнением, и приводящий с.у. в.т.п. в нормальное состояние (i и j — номера элементов синхронизации соответственно на входе и выходе LIQ_{ij})
- универсальный элемент-исполнитель $UNIV_{ij}$, который может одновременно корректировать значения векторов U_k и ликвидировать последствия аварий на оборудовании в.т.п.;
- индикатор состояний оборудования $INDS_{ij}$ вероятностного процесса; в отличие от остальных элементов управляющих воздействий на в.т.п. не посылает.

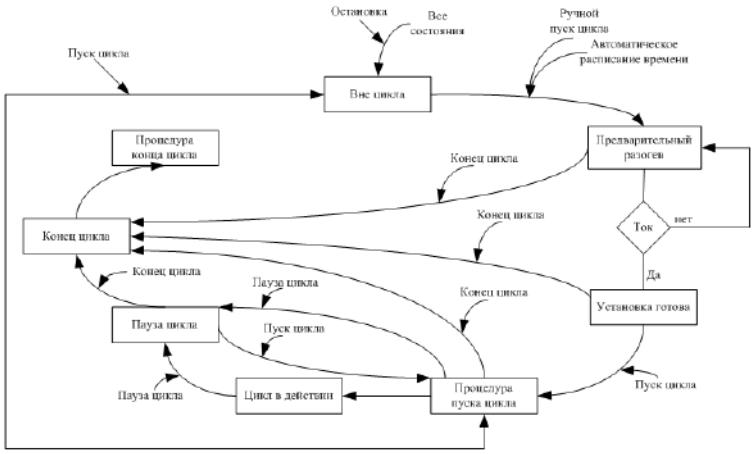
Взаимодействие исполнительных элементов с вероятностным процессом осуществляется с помощью множества устройств оборудования индивидуального ($DEVIN_{r1}$) и общего ($DEVSHR_{r2}$) пользования (r_1 и r_2 — номера устройств соответственно индивидуального и общего пользования). Исполнительные элементы с.у. в.т.п. с индексом ij представляют собой двухполюсники, на входы которых приходят сигналы из элементов синхронизации $SLAST_i$ и $SFIRST_j$, а на выходах формируются сигналы, поступающие на аналогичные элементы синхронизации с номером j. Связи между элементами с.у. осуществляются с помощью комбинации сигналов Sgn_{ij} сложной структуры, каждый из которых идентифицируется следующими характеристиками:

- индексом ij , показывающим направление его передачи (от i-го к j-му исполнительному элементу);
- булевым признаком pt_{ij} прихода сигнала в момент времени t_{ij}, принимающим значение $pt_{ij} = 1$ в момент прихода; в остальные моменты модельного времени t_{mod} (для $t_{mod} \leq t_{ij} pt_{ij} = 0$) элемент синхронизации находится в состоянии ожидания сигнала;
- признаком ps_{ij} типа сигнала, принимающим одно из четырех значений (00 — нормальное выполнение исполнительного элемента синхронизации; 01 — во время выполнения исполнительного элемента имела место ликвидация аварии на устройствах оборудования; 10 — произошел выход компонентов переменной управления U_j за допустимые границы диапазона значений; 11 — обе чрезвычайные ситуации имели место одновременно).

Для выполнения исполнительного элемента вероятностного процесса с индексом ij в общем случае требуются затраты 10 типов ресурсов системы (см. *Подпункт 7.7.1.1.4 Математическая модель микротехнологической операции*) Под нормальным выполнением исполнительного элемента понимается случай, когда во время выполнения операции управления не происходит отказов оборудования индивидуального и общего пользования.

Подпункт 7.7.1.1.9 Пример построения технологического процесса производства в рамках Industry 4.0

Рисунок. Пример схемы технологического цикла производства (линия эмалирования)
=



Для описания управления технологической линией эмалирования введено понятие “цикл” работы. Термин “цикл” подразумевает производственную фазу, в течение которой линия эмалирования производит готовые детали. Технологическая линия эмалирования считается готовой к производственной фазе или пуску “цикла” только после завершения этапа “предварительного разогрева”. Во время предварительного разогрева все "горячие" участки линии эмалирования доводятся до режимной температуры.

Объектом моделирования является технологическая схема организации производства для изготовления элементов электроаппаратуры специального назначения, которая имеет графовую структуру и поэтому может быть формализована в рамках разработанной модели.

Исходная информация для построения имитационной модели (и.м.) с.у. в.т.п. включает в себя множество параметров, характеризующих ее состав и структуру:

- количество реализаций имитационной модели (N) согласно процедуре Монте-Карло;
- множество $\{X_i\}$ запросов ресурсов ВТТП, требуемых для выполнения исполнительных элементов системы управления;
- параметры коммутации исполнительных элементов с элементами синхронизации, генераторами входных сигналов и финальным элементом функционирования;
- множество $\{REL\}$ надежностных характеристик оборудования, с помощью которого осуществляется управление функционированием в.т.п.;
- множество $\{\lambda_j\}$ значений компонентов важности откликов имитации;
- вектор $\{\delta_h\}$ коэффициентов важности откликов имитации;
- вектор DEVLIST конфигурации состава оборудовании, используемого в исследуемом варианте организации системы управления в.т.п.

Для заполнения этих параметров производится мониторинг выполнения функций с.у. в.т.п. Результаты обработки заполненных данных аппроксимируются некоторыми распределениями.

При выполнении множества микротехнологических операций ($\{MTCO_{ij}\}$) согласно технологической схеме организации производства для формализации технологического цикла можно использовать вероятностный сетевой график, заменяя компоненты технологической схемы на операции и события выполнения операций в ИМ ВСГР.

Многие операции изготовления элементов электроаппаратуры специального назначения используют оборудование индивидуального и общего пользования, которое может отказывать (а в ряде случаев могут возникать аварии на оборудовании). На оборудовании имеются специальные элементы индикации его использования и определения текущего состояния автоматики, регулирующей выполнение $\{MTCO_{ij}\}$. Кроме того, для повышения надежности функционирования оборудования может использоваться одиночное и групповое резервирование устройств. В экстренных ситуациях допускается переход оборудования на профилактику или ликвидацию последствий аварий оборудования.

Для управления динамикой выполнения $MTCO_{ij}$ и своевременного переключения оборудования на профилактику можно использовать имитационную модель ВСГР путем простой замены в ней устройств оборудования на агрегаты-имитаторы функционирования устройств оборудования общего и индивидуального пользования, а сами операции

на ИМ ВСГР представить в имитационной модели набором агрегатов-имитаторов событий и агрегатов-имитаторов технологических операций.

Пункт 7.7.1.2. Решатели задач ostis-системы адаптивного управления вероятностным технологическим процессом производства

⇒ *подраздел**:
- *Подпункт 7.7.1.2.1 Проблемы адаптивного управления производственной деятельностью при разработке интеллектуальных компьютерных систем нового поколения*
- *Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*
- *Подпункт 7.7.1.2.4 Формализация управления технологическим циклом производства на основе применения ostis-систем*
- *Подпункт 7.7.1.2.5 Принципы оптимизации процессов управления технологическим циклом с использованием нейросетевого моделирования*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*
- *Подпункт 7.7.1.2.7 Решатели задач ostis-системы адаптивного управления технологическим циклом производства*
- *Подпункт 7.7.1.2.8 Система адаптации управления*
- *Подпункт 7.7.1.2.9 Примеры реализации систем адаптации управления*

При осуществлении процессов управления технологическим циклом производства часто возникает потребность комплексного учета многообразия факторов воздействия на технологический процесс случайных сбоев используемого оборудования, а также воздействий внешней среды, включая и человеческий фактор. В этой связи является актуальной реализация адаптивного управления процессом производства (с обратными связями по управлению) в составе технических средств управления технологическим циклом и программного обеспечения адаптации процесса управления на случайные внешние возмущения в ходе его реализации.

Адаптивное управление технологическим циклом понимается как способность системы управления адекватно реагировать на внешние возмущения и штатные управляющие воздействия изменением соответствующих параметров управления в процессе функционирования системы.

Под оптимальными управлением понимается формализованная нейронной сетью структура адаптивного управления технологическим циклом, построенная в опорных узлах вероятностной сетевой графовой структуры (СГС) или модели полумарковской сети (ПМС) в рамках заданного критерия качества.

Формализация процесса управления технологическим циклом производства с вероятностными характеристиками основана на использовании в структуре контура управления специальных сигналов и стандартных элементов, которые в дальнейшем участвуют в формировании регулирующих воздействий непосредственно на используемое оборудование либо выдачи рекомендаций по выполнению определенных действий операторами-людьми.

Данные регулирующие воздействия (исполнение данных рекомендаций) могут приводить к изменениям параметров функционирования системы. Воздействия формируются в моменты изменения состояния ТЦ.

Необходимость оперативного изменения параметров функционирования системы возникает вследствие наличия внешних факторов; человеческого фактора, существования возможности возникновения отказов оборудования и аварий.

Система управления ТП связана с оборудованием ТП посредством средств программно-аппаратного сопряжения и в процессе реализации ТП получает сигналы о его состоянии. Изменение состояния ТП приводит к изменению состояния системы управления.

Для построения контроллера системы управления могут быть использованы статистические данные функционирования ТЦ, собранные с применением соответствующих средств аппаратно-программного сопряжения, либо может быть осуществлено использование имитационного моделирования функционирования ТЦ на базе модели ТЦ, построенной в соответствии с изложенной онтологией для технологических процессов с вероятностной природой.

Подпункт 7.7.1.2.1 Проблемы адаптивного управления производственной деятельностью при разработке интеллектуальных компьютерных систем нового поколения

Современный анализ состояния разработок в области исследования управляемых производственных систем показывает, что проблема определения параметров функционирования подобных объектов исследования в режиме реального времени возникает прежде всего при необходимости производства сложных технических изделий, требующих точности их изготовления и высокой производительности труда.

При этом в рамках решения многокритериальной задачи оптимизации управления предъявляются строгие требования к качеству и алгоритму выполнения производственного процесса, минимизации влияния человеческого фактора на качество реализации технологического цикла производства, исключению возникновения аварийных ситуаций техногенного характера. Подобная ситуация характерна для роботизированных производственных систем, работающих под управлением программно-аппаратного контроллера, который администрирует работу системы управления технологическим циклом в соответствии с заложенными программами.

Вместе с тем возникающие нештатные ситуации ввиду наличия сбоев оборудования, случайных внешних управляющих воздействий, включая и наличие человеческого фактора, приводят к отклонению параметров функционирования производственной системы от заданных, в связи с чем возникает необходимость корректировки параметров управления в режиме реального времени на основе моделей нейрорегуляторов, работающих в составе средств программного-аппаратного сопряжения с технологическим циклом производства.

Существующие специальные модели искусственного интеллекта, такие как нейронные сети, обладают уникальными свойствами, могут применяться в качестве универсальных аппроксиматоров, имеющих способность к обобщению. Эти особенности делают целесообразным применение моделей такого типа при решении сложных задач адаптивного управления.

Современное направление конвергенции работ в области создания интеллектуальных систем (см. *Голенков В.В..оОбучеИСкОС-2018ст*) требует разработки соответствующего программного обеспечения с элементами когнитивных способностей на основе семантически совместимых технологий искусственного интеллекта. Данное направление включает в себя также создание компьютерных систем, обеспечивающих интеллектуализацию процессов принятия аналитических управленческих решений, что напрямую связано с адаптацией процессов управления сложными динамическими системами (техническими объектами) в режиме реального времени, созданием семантически совместимых баз знаний в области анализа функционирования динамических систем и оптимизации функционирования сложных технических систем на их основе посредством создания открытого программного кода интеллектуальных компьютерных систем поддержки принятия решений.

Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0

Концепция Industry 4.0 была сформулирована в Германии в 2011 году. Она подразумевает создание и внедрение в производство так называемых киберфизических систем (КФС) и использование интернета вещей и услуг в производственных процессах (см. *Таберко В.В..ПроекПРПвК-2018ст*). При этом под КФС понимается совокупность интеллектуальных, легко интегрируемых физических компонентов со встроенными в них вычислительными ресурсами, тесно взаимодействующих между собой и отслеживающих изменения в состоянии внешнего мира.

Основные принципы концепции Industry 4.0:

- **Взаимодействие**. Возможность взаимодействия устройств, датчиков, людей посредством Интернета вещей (IoT), Интернета людей (IoP), Интернета услуг (IoS).
- **Виртуализация**. Означает способность киберфизической системы контролировать физические процессы. Данные сенсоров проецируются на модель предприятия, включающих состояние всех киберфизических систем. В случае возникновения нештатной ситуации должна быть возможность уведомить оператора, предоставив ему информацию по ее устранению и обеспечению безопасности, тем самым осуществляя поддержку принятия решений персоналом.
- **Децентрализация**. Растущая потребность в штучных партиях заказных продуктов увеличивает сложность централизованного управления производством. КФС могут иметь встроенные вычислительные модули, позволяющие им принимать решения самостоятельно и переадресовывать задачу управляющей системе только в случае необходимости. Несмотря на это, необходимо обеспечить контроль качества конечного продукта и прослеживаемость, что требует централизованного управления. К примеру, необходимые шаги производственного процесса могут быть закодированы в RFID-метках, что освобождает от необходимости централизованного управления данным аспектом производства малых партий продукта.
- **Анализ и реагирование в реальном времени**. С целью управления производством необходимо, чтобы данные с сенсоров постоянно собирались и анализировались в режиме реального времени. В случае отказа одной производственной установки, можно "перепоручить" ее задачу другой.

- **Ориентированность на услуги**. Услуги компаний, КФС и людей доступны в Интернете услуг и могут быть использованы другими участниками. Услуги могут предоставляться как внутри предприятия, так и другим предприятиям. КФС предоставляют свои услуги в виде веб-служб. Это позволит реализовать производство продукта путем комбинирования производственных операций в соответствии со спецификацией клиента, закодированной, например на RFID-метке.
- **Модульность**. Система должна быть гибкой, то есть легко адаптируемой к меняющимся требованиям (например, сезонным изменения в потреблении, изменению характеристик продукта или производства). Адаптация должна осуществляться заменой или расширением отдельных компонентов системы. Обеспечение совместимости компонентов требует наличия стандартизированных механизмов взаимодействия, позволяющих автоматически идентифицировать компоненты и включать их в интернет услуг. Полная совокупность производственной деятельности предприятия может быть описана в качестве набора технологических процессов производства с вероятностными характеристиками в соответствие с описанной онтологией данной предметной области.

Внедрение концепции Industry 4.0 на промышленных предприятиях сопровождается построением единой онтологической модели производства, которая является ядром комплексного информационного обслуживания предприятия (см. *Taberko V..Princ fEtDaUoS-2020art*). Потребность в комплексной автоматизации сложных процессов, требующих согласованной работы множества служб и технических средств, создает потребность в разработке систем, осуществляющих интеграцию вычислительных ресурсов и физико-технических процессов.

Для того чтобы обеспечить широкое применение технологий искусственного интеллекта в автоматизации предприятия, все корпоративные знания предприятия должны быть записаны на формальном языке представления знаний. Источниками таких знаний могут служить существующие описания работы предприятий в рамках принятых международных стандартов. В результате реализации онтологической модели предприятия возникает возможность интеграции и применения систем и подходов различной природы для автоматизации различных аспектов производственной деятельности.

Онтологическая модель производственной системы при этом включает в себя модели производственных процессов данного предприятия, которые можно представить в описанной в данной главе формализации.

Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS

В рамках решения задачи адаптивного управления в изложенной формализации может быть предложено несколько подходов:

- Прямое управление с эталонной моделью (последовательная схема управления), при котором путем обучения нейронной сети на базе существующих "оптимальных" сигналов системы управления (см. *Omidvar O..NeuraSfC-1997bk*), приводящих к формированию желаемых траекторий в фазовом пространстве системы, строится нейросетевой контроллер системы управления (*Smorodin V..Appli oNMfAC-2019art*). Данная схема нейроуправления является наиболее простой, но имеет недостатки в виде необходимости наличия репрезентативной выборки статистики работы существующего физического контроллера.
- Прямое инверсное управление (см. *White D.A..Handb oICNF-1992bk*), при котором нейроконтроллер обучается воспроизводить зависимость между управляющим сигналом в текущий момент времени и наблюдениями за объектом управления в следующий момент времени в процессе моделирования функционирования контура управления, такой подход показан в работах *Ivaniuk D.NeuroPIDCfaP-2013art Иванюк Д.С.НейроПОУ-2014cm* .
- Могут быть предложены схемы построения нейроконтроллеров на основе обучения с подкреплениям, в частности:
  - Управление с помощью нейросетевого контроллера, сформированного через процедуру обучения с подкреплением на задаче поиска в некотором ("геометрическом") смысле оптимальной траектории в фазовом пространстве системы управления в явном виде (см. *Hagan M..NeuraNfC-1999art White D.A..Handb oICNF-1992bk Smorodin V..Contr oaTCoPPB-2019art*).
  - Управление с помощью нейросетевого контроллера, сформированного через процедуру обучения с подкреплением на задаче неявного поиска оптимальной траектории путем максимизации определенной оценки качества управления R (см. *Smorodin V..AdaptCoRPS-2019art*, *Smorodin V..SoftwCfACoaP-2022art*).

Поскольку задача подбора структуры нейронной сети в каждом из случаев является сложной и трудноформализуемой, перспективным направлением является использование эволюционных методов для ее автоматизации (см. *Stanley K..EvolvNNtAT-2002art*).

Подпункт 7.7.1.2.4 Формализация управления технологическим циклом производства на основе применения ostis-систем

В соответствие с изложенной онтологией под технологическим циклом производства подразумевается последовательность действий и операций, в результате которых осуществляется производство готовой продукции.

Функциональное взаимодействие компонентов комплекса управления и работающего в режиме реального времени технологического цикла производства осуществляется на основе непрерывного мониторинга состояния оборудования и параметров управления с помощью регистров-индикаторов и технических средств сопряжения.

С целью обеспечить возможность интеграции различных моделей (включая нейросетевые) с другими интеллектуальными системами предлагается формализация системы принятия решений на базе Технологии OSTIS.

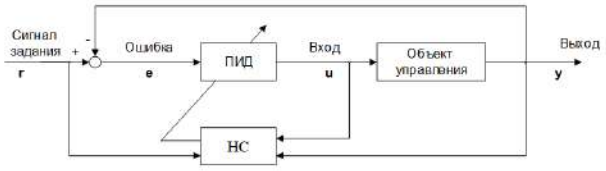
Подпункт 7.7.1.2.5 Принципы оптимизации процессов управления технологическим циклом с использованием нейросетевого моделирования

При управлении с эталонной моделью ставится задача построения такого нейросетевого контроллера таким образом, чтобы происходило приемлемое обобщение собранных массивов данных об управлении технологическим циклом. С точки зрения теории машинного обучения в этом случае будет решаться задача обучения с учителем, при которой собранные данные будут использованы в качестве пар эталонных векторов входов и выходов контроллера (управляющих сигналов контроллера). С точки зрения теории динамических систем фазовое пространство состояний системы управления ТП — среда, в которой на основании наблюдаемого состояния необходимо принимать решения о выборе текущего управления. И поэтому процесс обучения нейронной сети можно рассматривать как процесс запоминания корректных траекторий в фазовом пространстве, соответствующим оптимальным управлениям.

Процесс обучения при инверсной схеме является схожим с точки зрения теории машинного обучения, с той разницей, что источником формирования оптимизируемой функции потерь являются желаемые сигналы наблюдений за объектом управления.

Рисунок. Схема прямого управления технологической системой
=

Рисунок. Пример реализации схемы с инверсным управлением
=



Возможен подход при котором применяются методы обучения с подкреплением для построения эффективного контроллера системы управления. Методы обучения с подкреплением подразумевают исследовательское поведение агента в процессе построения контроллера, что может иметь позитивный эффект при решении задач со сложной структурой принятия управляющих решений.

Пусть задан некоторый функционал оценки качества управления R (качества выбора политики управления (выбора агентом (системой управления) действий) π), рассчитываемый на промежутке времени реализации вероятностного сетевого графика ТП $[0, t]$. (Данный функционал может "награждать", например, за снижение издержек производства, "штрафовать" за простой оборудования, возникновения отказов оборудования либо аварий).

Принятие решений в этой среде может осуществляться нейронной сетью (агент под управлением нейросетевого контроллера).

Совершение агентом некоторых действий в фазовом пространстве системы управления приводит к построению некоторой траектории (в данном контексте построение управления рассматривается как построение траектории в фазовом пространстве системы управления технологического процесса производства).

Фазовое пространство состояний системы управления ТП — среда, в которой на основании наблюдаемого состояния необходимо принимать решения о выборе текущего управления. Анализ фазового пространства состояний позволяет рассматривать задачу управления в геометрическом контексте.

Задача поиска оптимальной траектории в фазовом пространстве системы, таким образом, сводится к максимизации R (как в текущий момент, так и в будущем — то есть к контроллеру системы управления ТЦ предъявляется требование формирование политики выбора действий π, максимизируещей оценку качества управления R).

Можно выделить две группы алгоритмов обучения с подкреплением, используемые для решения сложных задач управления:
1. “value-based” — обучение контроллера оценке функции будущего вознаграждения;
2. “policy-based” — обучение контроллера распределению действий, приводящих к выбору оптимальной политики.

Применение методов обучения с подкреплением подразумевает создание среды, в которой агент совершает действия. Агент совершает действия на основания текущего набора наблюдений за средой, а по результату совершения действий и последовавшего за ним возможного изменения состояния среды, получает численную награду.

Средой, в которой действует агент, в контексте решения задач управления технологическим циклом является система управления технологическим циклом производства, которая делает доступным для агента наблюдение за сигналами регистров о состоянии оборудования и параметров управления. На основании решений агента система принятия решений формирует запросы на изменения параметров управления.

В контексте применения подходов к построению контроллера системы управления, основанных на обучении с подкреплением могут быть предложены следующие методы к организации среды, в которой действует и обучается агент:
1. использование наблюдений за функционированием физического технологического процесса, при котором имплементированы схемы сравнения действий физической системы управления с действиями, предложенными агентом, с оценкой их соответствия текущей ситуации и допустимостью
2. использование имитационного моделирования технологического процесса производства в качестве базы для построения среды для обучения с подкреплением

Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования

Процесс обучения нейронной сети заключается в поиске таких значений настраиваемых параметров сети (весовых коэффициентов связей между нейронами и уровней активации нейронов), чтобы она осуществляла правильное отображение. Обучение часто можно рассматривать как нелинейную оптимизационную задачу минимизации некоторой функции потерь относительно настраиваемых параметров сети. Форма этой функции определяется постановкой задачи, решаемой с применением нейронной сети.

При подходе, основанном на управлении с эталонной моделью, нейроконтроллер обучается с учителем. При этом используется функция потерь, оценивающая отклонение выходных сигналов сети от эталонных, и производится оптимизация данной функции (обычно градиентными методами).

Q-обучение является примером “value-based” алгоритма обучения с подкреплением, который может быть применен для поиска настраиваемых параметров модели при решении задач управления технологическим циклом.

Идея Q-обучения состоит в том, что агент в процессе обучения строит верную функцию Q оценки вознаграждения за следующее состояние, к которому может привести выбор некоторого управления (см. *Watkins C..QL-1992art*).

Функция вознаграждения при этом основана на оценке качества управления контроллера. Определение функции вознаграждения играет важную роль, определяя поведение агента, формируемое при обучении. Выбор функции вознаграждения позволяет выбрать для оптимизации те критерии поведения агента, которые пользователь системы считает важными.

В качестве аппроксиматора функции вознаграждения может быть использована нейронная сеть. В этом случае задачей обучения является поиск таких значений настраиваемых параметров нейросети, при которых приближенная функция Q будет достаточно близкой к оптимальной функции $Q*$ при данной политике выбора действий π (см. *Mnih V..HumanLCtDRL-2015art*):

$Q^*(s,a) = \mathbb{E}(r + \gamma \max_{a'} Q^*(s',a')|s,a)$

для которой выполняется уравнение Беллмана:

$Q(s,a) \approx Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a\right]$

При решении сложных практических задач агенту часто бывает недоступна полная информация о состоянии среды. В этой ситуации агент, который пользуется аппроксиматором Q, зависящим только от текущего наблюдаемого состояния среды может быть неэффективным при достаточно сложной структуре среды или наличии в ней распределенных во времени процессов, как связанных с действиями агента, так и независимыми от них. В описанной ситуации возможно использование в качестве аппроксиматора рекуррентной нейросети, обладающей внутренним состоянием (см. *Bakker B.ReinfLwLSTM-2001art*). LSTM-блоки (предложенные в работе *Hochreiter D.LongSTM-1997art*), при использовании в рекуррентной сети позволяют ей аппроксимировать сложные зависимости, растянутые на длительные периоды времени.

В policy-based методах вместо аппроксимации числовой функции, оценивающей возможные награды получаемые в окружении агентом за совершенные действия, происходит построение напрямую функции политики выбора действий, связывающей состояния с действиями агента. Имеет место параметризации политики выбора действий настраиваемыми параметрами модели, под управлением которой функционирует агент.

Численная функция (функция награды) при этом может быть использована для оптимизации политики относительно настраиваемых параметров, но не используется для выбора действий.

Стохастическая политика выбора действий возвращает распределение вероятностей возможных действий. Такие политики используются в частично наблюдаемых окружениях при наличии неопределенности.

Было показано, что для определенных классов задач методы, основанные на политиках сходятся быстрее чем value-based (Q-обучение), предпочтительны в пространствах выбора действий высокой размерности (см. *Sutton R.S..ReinfLaI-1998bk*). Гарантируется сходимость по крайней мере к локальному максимуму качества.

Политика π параметризована настраиваемыми параметрами θ.

$\pi_\theta(a|s) = P[a|s]$

Эта политика возвращает распределение действий a при наблюдаемом состоянии среды s.

С целью поиска настраиваемых параметров необходимо решить оптимизационную задачу максимизации функции оценки качества $J(\theta)$.

$J(\theta) = E_{\pi\theta}(\sum \gamma r)$

Правила обновления настраиваемых параметров на шаге t:

$\theta_{(t+1)} := \theta_t + \alpha \bigtriangledown J(\theta_t)$

Согласно Policy Gradient Theorem[5]

$\bigtriangledown E_{\pi\theta}(r(\tau)) = E_{\pi\theta}(r(\tau) \bigtriangledown log\pi_{\theta(\tau)})$

что может быть преобразовано как

$\bigtriangledown E_{\pi\theta}(r(\tau)) = E_{\pi\theta}(r(\tau)(\sum_{t=1}^{T} \bigtriangledown log\pi_\theta(a_t|s_t)))$

Алгоритм REINFORCE, предназначенный для изучения агентом политики выбора действий, приводящей к максимизации кумулятивных будущих наград, может быть сформулирован следующим образом (см. *Sutton R.S..ReinfLaI-1998bk*):

1. Инициализировать параметры политики θ
2. Сгенерировать эпизод взаимодействия агента со средой с $\{S_i\}$, $\{A_i\}$, $\{R_i\}$ — последовательностями наблюдаемых агентом состояний среды, выборов действий, полученных наград длинной T.
3. Для каждого шага t вычислить discounted reward $G_t := \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
4. Обновить параметры по правилу $\theta := \theta + \alpha\gamma^t G \bigtriangledown_\theta ln\pi(A_t|S_t,\theta)$
5. Повторить шаги 2-4 до сходимости.

Подпункт 7.7.1.2.7 Решатели задач ostis-системы адаптивного управления технологическим циклом производства

В рамках Технологии OSTIS решатели задач строятся на базе мультиагентного подхода. В соответствии с этим подходом решатель задачи строится как набор агентов, называемых sc-agents. Все такие агенты разделяют память и могут обмениваться данными через специальные семантические структуры (sc-texts). Важно отметить, что некоторые агенты могут быть неатомарными, то есть представленными в виде двух или более sc-агентов.

Полный решатель для задачи управления может быть представлен как разложение абстрактного неатомарного sc-агента.

abstract non-atomic sc-agent of cycle recommendation system

⇒ *decomposition of abstract sc-agent**:
{• *abstract sc-agent of interaction with the observation system*
• *abstract sc-agent of forming recommendations*
• *abstract sc-agent of forming requests*
}

1. abstract sc-agent of interaction with the observation system — предназначен для извлечения наблюдений из средств аппаратно-программного сопряжения в технологическом цикле, инициирует работу агента, ответственного за формирование рекомендаций.
2. abstract sc-agent of forming recommendations — на основании полученных наблюдений инициирует работу контроллера для получения рекомендаций по осуществлению управляющих воздействий.
3. abstract sc-agent of forming requests — на основании данных полученных от агента формирования рекомендаций, формирует запросы на изменения переменных управления в.т.п. посредством соответствующих программно-аппаратных средств сопряжения.

Построение контроллера, используемого в 2) может быть осуществлено на базе нейронных сетей.

Подпункт 7.7.1.2.8 Система адаптации управления

При разработке системы адаптации управления в рамках построения решателя задач ostis-системы предложенной структуры, необходимо реализовать агент для формирования рекомендаций на основании полученных наблюдений. В основе данного агента лежит контроллер, сформированный в соответствие с одним из алгоритмов, описанных выше.

Подпункт 7.7.1.2.9 Примеры реализации систем адаптации управления

В качестве примера можно рассмотреть задачу поиска оптимальной стратегии обслуживания оборудования технологического цикла производства (см. *Smorodin V..SoftwCfACoaP-2022art*).

Решатель такой задачи может быть представлен как разложение абстрактного неатомарного sc-агента.

abstract non-atomic sc-agent of cycle maintenance recommendation system
⇒ *decomposition of abstract sc-agent**:
{• *abstract sc-agent of interaction with the observation system*
• *abstract sc-agent of forming recommendations*
• *abstract sc-agent of forming maintenance requests*
}

1. abstract sc-agent of interaction with the observation system — предназначен для извлечения наблюдений из средств аппаратно-программного сопряжения в технологическом цикле, инициирует работу агента, ответственного за формирование рекомендаций.
2. abstract sc-agent of forming recommendations — на основании полученных наблюдений инициирует работу нейроконтроллера для получения рекомендаций по совершению операций обслуживания.
3. abstract sc-agent of forming maintenance requests — на основании данных полученных от агента формирования рекомендаций, формирует запросы на обслуживание для соответствующих программно-аппаратных средств сопряжения.

Для построения нейроконтроллера будет использовано обучение с подкреплением. В качестве среды для обучения с подкреплением будет использоваться имитационная модель технологического цикла производства.

Для реализации имитационной модели в описанной формализации используются:
- распределения длительности безотказной работы для оборудования i-го типа $\Phi^i_{1r}(\tau_{NOr})$ в режиме $r(M_i)$;
- распределения длительности восстановления (ремонта) оборудования i-го типа после отказа $\Phi^i_{2r}(\tau_{ROr})$;
- распределения длительности ликвидации аварии на оборудовании i-го типа $\Phi^i_{3r}(\tau_{em1})$;
- вероятности возникновения аварии при отказе i-го узла $\Phi^i_{4r}(C_{em1})$.

Имитационная модель функционирует в течение заданного интервала времени, перезапуская производственный цикл и, возможно, осуществляя перед этим запуском профилактические действия. Для принятия решения о проведении и содержании профилактических действий используется система управления технологическим циклом.

Система управления технологическим циклом производства делает доступным для агента наблюдение за временем безотказной работы всех узлов M_i.

Формирование функции вознаграждения включает такие компоненты как время непрерывной работы цикла(R_{nop}), суммарный объем затрат на обслуживание и ликвидацию отказов и аварий оборудования(R_{cost}), суммарное число

отказов оборудования (R_f), в том числе, приведшее к аварии (R_{fe}), суммарное число профилактик за цикл (R_{rep}). В процессе обучения модели значение функции вознаграждения формируется следующим образом во время единичного прохождения процедуры “профилактика-цикл”. Каждый из компонентов награды входит в уравнение с заданным весовым коэффициентом α_i, характеризующим его значимость.

$R = \alpha_1 R_{nop} + \alpha_2 R_{cost} + \alpha_3 R_f + \alpha_4 R_{fe} + \alpha_5 R_{rep}$

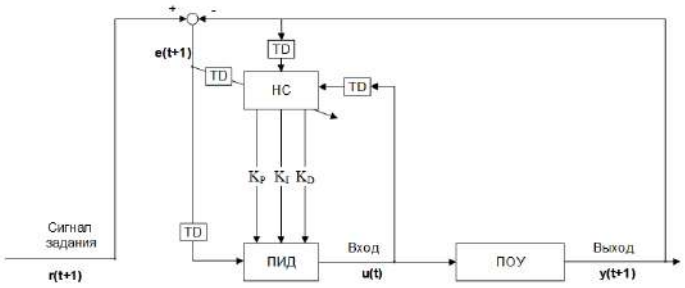
Для управления агентом использованы нейросети на базе МСП с блоком LSTM. В случае с policy gradient используется sigmoid, выход сети представляет собой распределение вероятностей действий. Структура сети:
1. Dense x64 ReLU
2. Dense x64 ReLU
3. LSTM x32 ReLU
4. Dense x6 softmax

В качестве другого примера решения задачи адаптивного управления, с использованием нейросетевых подходов, можно рассмотреть задачу стабилизации температуры в процессе функционирования пластинчатой пастеризационно-охладительной установки (ПОУ) (см. *Иванюк Д.С.НейроПОУ-2014ст*). ПОУ предназначена для тепловой обработки и охлаждения молочных продуктов в непрерывном тонкослойном закрытом потоке. Для автоматизации регулирования температурного режима в состав ПОУ входит система управления на базе промышленного контроллера. От применяемых алгоритмов управления напрямую зависит качество получаемой продукции.

Основными причинами, вызывающими колебания температуры t_{mn} нагревания молока являются непостоянство расхода G_m продукта, непостоянство температуры t_0 исходного молока, изменение расхода G_p пара, обусловленное колебания его давления P_{pr}, изменение коэффициента теплопередачи K вследствие отложения белка молока на теплопередающих поверхностях. Для стабилизации температуры t_{mn} нагревания молока в качестве управляющего воздействия в основном применяют расход пара G_p. Его регулируют посредством управляемого клапана.

Общая структура самонастраивающегося нейро-ПИД регулятора показана на рисунке ниже, где выходами нейронной сети — являются пропорциональный (K_P), интегральный (K_I) и дифференциальный (K_D) коэффициенты.

Рисунок. Общая структура нейро-ПИД регулятора
=



ПИД-регулятор в дискретном времени описывается следующим выражением

$u_n = u_{n-1} + K_p(e_n - e_{n-1}) + K_I e_n + K_D(e_n - 2e_{n-2} + e_{n-2})$

где K_P, K_I и K_D — пропорциональный, интегральный и дифференциальный коэффициенты соответственно, u_n определяет вход объекта управления в момент $t = nT_0$ и e_n — ошибка между желаемым значением выхода r_n и реальным, то есть

$e_n = r_n - y_n$

,T_0 определяет единичный интервал времени.

Для настройки K_P, K_I и K_D во время работы мы будем использовать трехслойный персептрон. Каждый слой состоит из N_1, N_2 и N_3 нейронов соответственно. Количество нейронов выбирается исходя из экспертного опыта и сложности объекта управления, N_3 равняется трем — количество коэффициентов ПИД. Для использования алгоритма обратного распространения ошибки мы должны выбрать функцию E, значение которой должно быть минимизировано. В качестве такой функции будет выступать ошибка управления пев момент времени nT_0.

$E_n = 1/2e_n^2$

Для накопления ошибок сохраняем полученные ранее данные — $E_{n-p}, ..., E_{n-2}, E_{n-1}, E_n$, где p определяет количество сохраненных ранее образов, используемых для обучения сети.

§ 7.7.2. Построение умных предприятий рецептурного производства с помощью ostis-систем

⇒ *подраздел**:
- *Пункт 7.7.2.1. Проблемы проектирования умных предприятий*
- *Пункт 7.7.2.2. Подходы к проектированию и стандартизации умных предприятий рецептурного производства с помощью ostis-систем*

Пункт 7.7.2.1. Проблемы проектирования умных предприятий

⇒ *подраздел**:
- *Подпункт 7.7.2.1.1 Проблемы разработки систем комплексной автоматизации*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

Подпункт 7.7.2.1.1 Проблемы разработки систем комплексной автоматизации

Существующие средства автоматизации деятельности предприятия имеют высокую стоимость, трудны в освоении и адаптации к конкретному производству. Как правило, такие средства, с одной стороны, жестко ориентированы на решение некоторого ограниченного класса задач, с другой стороны, разработчики стремятся сделать такого рода средства как можно более универсальными, наращивая их частными решениями, что приводит к сложности и громоздкости таких систем. Вследствие подобного подхода к наращиванию функционала существующие средства автоматизации деятельности предприятия имеют низкий уровень гибкости (возможности внесения изменений), что приводит к существенным накладным расходам при адаптации таких средств к новым требованиям. Как правило, внесение изменений в указанные средства требует вмешательства разработчиков (часто сторонних с точки зрения предприятия), что влечет значительные временные и финансовые затраты. Как следствие указанных проблем, далеко не всякое предприятие может обеспечить высокий уровень автоматизации своей деятельности, даже в случае наличия на рынке подходящих решений (см. *Golenkov V.V..OntolDoBME-2017art*, *Голенков В.В..ПроекПРПнОО-2017cm*). Отсутствие общих унифицированных моделей и средств построения систем автоматизации деятельности предприятия приводит к большому количеству дублирований аналогичных решений как в рамках различных предприятий, так и в рамках разных подразделений одного предприятия. При этом часто возникает ситуация, когда некоторые частные системы, решающие различные задачи в рамках одного предприятия, оказываются несовместимыми между собой, что приводит к дополнительным расходам на реализацию механизмов согласования, например, преобразование форматов данных. Отсутствие такого рода моделей препятствует дальнейшему повышению уровня автоматизации предприятия, в частности, в области автоматизации принятия решений в нештатных ситуациях, прогнозирования дальнейшего развития событий.

Подпункт 7.7.2.1.2 Требования, предъявляемые к стандартизации предприятий

Средства автоматизации предприятия должны оперативно и с минимальными затратами времени сотрудников адаптироваться к любым изменениям самого производства — к расширению или сокращению объемов производства, изменениям номенклатуры производства, изменению используемого оборудования, изменению общей структуры производства, изменению взаимодействия с поставщиками и потребителями, к изменению нормативно-правовых актов (включая стандарты), к различного рода непредвиденным обстоятельствам. Адаптация средств автоматизации предприятия ко всем видам изменений самого предприятия и всем аспектам его взаимодействия с внешней средой требует внесения изменений в модель предприятия, полностью отражающую текущее состояние его деятельности. Средства автоматизации предприятия должны быть гибкими не только для оперативной адаптации к реконфигурации производства, но и для оперативного внесения изменений в сами средства автоматизации в направлении их постоянного совершенствования. Здесь существенным является не только снижение трудоемкости повышения уровня автоматизации, но и поддержка высоких темпов повышения уровня автоматизации, а также четко продуманный переходный процесс от одного уровня автоматизации к следующему, в ходе которого одновременно используется и устаревший вариант, и новый. Эксплуатация системы автоматизации предприятия текущего уровня и перманентный процесс повышения этого уровня требуют согласованного и квалифицированного взаимодействия сотрудников предприятия. Основой такого взаимодействия является хорошо структурированная, достаточно полная и оперативно актуализируемая модель предприятия, отражающая все аспекты текущего состояния структуры и деятельности предприятия, а также планы его развития. Такого рода комплексная модель называется знаниями предприятия, которыми надо управлять (добывать, хранить, модернизировать, распространять и так далее). Повышение уровня автоматизации предприятия предполагает существенное расширение числа

автоматически или автоматизированно решаемых задач, а это, в свою очередь, приводит к автоматизации решения интеллектуальных задач, то есть к использованию технологий искусственного интеллекта. К числу интеллектуальных задач, решаемых на предприятии, можно отнести:

1. анализ производственных ситуаций (в том числе нештатных);
2. принятие решений на различных уровнях;
3. планирование поведения в сложных обстоятельствах;
4. генерация, актуализация документации;
5. обучение новых и повышение квалификации действующих сотрудников;
6. и так далее.

Для того, чтобы обеспечить широкое применение технологий искусственного интеллекта в автоматизации предприятия, все корпоративные знания предприятия должны быть записаны на формальном языке представления знаний. При этом указанный язык должен быть удобен не только для использования в интеллектуальных компьютерных системах, но и для использования всеми сотрудниками предприятия.

Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности

Анализ работ в данной области позволил сформулировать наиболее важные и общие проблемы, связанные с разработкой и применением современных стандартов в различных областях:

- Прежде всего сложность ведения самих стандартов из-за дублирования информации, особенно сложность изменения терминологии.
- Дублирующаяся информация в документации, описывающей стандарт.
- Проблемы интернационализации стандартов – перевод стандарта на несколько языков фактически требует поддержки и координации независимых версий стандарта на разных языках.
- В результате несоответствия форматов разных стандартов. В результате автоматизация процесса разработки и применения стандартов усложняется.
- Неудобство использования стандарта, особенно сложность поиска нужной информации. Как следствие, сложность изучения стандартов.
- Сложность автоматизации проверки соответствия объекта или процесса требованиям определенного стандарта.
- etc.

Эти проблемы в основном связаны с представлением стандартов. Наиболее перспективным подходом к решению этих задач является преобразование каждого конкретного стандарта в базу знаний, в основе которой лежит набор соответствующих этому стандарту онтологий. Такой подход позволяет значительно автоматизировать процессы разработки стандарта и его применения.

В качестве примера рассмотрим стандарт ***ISA-88*** (базовый стандарт для рецептурного производства, см. *ISA88BC-2022el*). Хотя этот стандарт широко используется американскими и европейскими компаниями и активно внедряется на территории Республики Беларусь, он имеет ряд недостатков, перечисленных ниже. Опыт автора со стандартами ISA-88 и ISA-95 выявил следующие проблемы, связанные с версиями стандарта (см. *Taberko V..OntolAtBEwI-2022art*):

1. Американская версия стандарта – *ANSI/ISA-88.00.01-2010* – была обновлена и находится в третьем издании 2010 года;
2. *ISA-88.00.02-2001* — содержит структуры данных и рекомендации для языков;
3. *ANSI/ISA-TR88.00.02-2015* – описывает пример реализации ANSI/ISA-88.00.01;
4. *ISA-88.00.03-2003* – действие, описывающее использование общих рецептов сайта внутри и между компаниями;
5. *ISA-TR88.0.03-1996* – показывает возможные форматы представления процедур рецепта;
6. *ANSI/ISA-88.00.04-2006* – структура записей серийного производства;
7. *ISA-TR88.95.01-2008* – объясняет совместное использование ISA-88 и ISA-95;
8. В то же время европейская версия, утвержденная в 1997 году – *IEC 61512-1* – основана на более старой версии *ISA-88.01-1995*;
9. Русская версия стандарта – *ГОСТ Р МЭК 61512-1-2016* – идентична *МЭК 61512-1*, то есть также устарела. Также вызывает ряд вопросов, связанных с не очень удачным переводом оригинальных английских терминов на русский язык.

Другим стандартом, часто используемым в контексте Индустрии 4.0, является ***ISA-95*** (см. *ISA95-2022el*). ***ISA-95*** – это отраслевой стандарт для описания систем управления высокого уровня. Его основная цель — упростить разработку таких систем, абстрагироваться от аппаратной реализации и предоставить единый интерфейс для взаимодействия со слоями ERP и MES. Состоит из следующих частей (см. *Taberko V..OntolAtBEwI-2022art*):

1. *ANSI/ISA-95.00.01-2000*, Часть 1: “Модели и терминология” — состоит из стандартной терминологии и объектных моделей, которые можно использовать для определения того, какая информация подлежит обмену;

2. *ANSI/ISA-95.00.02-2001*, Часть 2: “Атрибуты объектной модели” — он состоит из атрибутов для каждого объекта, определенного в части 1. Объекты и атрибуты могут использоваться для обмена информацией между различными системами, а также могут использоваться в качестве основы для реляционных баз данных;
3. *ANSI/ISA-95.00.03-2005*, Часть 3: “Операционные модели управления производством” — основное внимание уделяется функциям и действиям Уровня 3 (Производство/MES);
4. *ISA-95.00.04* Часть 4: “Объектные модели и атрибуты для операций управления производством”. Комитет SP95 все еще разрабатывает эту часть ISA-95. Эта техническая спецификация определяет объектную модель, которая определяет обмен информацией между действиями MES (определено в Части 3 стандарта ISA-95). Модель и атрибуты Части 4 составляют основу для разработки и внедрения стандартов интерфейса, обеспечивая гибкий поток сотрудничества и обмена информацией между различными видами деятельности MES;
5. *ISA-95.00.05* Часть 5: “Транзакции между бизнесом и производством (B2M транзакции)”. Часть 5 стандарта ISA-95 все еще находится в разработке. Эта техническая спецификация определяет операции между рабочими местами и структурами автоматизации производства, которые могут использоваться вместе с моделями элементов частей 1 и 2. Операции объединяют и упорядочивают описанные производственные элементы и действия. в рамках предыдущей части стандарта.Такие операции возникают в любом отношении внутри организации, однако внимание данной технической спецификации уделяется взаимодействию между организацией и системой управления.
6. *ISA-95.00.06* Часть 6: “Транзакции между производственными операциями”.
7. *ISA-95.00.07* Часть 7: “Модель сервисных сообщений”.
8. *ISA-95.00.08* Часть 8: “Профили обмена информацией”

Модели помогают определить границы между бизнес-системами и системами управления. Они помогают ответить на вопросы о том, какие функции могут выполнять какие задачи и какой информацией необходимо обмениваться между приложениями.

Первый этап построения модели цифрового двойника требует встраивания данных на более низких уровнях производства, таких как производственные процессы и оборудование (см. *Lutska N..OntolMoDTiM-2022art*). Схема производства P&ID служит источником этих данных. Поэтому стандарт ISA 5.1 (см. *ISA51-2022el*) должен работать со схемой P&ID и широко используется в системах управления наряду со стандартом ISA 88 для полного описания нижних уровней производства. Этот стандарт полезен, когда требуется ссылка на оборудование в химической, нефтяной, энергетической, кондиционирующей, металлообрабатывающей и многих других отраслях промышленности. Стандарт позволяет любому человеку с достаточным уровнем знаний о предприятии читать блок-схемы, чтобы понять, как измерять и контролировать процесс, не вдаваясь в детали приборов или знаний эксперта по приборам. Он предназначен для предоставления достаточной информации, чтобы SA5.1 Целью настоящего стандарта было установить последовательный метод наименования приборов и контрольно-измерительных систем, используемых для измерения и контроля. Для этого представлена система обозначений, включающая символы и идентификационные коды. Последним выпуском подкомитета ISA5.1 является обновленный *ISA-5.1-2022*, “Символы приборов и идентификация”.

Обучение — это простой способ достичь этих стандартов. Международное общество автоматизации (*ISA*) — некоммерческая профессиональная ассоциация и признанный лидер в области обучения автоматизации и управлению, занимающийся подготовкой рабочей силы к технологическим изменениям и отраслевым вызовам. Однако цена относительно высока, около 1000 долларов на человека в день. Для 2 человек это 10000$ за обычный курс на 5 дней. Для одних стран это доступно, для других нет.

Принимаются во внимание различные установленные процедурные требования различных организаций, но это делается путем предоставления альтернативных методов символики, если это не противоречит целям стандарта. Существует множество вариантов добавления информации или упрощения символа при желании.

Эти и другие стандарты сейчас распространяются в виде документов, неудобных для автоматизированной обработки и, как отмечалось выше, сильно зависят от языка, на котором написан каждый документ.

Пункт 7.7.2.2. Подходы к проектированию и стандартизации умных предприятий рецептурного производства с помощью ostis-систем

⇒ *подраздел**:
- *Подпункт 7.7.2.2.1 Подходы к автоматизации предприятий*
- *Подпункт 7.7.2.2.2 Предлагаемый подход к автоматизации предприятий*
- *Подпункт 7.7.2.2.3 Принципы формализации стандартов рецептурного производства*
- *Подпункт 7.7.2.2.4 Формализация стандарта ISA-88*

Подпункт 7.7.2.2.1 Подходы к автоматизации предприятий

В настоящее время существует ряд подходов, ориентированных на повышение уровня автоматизации и гибкости предприятий различного рода. Рассмотрим те из них, которые оказали влияние на развитие подхода, предлагаемого в данной работе (см. *Golenkov V.V..OntolDoBME-2017art*, *Голенков В.В..ПроекПРПнОО-2017ст*).

- Онтологические модели предприятий. Подход к проектированию различного рода систем на основе онтологических моделей широко используется в настоящее время, при этом в особую область исследований выделяют "онтологии предприятия". Суть предлагаемых подходов состоит в построении онтологий, описывающих деятельность того или иного предприятия или его подразделений. Недостатками данных моделей являются отсутствие унификации представления различных видов знаний предприятия, отсутствие единого подхода к выделению и формированию онтологий, отсутствие единого подхода к построению иерархии онтологий, что ограничивает возможность построения комплексной взаимосвязанной системы онтологий.
- Модели управления знаниями предприятий. Управление знаниями в организации — это систематический процесс идентификации, использования и передачи информации, знаний, которые люди могут создавать, совершенствовать и применять. Это процесс, в ходе которого организация генерирует знания, накапливает их и использует в интересах получения конкурентных преимуществ. В настоящее время управление знаниями предприятия реализуется в виде систем управления знаниями. Наиболее актуальным направлением в формализации процесса накопления и управления знаниями предприятия является применение онтологического подхода к построению моделей такого рода процессов.
- Модели ситуационного управления. Термин "ситуационное управление" впервые появился в работах Д.А. Поспелова. Это направление получило дальнейшее развитие, а в ряде новых работ показано применение в реализации методов ситуационного управления онтологического подхода. Таким образом, модели и методы ситуационного управления могут быть использованы при построении онтологической модели предприятия с целью повышения эффективности разрабатываемых решений в конкретных производственных ситуациях.
- Многоагентные модели предприятий. В настоящее время многоагентная модель широко применяется при проектировании систем автоматизации производства на различных уровнях. Удобство такого подхода и широта его использования обусловлены схожестью многоагентной модели с реальными процессами, происходящими на предприятии. Действительно, в классической многоагентной системе под агентом понимается некий субъект, как правило, активный и способный взаимодействовать с окружающей средой. Будучи объединенными в коллективы, такие агенты способны решать задачи гораздо более сложные, чем мог бы решить один агент. К достоинствам многоагентного подхода можно отнести возможность построения на его основе распределѐнных многоуровневых систем. Наиболее очевидной интерпретацией такого рода модели в применении к конкретному предприятию является рассмотрение его работников как агентов, каждый из которых способен решать определенный класс задач, вынужденных координировать свои действия для достижения общей коллективной цели. С учѐтом иерархии структурных подразделений конкретной организации могут быть выделены и уровни иерархии агентов, соответствующие отделам или цехам.
- Модели реинжиниринга бизнес-процессов предприятий. Реинжиниринг бизнеспроцессов — это фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов предприятий для достижения резких, скачкообразных улучшений в основных актуальных показателях их деятельности: стоимость, качество, услуги и темпы. Он базируется на понятиях будущего образа фирмы и модели бизнеса, раскрываемых в. Для того, чтобы повысить эффективность реинжиниринга, необходимо обеспечить возможность построения формальных моделей, описывающих предприятие на разных уровнях детализации, и обеспечить унификацию таких моделей, их интегрируемость и иерархичность.

Основной недостаток всех приведенных выше моделей заключается в том, что ни одна из них не обладает достаточной полнотой, и для наиболее адекватного соответствия реальному предприятию его модель должна быть результатом интеграции всех этих моделей.

Подпункт 7.7.2.2.2 Предлагаемый подход к автоматизации предприятий

В основе предлагаемого подхода к решению указанных проблем лежат следующие принципы.

- Предприятие рассматривается как распределенная, интеллектуальная социотехническая система, в основе которой лежит хорошо структурированная общая база знаний предприятия.
- В рамках базы знаний предприятия интегрируются все вышеуказанные модели.
- Предприятие рассматривается как иерархическая многоагентная система. В качестве агентов выступают как сотрудники предприятия, так и программные (программно-аппаратные) агенты. Иерархичность многоагентной системы означает то, что агенты могут быть неатомарными, то есть коллективами взаимодействующих между собой агентов, причем такая структура может быть многократно вложенной.
- Весь комплекс средств (как информационных, так и материальных), обеспечивающих деятельность предприятия, оформляется в виде интегрированной распределенной интеллектуальной системы, которую будем

называть интеллектуальной корпоративной системой. Основными пользователями этой системы являются сотрудники предприятия.
- Проектирование онтологической модели предприятия сводится к проектированию онтологической модели его интеллектуальной корпоративной системы, которая далее может интерпретироваться имеющимся набором материальных ресурсов. При этом онтологическая модель предприятия является и объектом, и результатом проектирования.

Для реализации корпоративной системы предприятия предлагается использовать технологию OSTIS. Из этого следует:
- в качестве основы для представления знаний используется унифицированный, универсальный язык представления — SC-код;
- разработка системы сводится к разработке еѐ модели, описанной средствами SC-кода (sc-модели), которая затем интерпретируется одной из платформ интерпретации;
- база знаний имеет иерархическую структуру, позволяющую рассматривать хранимые знания на различных уровнях детализации (прежде всего это иерархия предметных областей (ПрО) и соответствующих им онтологий;
- частью технологии являются средства коллективного проектирования баз знаний, средства проектирования машин обработки знаний и их компонентов;
- модель обработки знаний основана на многоагентном подходе, позволяющем строить параллельные асинхронные машины обработки знаний, интегрировать различные частные модели обработки в рамках одной системы;
- все агенты взаимодействуют исключительно посредством общей памяти, хранящей конструкции SC-кода (sc-памяти); такой подход позволяет обеспечить гибкость системы и возможность параллельного решения различных задач;
- для разработки программ агентов используется внутренний параллельный Язык SCP, тексты которого также представлены в SC-коде, что позволяет обеспечить платформенную независимость таких агентов.

На данном этапе работы основное внимание уделено решению задачи разработки онтологической модели базы знаний, в частности — построению набора онтологий ПрО, описывающих содержание основных стандартов. Формальное представление стандартов является основой для согласования всех ключевых аспектов деятельности предприятия и построения общей онтологической модели всего предприятия в целом и отдельных его компонентов.

Подпункт 7.7.2.2.3 Принципы формализации стандартов рецептурного производства

Основой онтологического подхода к проектированию предприятия является формализация стандартов. Каждый стандарт рассматривается как онтология соответствующей ПрО, являющаяся основой для автоматизированного решения ряда задач, включая информационное обслуживание сотрудников, формальную оценку соответствия предприятия этим стандартам и так далее. Одной из важных проблем внедрения стандарта на предприятии является возможность неоднозначной трактовки некоторых положений стандарта, а также необходимость постоянной коррекции такой трактовки с целью приближения еѐ к смыслу оригинала. Кроме того, существуют особенности применения стандарта на каждом предприятии, необходимость актуализации используемого стандарта (так как любой стандарт постоянно эволюционирует), с последующим внесением изменений в структуру и организацию деятельности предприятия для обеспечения соответствия стандарту. Одним из путей решения такого рода проблем является построение его формальной семантической модели, которая могла бы одинаково интерпретироваться как компьютерной системой, так и человеком. Формальное семантическое представление стандарта позволяет без внесения каких-либо изменений в структуру такого представления дополнять его различного рода дидактической информацией (примерами, пояснениями и так далее), которая способствует пониманию стандарта сотрудниками предприятия. Кроме того, формальное семантическое представление стандарта обеспечивает значительное упрощение внесения изменений в такое представление стандарта, которые могут быть вызваны либо уточнением трактовки этого стандарта, либо эволюцией самого стандарта (его исходного документа). Построение формальной модели стандарта сводится к построению интегрированной формальной онтологии, специфицирующей соответствующую ПрО. Для этого необходимо отобразить структуру и содержание исходного текста стандарта на иерархию ПрО и соответствующих им онтологий. Использование онтологического подхода позволяет путѐм добавления интеллектуальных агентов построить на его основе интеллектуальную справочную систему, предоставляющую широкий спектр информационных услуг пользователям, в том числе способную отвечать на широкий спектр вопросов, ответ на которые может быть в явном виде не представлен в тексте стандарта, или поиск его в таком тексте затруднителен.

Подпункт 7.7.2.2.4 Формализация стандарта ISA-88

Структурно исходный документ первой части стандарта ***ISA-88*** (см. *ISA88BC-2022el*) состоит из 6 разделов:

- область применения стандарта (Scope of the standard);
- нормативные ссылки (Normative References);
- термины и определения (Definitions);
- процессы серийного производства и оборудование (Batch Processes and Equipment);
- понятия, используемые при управлении серийным производством (Batch Control Concepts);
- действия и функции процесса управления серийным производством (Batch Control Activities and Functions)

Отобразим структуру документа на иерархию ПрО и соответствующих им онтологий. Стандарт ISA-88 в целом соответствует ПрО предприятий рецептурного производства. Первые два раздела специфицируют документ стандарта и непосредственно к описанию ПрО рецептурных производств не относятся и, соответственно, не отображаются на иерархию ПрО и онтологий. Раздел 3 соответствует терминологической онтологии и логической онтологии ПрО предприятий рецептурного производства. Подраздел 4.1 соответствует ПрО процессных моделей рецептурных производств. Остальные подразделы четвертого раздела соответствуют ПрО физических моделей рецептурных производств. Раздел 5 соответствует ПрО моделей процедурного управления. Раздел 6 соответствует ПрО деятельности по управлению рецептурным производством. Запишем иерархию ПрО базового уровня в формальном виде на языке SCn:

ПрО предприятий рецептурного производства
- ⊂ *ПрО физических моделей рецептурных производств*
- ⊂ *ПрО процессных моделей рецептурных производств*
- ⊂ *ПрО моделей процедурного управления оборудованием рецептурных производств*
- ⊂ *ПрО деятельности по управлению рецептурным производством*

Заключение к Главе 7.7.

В настоящее время реализуются различные подходы при решении научных и прикладных задач по управлению роботизированными производствами в условиях наличия случайных внешних воздействий на объект управления.

В данном разделе предложена реализация процедуры синтеза обратных связей по управлению технологическим циклом производства на основе его формализации в рамках ostis-систем и построения моделей искусственных нейронных сетей для решения прикладных задач оптимизации управления сложными технологическими комплексами.

Полученные результаты дают возможность разработки гибридных интеллектуальных компьютерных систем, предназначенных для решения широкого класса прикладных задач адаптации управления технологическими объектами в режиме реального времени.

Глава 7.8.

Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения

⇒ *авторы**:
- *Самодумкин С. А.*
- *Зотов Н. В.*

⇒ *аннотация**:
[Глава посвящена частной технологии проектирования *интеллектуальных геоинформационных систем*, построенных по принципам *Технологии OSTIS. интероперабельность геоинформационных систем*, построенных по предлагаемой технологии, обеспечивается за счет перехода от карты к семантическому описанию элементов карты.]

⇒ *подраздел**:
- *§ 7.8.1. Требования, предъявляемые к интеллектуальным геоинформационным системам нового поколения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*
- *§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах*
- *§ 7.8.4. Решатель задач интеллектуальной геоинформационной ostis-системы*
- *§ 7.8.5. Картографический интерфейс интеллектуальной геоинформационной ostis-системы*
- *§ 7.8.6. Многократно используемые компоненты интеллектуальных геоинформационных ostis-систем*
- *§ 7.8.7. Средства автоматизации проектирования интеллектуальных геоинформационных ostis-систем*

⇒ *ключевой знак**:
- *Технология проектирования интеллектуальных геоинформационных систем*
- *Язык карт для ostis-систем*

⇒ *ключевое понятие**:
- *геоинформационная система*
- *интеллектуальная геоинформационная система*
- *задача геоинформационной системы*
- *интеллектуальная геоинформационная ostis-система*
- *база знаний интеллектуальной геоинформационной ostis-системы*
- *геоонтология*
- *объект местности*
- *геосемантическая характеристика объекта местности*
- *пространственное отношение*
- *решатель задач интеллектуальной геоинформационной ostis-системы*
- *картографический интерфейс*
- *картографический интерфейс интеллектуальной геоинформационной ostis-системы*

⇒ *ключевой параметр**:
- *класс объектов местности^*

⇒ *библиографическая ссылка**:
- *Крючков А.Н..ИнтелТвГС-2006кн*
- *Абламейко С.В..ГеогрИССЦК-2000ст*
- *Ивакин Я.А.МетодИПГСнОО-2009дс*
- *Белякова М.Л.ИнтелГСдУИТ-2016кн*
- *Губаревич А.В..ОнтолПИСвОИ-2017ст*
- *Губаревич А.В..СтрукБЗвИСпИ-2018ст*
- *Блискавицкий А.А.КонцеПГИСиУ-2012кн*
- *Блискавицкий А.А.СеманГОФГиИ-2014ст*
- *Ни Y.GeospS-2019art*
- *Janowicz K..GeospSaLSD-2012art*
- *ЦифроКМИО-2007ст*
- *Баталов Р.Н..ГеопрИДЗвИК-2021ст*
- *Березко А..ИнтелГИС-2009ст*

- *Журавков М.А..ГИСте пДПИСКГИМ-2004кн*
- *Глотов А.А.ПримеГТдФС-2014ст*
- *Глотов А.А.ИнтелГСПиН-2015ст*
- *Дулин С.К..ПроблОСГиСП-2016ст*
- *Кикоть А.В.СравнМиМИО-2020ст*
- *Кузнецов К.А..ПублиДоООП-2012ст*
- *Орехова Д.А.Разра иИМДП-2013дс*
- *Самодумкин С.А..СеманТКПИ-2011ст*
- *Самодумкин С.А.ИнтелГС-2012ст*
- *Samodumkin S.A.SemanToIGSD-2019art*
- *Samodumkin S.SemanToIGSD-2019art*
- *Samodumkin S.Next-IGS-2022art*
- *СистеООАД-2017кн*
- *Соколов М.С.Модел иАИОиА-2012дс*
- *Атаева О.М..СредаИПДГ-2011ст*
- *Шпаков М.В.РазраИГнОН-2004дс*
- *Янкелевич С.С..КонцеНВКОнЗ-2019ст*

Введение в Главу 7.8.

Современные программно-технологические комплексы *геоинформационных систем* очень эффективны, но сложны в освоении и применении, поэтому требуют специальной профессиональной подготовки конечных пользователей. Для внедрения систем геопространственного назначения в различные области знаний и сферы применения необходимо, чтобы специалисты различного вида деятельности без особых сложностей и дополнительного обучения могли решать характерные для *геоинформационных систем* задачи. Для этого необходим переход от традиционных геоинформационных систем к геоинформационным системам нового поколения, имеющим удобный *пользовательский интерфейс*.

Целью данной главы является создание комплексной технологии проектирования интеллектуальных геоинформационных систем нового поколения по принципам, лежащим в основе *Технологии OSTIS*.

§ 7.8.1. Требования, предъявляемые к интеллектуальным геоинформационным системам нового поколения

Согласно общепринятому определению *геоинформационная система — программная компьютерная система*, обеспечивающая ввод, манипулирование, анализ и вывод пространственно-соотнесенных данных (геоданных) о территории, социальных и природных явлениях при решении задач, связанных с инвентаризацией, анализом, моделированием, прогнозированием и управлением окружающей средой и территориальной организацией общества (см. *Крючков А.Н..ИнтелТвГС-2006кн*).

Следовательно, из самого определение *геоинформационной системы* вытекает необходимость реализации интеллектуальных задач.

задача геоинформационной системы
⇒ *разбиение**:
{• *задача анализа в геоинформационной системе*
• *задача моделирования в геоинформационной системе*
• *задача прогнозирования в геоинформационной системе*
• *задача управления в геоинформационной системе*
}

Все указанные задачи являются интеллектуальными и требуют поддержки принятия решения при их реализации.

В отличие от других классов *информационных систем*, в *геоинформационных системах* основным объектом исследования являются знания и данные об *объектах местности*, которые рассматриваются не только как пространственные данные и знания, но и являются интеграционной основой для различных *предметных областей*. При этом формализация таких *знаний* и их представление в *базах знаний интеллектуальных систем* требует установления отношений для описания свойств и закономерностей, присущих рассматриваемой *предметной области* и использующей *объекты местности*, установления геометрических характеристик, способных осуществить

привязку *объектов местности*, а также учитывает темпоральный (временной) характер существование *объектов местности*, что позволяет осуществить ретроспективный анализ. С учетом того, что *интеллектуальные системы* предназначены для удовлетворения *информационной потребности пользователей*, данный факт способствует расширению *предметных областей* и добавления новых функциональных возможностей в рамках предлагаемой частной *Технологии проектирования интеллектуальных геоинформационных систем*.

интеллектуальная геоинформационная система
:= [информационная система, основным объектом исследования которой являются знания и данные об объектах местности, выступающие интеграционной основой для решения прикладных задач в различных предметных областях]
⊃ *интеллектуальная геоинформационная ostis-система*
⊂ *ostis-система*
:= [интеллектуальная геоинформационная система, разработанная по принципам Технологии OSTIS]
⇒ *обобщенная декомпозиция**:
{• *база знаний интеллектуальной геоинформационной ostis-системы*
• *решатель задач интеллектуальной геоинформационной ostis-системы*
• *картографический интерфейс интеллектуальной геоинформационной ostis-системы*
}

Важным моментом, снижающим с одной стороны срок разработки *интеллектуальных систем*, а с другой — повышающим функциональные возможности *интеллектуальных систем*, использующих знания об *объектах местности*, является наличие технологии проектирования таких систем. При этом *Технология проектирования интеллектуальных геоинформационных систем* должна быть ориентирована на многократное использование функциональных компонентов системы с целью сокращения сроков проектирования и разработки прикладных систем. Таким образом, речь идет о создании частной *Технологии проектирования интеллектуальных геоинформационных систем*. В связи среди актуальных задач можно выделить следующие:

- проектирование пространственных онтологий и на основе их решение проблемы *семантической совместимости знаний предметных областей*;
- решение задачи управления метаданными и совершенствования поиска, доступа и обмена в условиях растущих объемов пространственной информации и сервисов, предоставляемых многочисленными источниками *геоинформации*;
- осуществление вывода знаний с использованием пространственной и тематической информации как составляющих *знаний объектов местности* с использованием *Языка вопросов* (см. *Главу 3.4. Язык вопросов для ostis-систем*);
- внедрение *картографического интерфейса* в *интеллектуальные ostis-системы* как естественного для человека способа представления информации об *объектах местности*.

Постоянная эволюция моделей и средств онтологического описания предметных областей, использующих пространственные и темпоральные компоненты, их неоднородность и неоднозначность, ставит новые задачи с точки зрения взаимодействия, интеграции и обеспечения совместимости различных прикладных систем за счет:

- интеграции *предметных областей* и соответствующих им онтологий (вертикальный уровень),
- расширения функциональных возможностей систем при помощи повторно используемых компонентов этих систем (горизонтальный уровень), в частности, проектирование компонентов для новых территорий или в новом временном интервале.

С целью выполнения предъявленных требований предлагается рассматривать карту как *информационную конструкцию*, элементами которой являются *объекты местности*. Для этого предложены:

- *Предметная область и онтология объектов местности*;
- *Спецификация Языка карт для ostis-систем*;

Переход от карт к их *смыслу* осуществляется на основе:

- формального описания *Синтаксиса Языка карт для ostis-систем*;
- формального описания *Денотационной семантики Языка карт для ostis-систем*.

При этом *семантическая совместимость геоинформационных систем* и их компонентов обеспечивается благодаря общей для них онтологии *объектов местности*, которая необходима для интероперабельности *геоинформационных систем* различного назначения и их компонентов. А *интероперабельность геоинформационных систем* обеспечивается за счет перехода от карты к семантическому описанию элементов карты, то есть *объектов местности* и связей (*пространственных отношений*) между ними.

Наличие данных обстоятельств определяет существование научно-технической проблемы *интеллектуализации геоинформационных систем* и создание *Технологии проектирования интеллектуальных геоинформационных систем*, в основе которых лежат *принципы проектирования ostis-систем*.

§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами

Одним из направлений повышения эффективности использования информационно-вычислительных средств является *интеллектуализация геоинформационных систем*.

интеллектуализация геоинформационных систем предполагает:
- возможность общения конечного пользователя с системой на *Языке вопросов* (см. *Главу 3.4. Язык вопросов для ostis-систем*);
- использование различных *интероперабельных решателей задач* с возможностью объяснения полученных решений (см. *Главу 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*);
- использование *картографического интерфейса* для визуализации исходных данных и результатов (см. *Главу 4.1. Общие принципы организации интерфейсов ostis-систем*).

Реализация возможностей *интеллектуальных геоинформационных систем* может быть осуществлена с помощью:

- *систем управления базами знаний*,
- мультимедийных *баз знаний* и *данных* по областям применения,
- интероперабельных *решателей задач*,
- интеллектуального *картографического интерфейса*,
- *экспертных систем* в различных областях деятельности людей,
- *систем поддержки принятия решений*,
- *систем интеллектуальной помощи*.

интеллектуализация геоинформационных систем предполагает решение следующих задач:
- использование цифрового картографического материала и данных *дистанционного зондирования Земли* в проблемно-ориентированных областях (см. *Абламейко С.В..ГеогрИССЦК-2000ст*);
- планирование действий в динамически меняющейся ситуации в условиях неполных или нечетких данных с использованием экспертных знаний (см. *Ивакин Я.А.МетодИПГСнОО-2009дс*);
- разрешение земельных споров;
- анализ чрезвычайных ситуаций и подготовка материалов для принятия решений по предотвращению или ликвидации их последствий;
- создание систем поддержки принятия решений для прикладных *геоинформационных систем* территориального планирования и управления (см. *Белякова М.Л.ИнтелГСдУИТ-2016кн*);
- разработка диагностических экспертных систем по геологоразведочной деятельности со средствами удаленного доступа к ним;
- логистическое планирование, создание экспертных систем и программных средств управления предприятиями;
- создание систем контроля и навигации;
- создание *экспертных систем* прогнозирования возникновения и развития на местности техногенных и природных ситуаций: наводнений, землетрясений, экстремальных погодных условий (осадки, температура), эпидемий, распространения радионуклидов, химических выбросов, метеопрогноз и так далее;
- создание *экспертных систем* выбора участков местности для строительства различных объектов;
- создание *экспертных систем* планирования эффективного использования сельскохозяйственных земель;
- создание *экспертных систем* и программных средств для анализа геоданных;
- создание систем распознавания образов и изображений по данным *дистанционного зондирования Земли*;
- создание банков цифровой картографической информации со средствами удаленного доступа к ним;
- обработка изображений;
- ретроспективный анализ событий (см. *Губаревич А.В..ОнтолПИСвОИ-2017ст*, *Губаревич А.В..СтрукБЗвИСпИ-2018ст*);
- создание *информационно-поисковых систем* по наукам о Земле и *Геоинформатике*;
- разработка обучающих систем для подготовки специалистов и экспертов со средствами удаленного доступа к ним.

Полное решение поставленных выше задач требует использования стандартов открытых систем и использование онтологий *объектов местности* как интегрирующих элементов различных *предметных областей*.

§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах

⇒ *подраздел**:
- *Пункт 7.8.3.2. Геосемантические характеристики объектов местности*
- *Пункт 7.8.3.3. Топологические пространственные отношения между объектами местности*
- *Пункт 7.8.3.4. Стратифицированная модель информационного пространства объектов местности*
- *Пункт 7.8.3.1. Базовая классификация объектов местности*

Основным подходом к обеспечению *интероперабельности* является разработка *онтологий*. Наиболее часто онтологии, используемые в *геоинформатике*, обычно рассматриваются как доменные *онтологии*, которые принято называть *географическими онтологиями*, или геоонтологиями (см. *Блискавицкий А.А.КонцеПГИСиУ-2012кн*, *Блискавицкий А.А.СеманГОФГиИ-2014ст*). Одной из задач в разработке *онтологий* является четкое и однозначное определение семантики примитивных терминов (атомарных элементов, которые не могут быть далее разделены). Чтобы решить эту проблему исследователи предложили обосновать примитивные термины геонтоглогий на основе географических явлений (см. *Hu Y.GeospS-2019art*, *Janowicz K..GeospSaLSD-2012art*).

Определение *онтологии* дано в *Главе 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*. Применительно к *базам знаний онтология* — это *формализация* некоторой области *знаний* на основе концептуальной схемы со структурой, содержащей классы объектов, их связи и правила, допускающей компьютерный анализ. Соответственно в состав онтологии предметной области входят экземпляры, понятия, атрибуты и отношения. *предметные области*, для которых целесообразна разработка *геоинформационных систем*, предполагают построение онтологии, которую будем называть *геоонтологией*.

геоонтология
- ≔ [географическая онтология]
- ≔ [онтология предметных областей, в состав экземпляров объектов местности которых входят их геосемантические характеристики]
- ≔ [онтология предметных областей, экземпляры объектов которых используют пространственно-соотнесенные данные о территории, социальных и природных явлениях]
- ⊂ *онтология*
- ∋ *класс объектов местности^*
- ∋ *пространственное отношение*

класс объектов местности^
- ≔ [класс геопространственных понятий естественного или искусственного происхождения, природных явлений, имеющие общие признаки (семантические атрибуты), характерные для определенного класса объектов местности и описывающие внутренние характеристики понятия]
- ∋ *пример'*:
  автомагистраль

Класс *объектов местности* “автомагистраль”, обладает общим семантическим атрибутом “материал покрытия”, а смысловое значение данного признака принимает значение, определенное на шкале значений признаков {асфальт (асфальтобетон); цементобетон; булыжник; брусчатка; гравий; камень колотый; клинкер; шлак; щебень; битумоминеральная смесь; металл; твердое покрытие; грунт; лед; без покрытия}.

объект местности
- ≔ [определенный элемент земной поверхности естественного или искусственного происхождения, природное явление, реально существующие на рассматриваемый момент времени в пределах области локализации, для которого известно или может быть установлено местоположение, включая размеры и положение границ, и заданы признаки, отражающие семантические атрибуты такого элемента, характерные для определенного *класса объектов местности*, с заданными *пространственными отношениями*, отражающими связи с другими *объектами местности*]

Пункт 7.8.3.1. Базовая классификация объектов местности

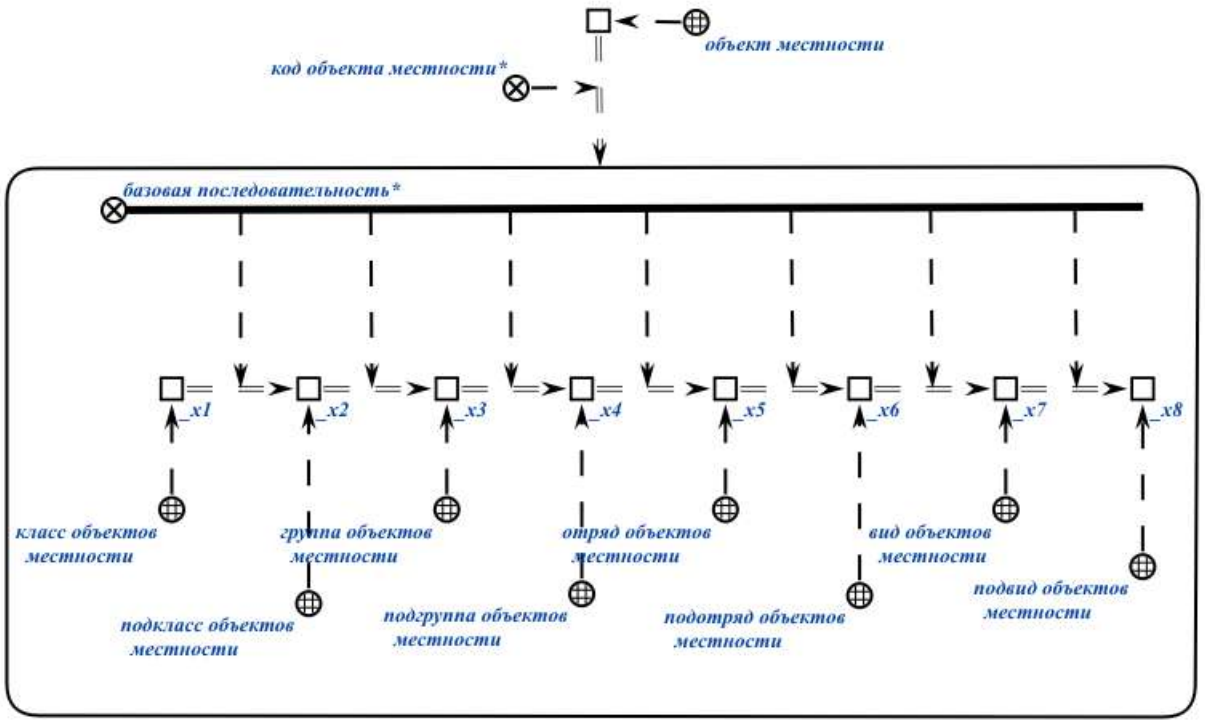
объект местности
⇒ *разбиение**:
Типология объектов местности по тематике^
= {• *водный объект местности (сооружение)*
• *населенный объект местности*
• *промышленный (сельскохозяйственный или социально-культурный) объект местности*
• *дорожная сеть (сооружение)*
• *растительный покров (грунт)*
}

В основу построения онтологической модели *объектов местности* положим разработанный и действующих в настоящее время в *Республике Беларусь* классификатор топографической информации, отображаемой на топографических картах и планах городов (см. *ЦифроКМИО-2007ст*). В соответствии с данным обстоятельством объектами классификации являются объекты местности, которым соответствуют объекты карты, а также признаки (характеристики) этих объектов. С этой целью в онтологической модели *объекты местности* разбиваются по типу локализации на *площадные объекты*, *линейные (полилинейные) объекты* и *точечные объекты*.

На следующем этапе разработки онтологии *объектов местности* зададим разбиение *объектов местности* по ортогональным основаниям, что соответствует размещению объектов в соответствии с тематическими слоями в *геоинформационных системах*.

Онтология *объектов местности* представляет собой дерево классификации в соответствии с иерархией, приведенной на рисунке *Рисунок. Уровни иерархии классов объектов местности*. Для каждого класса *объектов местности* установлены родовидовые связи.

Рисунок. Уровни иерархии классов объектов местности
=



Для каждого *объекта местности* выделены основные, присущие только ему, семантические характеристики. Особо отметим, что метрические характеристики таким свойством не обладают. Согласно данному классификатору каждый класс *объектов местности* имеет уникальное однозначное обозначение. Иерархия классификатора имеет восемь ступеней классификации и состоит из *кода класса*, *кода подкласса*, *кода группы*, *кода подгруппы*, *кода отряда*, *кода подотряда*, *кода вида*, *кода подвида*. Таким образом, благодаря способу кодирования уже заданы родовидовые связи, отражающие соотношения различных классов *объектов местности*, а также установлены

характеристики конкретного класса *объектов местности*. В связи с тем, что задаются основные свойства и отношения не конкретных *физических объектов*, а их классов, то такая информация является по отношению к конкретным *объектам местности* метаинформацией, а совокупность данной метаинформации представляет собой онтологию *объектов местности*, которая в свою очередь является частью *базы знаний интеллектуальной геоинформационной системы*.

объект местности
⇒ *разбиение**:
Типология объектов местности по локализации^
= {• *точечный объект местности*
:= [объект местности, который не выражается в масштабе карты]
⇒ *пример**: *включение**:
• *колодец*
• *осветительная опора*
• *дорожный знак*
• *линейный объект местности*
:= [объект местности, длина которого выражается в масштабе карты]
⇒ *пример**: *включение**:
• *мост*
• *полилинейный объект местности*
:= [объект местности, состоящий из двух и более сегментов линейных объектов]
⇒ *пример**: *включение**:
• *река*
• *дорога*
• *улица*
• *площадной объект местности*
:= [объект местности, площадь которого выражается в масштабе карты]
⇒ *пример**: *включение**:
• *озеро*
• *административный район*
• *государство*
}

Пункт 7.8.3.2. Геосемантические характеристики объектов местности

Особенностью *геоонтологии* является использование для формализации *предметных областей* специальных элементов, уточняющих пространственные характеристики объектов местности, которые назовем ***геосемантическими характеристиками объектов местности***.

геосемантическая характеристика объекта местности
⇒ *разбиение**:
{• *координатное местоположение объекта местности*^
• *пространственное отношение*
• *пространственное отношение главных направлений*
• *динамика состояния объекта местности*^
}

координатное местоположение объекта местности
:= [географическое положение, расположение *объекта местности* или явления, которое задается в *системе геодезических координат*]

геокодирование
:= [установление связи между *объектом местности* и его *местоположением*]
⊂ *действие*

пространственное отношение
:= [класс отношений, задающие семантические свойства объекта местности по отношению к другим объектам местности]

⇒ *разбиение**:
{• *топологическое пространственное отношение*
• *отношение пространственной упорядоченности*
• *метрическое пространственное отношение*
}

отношение пространственной упорядоченности
⇒ *разбиение**:
{• *отношение расположения объектов местности*
• *отношение главных направлений объектов местности*
}

отношение расположения объектов местности
⊂ *ориентированное отношение*
:= [позволяет определить, какое положение занимает один *объект местности* по отношению к другому *объекту местности*]
∋ *объект местности располагается перед другим объектом местности**
∋ *объект местности располагается за другим объектом местности**
∋ *объект местности располагается слева от другого объекта местности**
∋ *пример'*:
Водонапорная башня располагается слева от дороги
∋ *объект местности располагается справа от другого объекта местности**
∋ *объект местности располагается над другим объектом местности**
∋ *объект местности располагается под другим объектом местности**
∋ *объект местности располагается ближе другого объекта местности**
∋ *объект местности располагается дальше другого объекта местности**

отношение главных направлений объектов местности
⊂ *ориентированное отношение*
:= [позволяет определить, какое главное направление занимает один *объект местности* по отношению к другому *объекту местности*]
⇒ *примечание**:
[*отношение главных направлений объектов местности* инвариантно относительно сдвига и масштабирования]
∋ *объект местности по отношению к другому объекту местности занимает главное направление север**
∋ *объект местности по отношению к другому объекту местности занимает главное направление северо-восток**
∋ *объект местности по отношению к другому объекту местности занимает главное направление восток**
∋ *объект местности по отношению к другому объекту местности занимает главное направление юго-восток**
∋ *объект местности по отношению к другому объекту местности занимает главное направление юг**
∋ *объект местности по отношению к другому объекту местности занимает главное направление юг-запад**
∋ *объект местности по отношению к другому объекту местности занимает главное направление запад**
∋ *объект местности по отношению к другому объекту местности занимает главное направление северо-запад**

метрическое пространственное отношение
:= [характеризует информацию о расстоянии между *объектами местности*]
⇒ *измерение**:
километр
⇒ *измерение**:
метр
⊃ *шкальное метрическое пространственное отношение*
⇒ *измерение**:
шкала

система геодезических координат
:= [система координат, используемая для определения местоположения объектов на Земле]
∋ *пример'*:

WGS84

⇒ *примечание**:

[Всемирная система геодезических параметров Земли 1984 года, в число которых входит система геоцентрических координат. В отличие от локальных систем, является единой системой для всей планеты.]

∋ *пример'*:

СК-63

∋ *пример'*:

СК-95

Пункт 7.8.3.3. Топологические пространственные отношения между объектами местности

Между экземплярами *объектов местности* можно установить *топологические пространственные отношения*: *включение**, *граничить**, *пересечение** и *примыкание**.

топологическое пространственное отношение

:= [класс *пространственных отношений*, заданных над *объектами местности*, находящихся в отношении связности и смежности между *объектами местности*]

⇒ *примечание**:

топологическое пространственное отношение инвариантно относительно перемещения, поворота и масштабирования

∋ *включение**

⊃ *включение точечного объекта местности в площадной объект местности**

⇒ *пример**:

⊃ *включение линейного (полилинейного) объекта местности в площадной объект местности**

⇒ *пример**:

⊃ *включение площадного объекта местности в площадной объект местности**

⇒ *пример**:

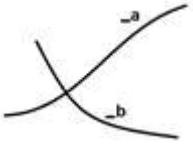
∋ *граничить**

⇒ *пример**:

∋ *пересечение**

⊃ *пересечение двух линейных (полинейных) объектов местности**

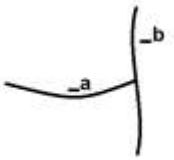
⇒ *пример**:



⊃ *пересечение линейного (полинейного) и площадного объектов местности**

⇒ *пример**:

∋ *примыкание**
⇒ *пример**:



Отношение “включения*” будет устанавливаться между *площадным* и *линейным*, *площадным* и *точечным*, *площадными объектами местности*. Отношение “пересечения*” будет устанавливаться между *линейными* и *площадными* и *линейными объектами местности*. Отношение “граничить*” будет устанавливаться между *площадными объектами местности*. Отношение “примыкания*” устанавливается между *линейными объектами местности*. Для всех *картографических отношений* существуют структуры для их хранения.

Пункт 7.8.3.4. Стратифицированная модель информационного пространства объектов местности

С целью *интеграции предметных областей* с пространственными компонентами *геоинформационных систем*, соответственно повышения *интероперабельности* этих систем, предлагается *Стратифицированная модель информационного пространства объектов местности*, которая задается следующим образом:

$$S^{\mu}, \mu \in I = \{S_{PO\mu}, S_{OM}, E_{OM}\}, \tag{7.8..1}$$

где I - множество *предметных областей*;

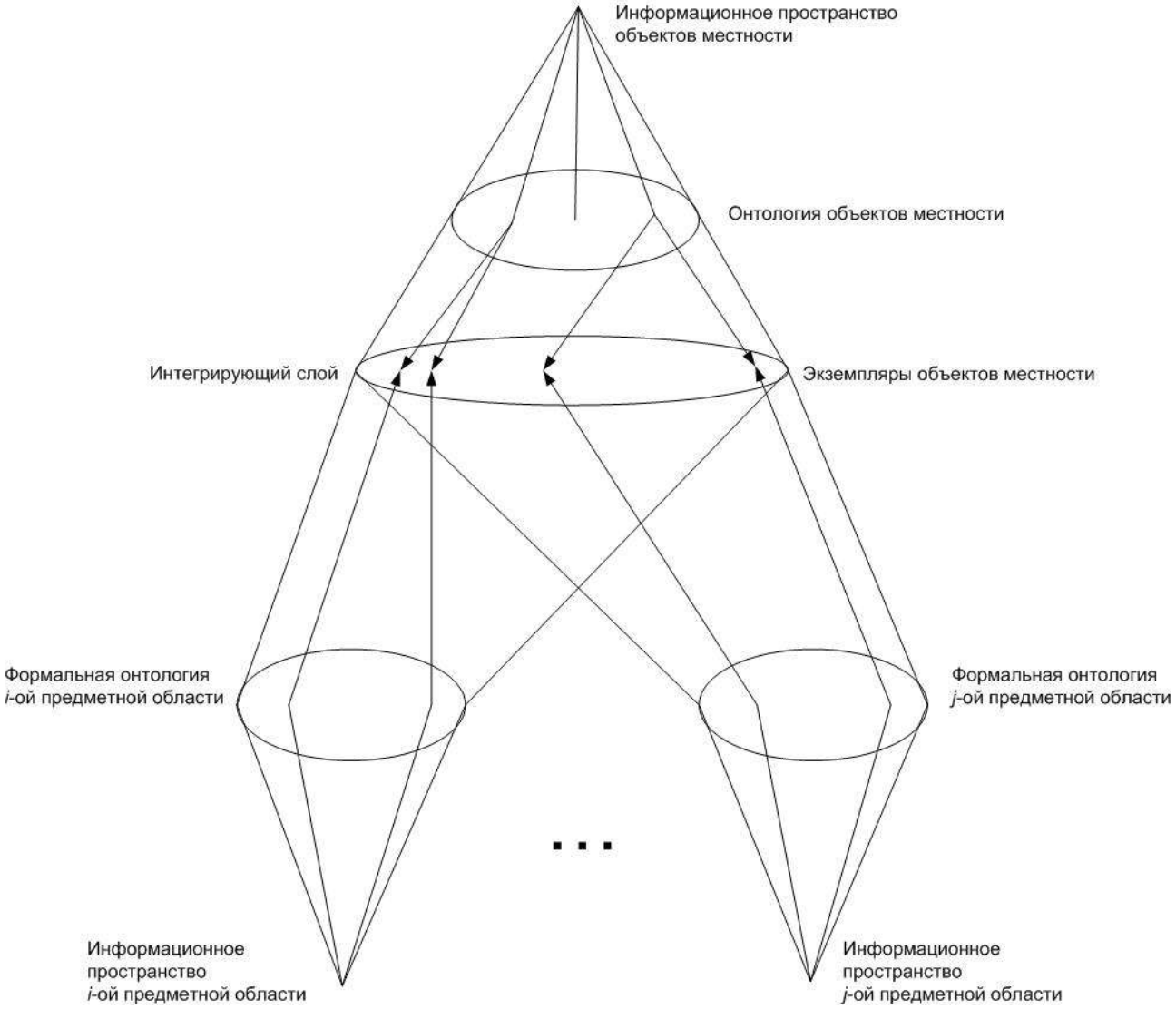
$S_{PO\mu}$ — онтология μ-ой *предметной области*;

S_{OM} — онтология *объектов местности*;

E_{OM} — экземпляры *объектов местности*.

На *Рисунок. Стратифицированная модель информационного пространства объектов местности* представлена геометрическая интерпретация предложенной *гибридной модели знаний*, где показано, что слой экземпляров *объектов местности* является интегрирующим слоем с предметными знаниями различных *предметных областей*, в которых уже непосредственно используются конкретные *объекты местности*. При такой организации знаний возможно многократно использовать разработанную онтологию *объектов местности* в разных предметных областях и, соответственно, для решения различных прикладных задач.

Рисунок. Стратифицированная модель информационного пространства объектов местности
=



§ 7.8.4. Решатель задач интеллектуальной геоинформационной ostis-системы

решатель задач интеллектуальной геоинформационной ostis-системы представляет собой коллектив взаимодействующих друг с другом *sc-агентов*, позволяющих решать *геоинформационные задачи*. Ниже приведена базовая декомпозиция *решателя задач интеллектуальной геоинформационной системы* на основные классы sc-агентов геоинформационного назначения.

решатель задач интеллектуальной геоинформационной системы
⇒ *декомпозиция**:
- {• *Абстрактный sc-агент вычисления геометрических характеристик объектов местности*
  ⇒ *декомпозиция**:
  - {• *Абстрактный sc-агент обработки точечных объектов местности*
    • *Абстрактный sc-агент обработки линейных (полилинейных) объектов местности*
    • *Абстрактный sc-агент обработки площадных объектов местности*
    }
- • *Абстрактный sc-агент определения типа локализации объекта местности*
- • *Абстрактный sc-агент сопряжения с различными картографическими системами и сервисами, системами измерений и временными интервалами*
  ⇒ *декомпозиция**:
  - {• *Абстрактный sc-агент сопряжения с картографическими системами*

- *Абстрактный sc-агент сопряжения с единицами измерений*
- *Абстрактный sc-агент сопряжения с временными интервалами*

}

- *Абстрактный sc-агент установления топологических связей между объектами местности*
- *Абстрактный sc-агент верификации базы знаний объектов местности*

⇒ *декомпозиция**:

{
- *Абстрактный sc-агент верификации полноты заполнения базы знаний объектов местности*
- *Абстрактный sc-агент верификации корректности значений семантических атрибутов объектов местности*
- *Абстрактный sc-агент верификации корректности значений пространственных атрибутов объектов местности*

}

}

§ 7.8.5. Картографический интерфейс интеллектуальной геоинформационной ostis-системы

картографический интерфейс — это *пользовательский интерфейс*, предназначенний для визуального отображения *объектов местности* на каком-то *языке карт*. *картографический интерфейс* как вид *пользовательского интерфейса* должен обладать следующими свойствами:

- высоким уровнем согласованности понятий, используемых при визуализации информации об объектах местности;
- высоким уровнем простоты (естественностью) и понятности для любого конечного пользователя (интерфейс должен быть снисходительным к уровню подготовки пользователя, то есть дружелюбным);
- высоким уровнем привлекательности (естетичности) и легкости восприятия;
- и так далее.

Частным видом *картографического интерфейса* является *картографический интерфейс интеллектуальной геоинформационной ostis-системы*.

Разрабатываемый *Язык карт для ostis-систем* относится к семейству семантически совместимых языков — *sc-языков* и предназначен для формального описания *объектов местности* и отношений между ними в *геоинформационных системах*. Поэтому ***Синтаксис Языка карт для ostis-систем***, как и *синтаксис* любого другого *sc-языка*, является *Синтаксисом SC-кода*. Такой подход позволяет:

- использовать минимум средств для интерпретации заданных *объектов местности* на карте;
- использовать *Язык вопросов для ostis-систем*;
- сводить поиск на большую часть заданных *вопросов* к поиску информации в текущем состоянии *базы знаний ostis-системы*.

Денотационная семантика Языка карт для ostis-систем включает *пространственные отношения* и *геосемантические характеристики объектов местности*.

Язык карт для ostis-систем

:= [Предлагаемый нами вариант внешнего языка для представления и визуализации информации об объектах местности в картографическом интерфейсе интеллектуальной геоинформационной ostis-системы]

∈ *sc-язык*

⇒ *синтаксис языка**:

Синтаксис Язык карт для ostis-систем

⊂ *Синтаксис SC-кода*

⇒ *денотационная семантика языка**:

Денотационная семантика Языка карт для ostis-систем

:= [Онтологии объектов местности, пространственных отношений между ними и их геосемантических характеристик]

⊃ *Семантическая классификация вопросов*

⇒ *операционная семантика языка**:

Операционная семантика Языка карт для ostis-систем

:= [Коллектив sc-агентов интерпретации Языка карт для ostis-систем]

Ключевое достоинство *картографического интерфейса интеллектуальной геоинформационной ostis-системы* и соответствующего ему *Языка карт для ostis-систем* по сравнению с другими видами *пользовательских интерфей-*

сов состоит в том, что с помощью их можно визуализировать любые *классы объектов местности*^ и информацию о них достаточно простым способом, понятным любому пользователю.

Покажем пример отображения *семантической окрестности объекта местности* “Река Березина” на *SCg-коде* (*Рисунок. Описание Реки Березина на SCg-коде*) и *Языке карт* (*Рисунок. Описание Реки Березина на Языке карт*).

Рисунок. Описание Реки Березина на SCg-коде

=

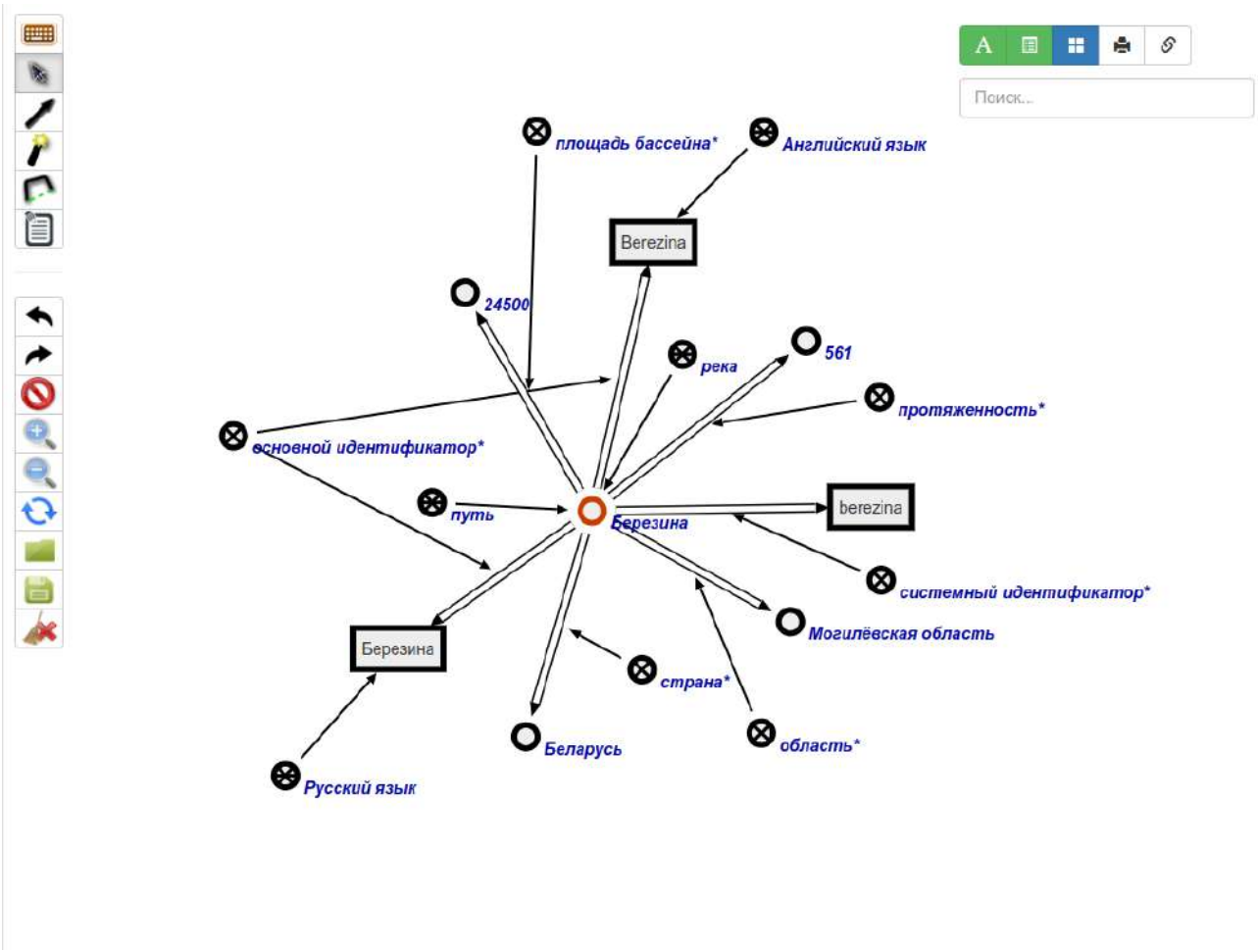
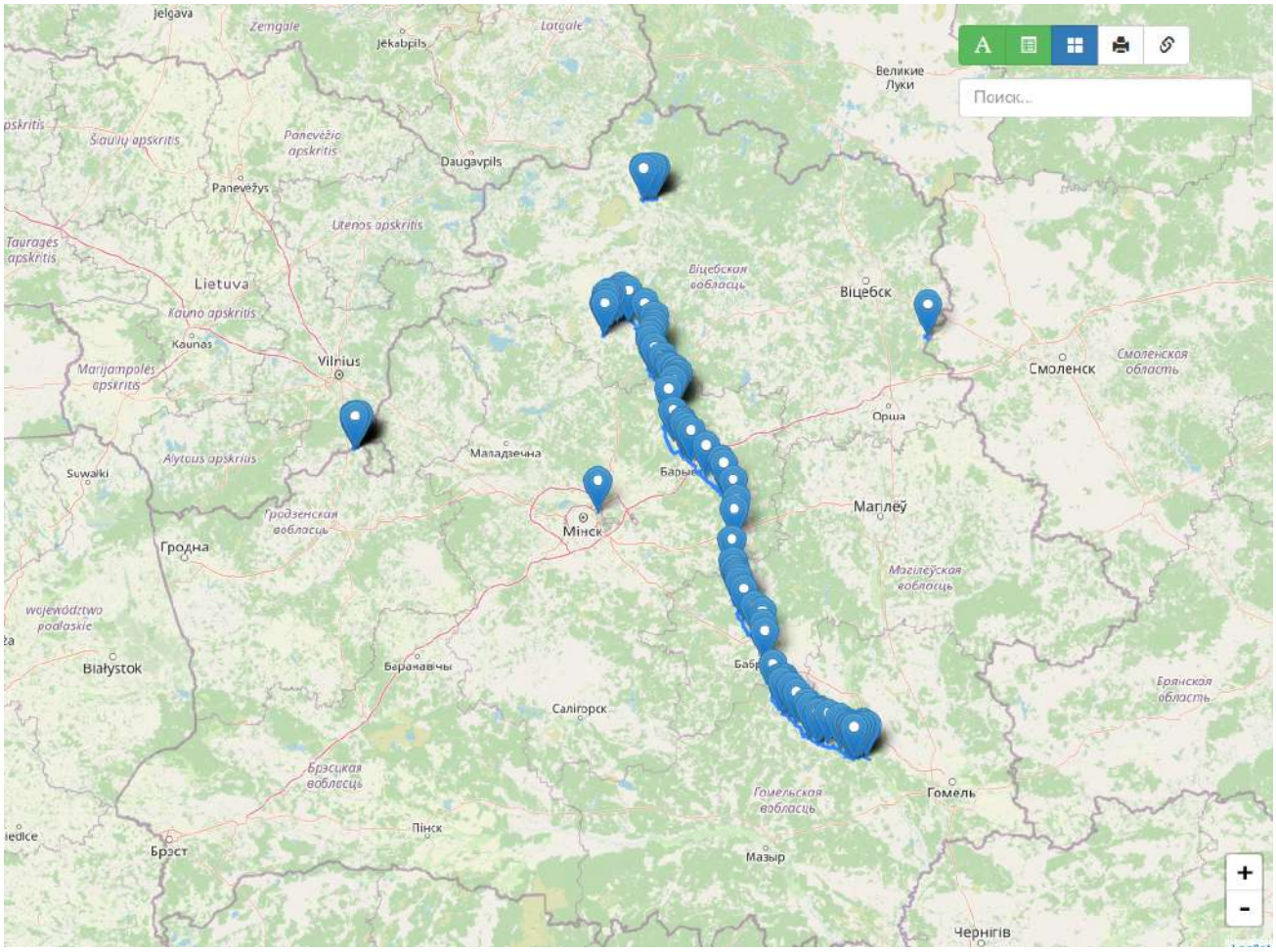


Рисунок. Описание Реки Березина на Языке карт
=



На обоих рисунках показаны *семантические окрестности* одного и того же *объекта местности*, но представленных на разных *внешних языках представления знаний*: *SCg-коде* и *SCg-коде*.

Отображение информации об *объекте местности* происходит при помощи средств *Метасистемы OSTIS* (см. *Главу 7.2. Метасистема OSTIS*) и *Технологии OpenMapStreet* и включает следующие этапы:

- поиск *семантической окрестности* заданного *объекта местности* (то есть *пространственных отношений* между заданным *объектом местности* и другими *объектами местности*, а также геосемантических характеристик заданного *объекта местности*);
- определение географических кодов *территориальных объектов местности*, в которых располагается заданный *объект местности*, а также других объектов местности, связанных *пространственными отношениями* с заданным *объектом местности*;
- получение картографических данных о *территориальных объектах местности* и их соотнесение с информацией о *территориальных объектах местности* в базе знаний;
- определение класса *объекта местности*;
- визуализация *семантической окрестности* заданного *объекта местности* на *Языке карт*.

§ 7.8.6. Многократно используемые компоненты интеллектуальных геоинформационных ostis-систем

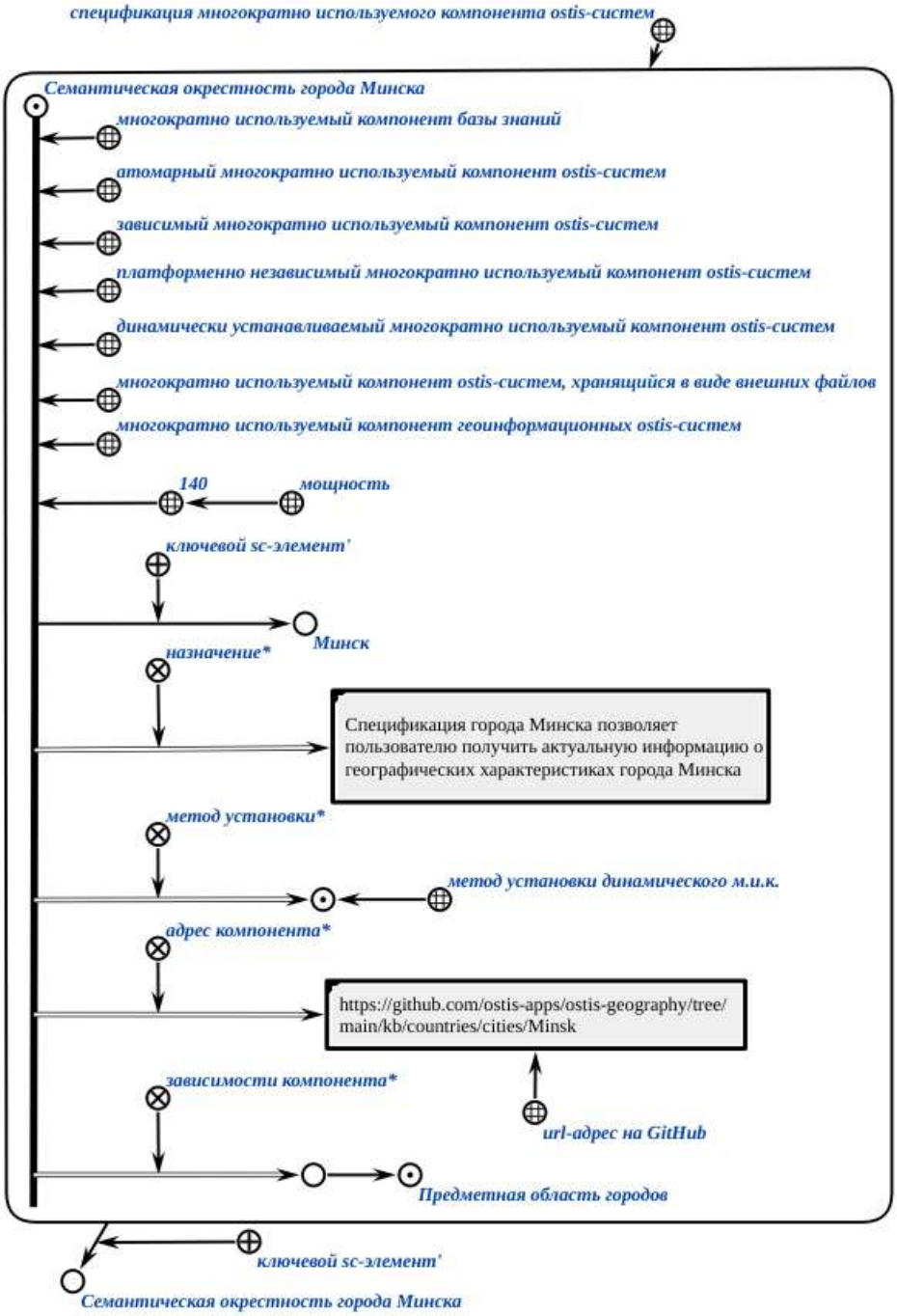
Одним из требованием, предъявляемым к *Технологии проектирования интеллектуальных геоинформационных систем*, является обеспечение возможности совместного использования в рамках *интеллектуальных геоинформационных ostis-систем* различных *видов знаний* и различных *моделей решения задач*, а также различных видов *интерфейсов*. Следствием данного требования является необходимость реализации компонентного подхода на

всех уровнях, от простых компонентов баз знаний, решателей задач и интерфейсов до целых *встраиваемых ostis-систем* (см. *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*).

Таким образом, *база знаний интеллектуальной геоинформационной ostis-системы* должна включать спецификацию *объектов местности*, то есть описание их *семантических окрестностей* (см. *SCg-текст. Пример многократно используемого компонента базы знаний интеллектуальных геоинформационных ostis-систем*), *решатель задач интеллектуальной геоинформационной ostis-системы* — спецификацию sc-агентов, решающих различные *геоинформационные задачи* (например, sc-агент классификации *географических объектов* (см. *SCg-текст. Пример многократно используемого компонента решателя задач интеллектуальных геоинформационных ostis-систем*), sc-агент определения *пространственных отношений* между *объектами местности* и так далее), а *картографический интерфейс интеллектуальной геоинформационной ostis-системы* — описание *отношений* между *объектами местности* на *карте*.

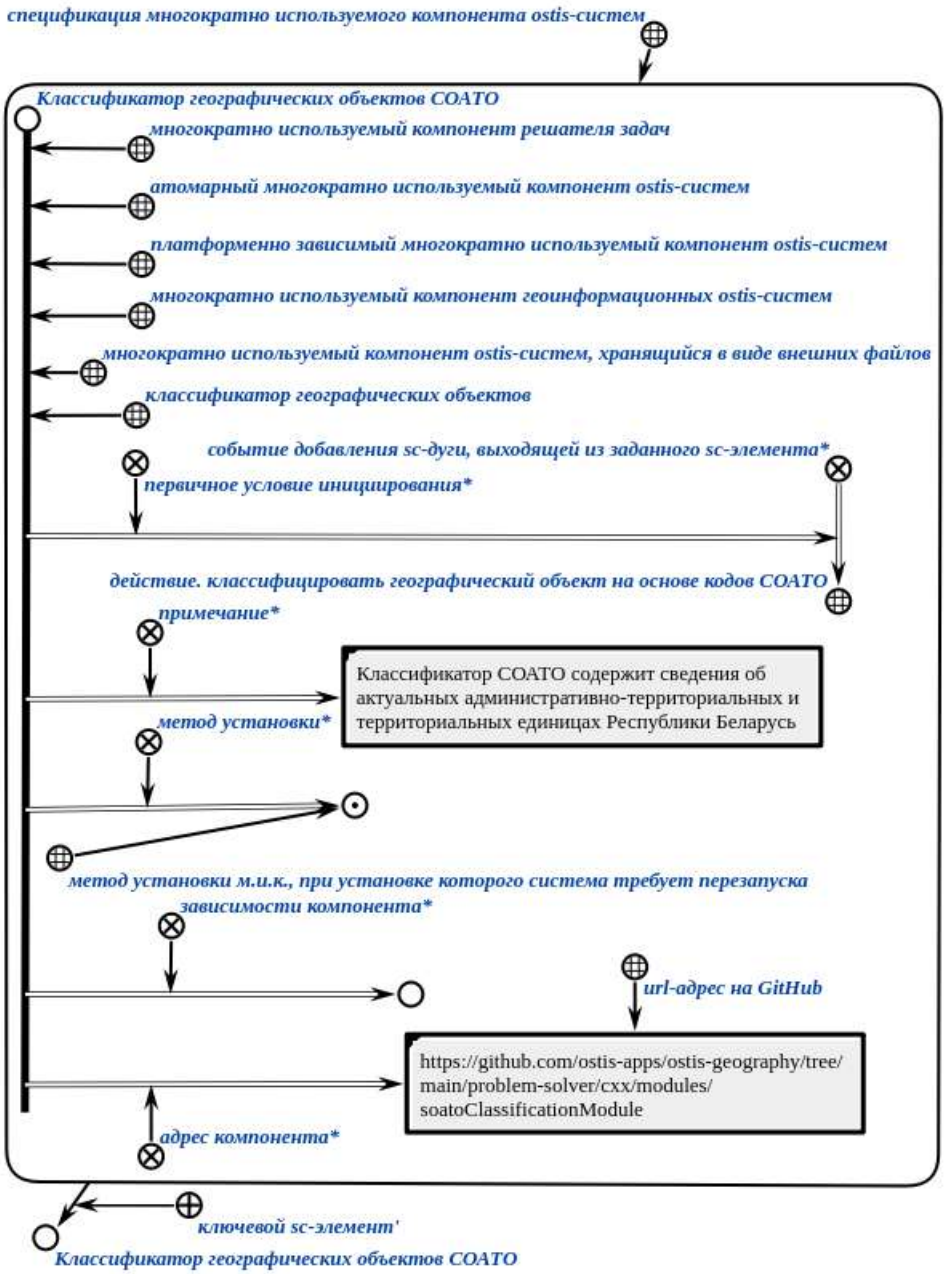
SCg-текст. Пример многократно используемого компонента базы знаний интеллектуальных геоинформационных ostis-систем

=



SCg-текст. Пример многократно используемого компонента решателя задач интеллектуальных геоинформационных ostis-систем

=



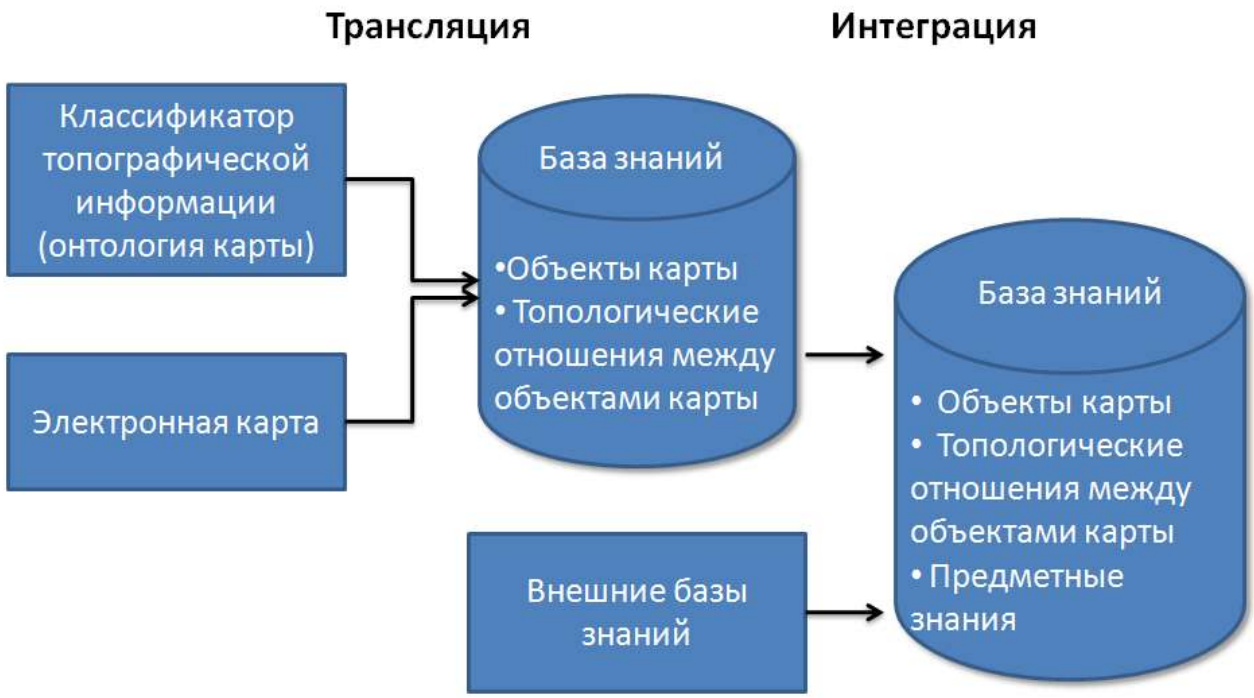
§ 7.8.7. Средства автоматизации проектирования интеллектуальных геоинформационных ostis-систем

Проектирование *интеллектуальных геоинформационных систем* осуществляется поэтапно. На первом этапе формируется *база знаний предметной области* и с этой целью анализируется электронная карта и осуществляется трансляция в *базу знаний объектов местности* с установлением *геосемантических характеристик объектов местности* для соответствующей территории. На данном этапе определяется, во-первых, к какому классу принадлежит исследуемый *объект местности* и, далее в зависимости от типа объекта, формируется *понятие базы знаний*, соответствующее конкретному *объекту местности*. Таким образом, создается множество понятий, описывающих конкретные *объекты местности* для каждого *класса объектов местности*^. Следует отметить, что именно на данном этапе формирования *базы знаний* устанавливаются *геосемантические характеристики объектов местности*.

На втором этапе проектирования *интеллектуальной геоинформационной системы* происходит интеграция полученной на первом этапе *базы знаний* с внешними *базами знаний*. На этом этапе, помимо географических *знаний*, добавляться *знания* смежных *предметных областей*, тем самым становится возможным установление межпредметных связей. Наглядным примером служит интеграция с биологическими классификаторами, которые в реализации представляют собой онтологию объектов флоры и фауны. Такая интеграция расширяет функциональные и интеллектуальные возможности прикладной *интеллектуальной геоинформационной системы*. Отметим, что на данном этапе снимается омонимия в названиях *объектов местности*, принадлежащих классам населенных пунктов. Для населенных пунктов *Республики Беларусь* это достигается за счет использования *системы обозначений объектов административно-территориального деления и населенных пунктов* и осуществляется семантическое сопоставление *объектов местности* по следующему принципу:

- определяется *класс объекта местности*^;
- определяется подкласс, вид, подвид *объекта местности* в соответствии с классификатором *объектов местности*, то есть виды *объектов местности* в *геоонтологии*;
- определяются атрибуты и характеристики, которые присущи данному *классу объектов местности*^;
- определяются значения характеристик для данного *класса объекта местности*^;
- устраняется омонимия идентификации;
- устанавливаются соответствующие связи между *объектом местности* и *понятием* в *базе знаний* с установленными *геосемантическими характеристиками объектов местности*;
- устанавливаются *пространственные отношения* между *объектами местности*, отнесенными к определенным классам.

Рисунок. Этапы проектирования баз знаний интеллектуальных геоинформационных ostis-систем
=



Заключение к Главе 7.8.

Перечислим основные положения данной главы:

- развитие *геоинформационных систем* заключается в их интеллектуализации, благодаря чему расширяется круг прикладных задач с использованием *знаний* об *объектах местности*;
- предложено карту рассматривать как *информационную конструкцию*, элементами которой являются *объекты местности*, тем самым обеспечивается *интероперабельность геоинформационных систем* за счет перехода от карты к семантическому описанию элементов карты, то есть объектов местности и связей (пространственных отношений) между ними;

- обеспечение *интероперабельности* достигается благодаря разработке онтологий предметных областей, а установление *геосемантических характеристик объектов местности* позволяет задавать пространственные характеристики *объектов местности*;
- наличие частной *Технологии проектирования интеллектуальных геоинформационных систем* обеспечивает процесс проектирования интеллектуальных геоинформационных систем, построенных по принципам ostis-систем.

Глава 7.9.

Обеспечение информационной безопасности в рамках Экосистемы OSTIS

⇒ *автор**:
- *Чертков В. М.*
- *Захаров В. В.*

⇒ *аннотация**:
[В главе рассмотрены методы и средства обеспечения безопасности традиционных информационных систем, особенности обеспечения информационной безопасности интеллектуальных систем нового поколения (см. *Главу 1.2. Интеллектуальные компьютерные системы нового поколения*) и принципы, лежащие в основе обеспечения информационной безопасности ostis-систем.]

⇒ *подраздел**:
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

⇒ *библиографическая ссылка**:
- *Исобоев Ш.И..ИнтелСМБСБСнОМО-2022ст*
- *Скрыпников А.В..РешенЗИБсИ-2021ст*
- *Частикова В.А..МетодПСАИ-2022ст*
- *Абдурахман Д.Д.ИскусИиМОвК-2022ст*
- *Остроух А.В.ИнтелСМ-2020кн*
- *Баранович А.Е.СеманАИБКЗ-2011ст*
- *Хоанг В.К..РешенОЗвРППБРсСБД-2013ст*
- *Голенков В.В..СеманМПиОБ-2017ст*
- *Дементьев А.В.МетриСД-2022ст*
- *Golenkov V.V..Metho aTfECoC-2019art*
- *Дружинин В.Н..КогниПУдВ-2002кн*
- *Созинова Е.Н.ПримеЭСдАиОИБ-2011ст*

Введение в Главу 7.9.

Большое разнообразие моделей обеспечения информационной безопасности, все возрастающий объем данных, которые необходимо анализировать для обнаружения атак на информационные системы, изменчивость методов атак и динамическое изменение защищаемых информационных систем, необходимость оперативного реагирования на атаки, нечеткость критериев обнаружения атак и выбора методов и средств реагирования на них, нехватка высококвалифицированных специалистов по защите влечет за собой потребность в использовании методов *Искусственного интеллекта* для решения задач безопасности.

§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения

Информационную безопасность интеллектуальных систем следует рассматривать с двух точек зрения:
- с точки зрения применения *Искусственного интеллекта* в информационной безопасности;
- с точки зрения организации информационной безопасности в интеллектуальных системах.

Применение *Искусственного интеллекта* в информационной безопасности

Следует отметить, что *Искусственный интеллект* активно применяется для мониторинга и анализа уязвимостей безопасности в сетях передачи информации (см. *Исобоев Ш.И..ИнтелСМБСБСнОМО-2022ст*). Система *Искусственного интеллекта* позволяет машинам более эффективно выполнять поставленные задачи, такие как:
- визуальное восприятие, распознавание речи, принятие решений и перевод с одного языка на другой;

- *обнаружение вторжений — Искусственный интеллект* может обнаруживать сетевые атаки, заражения вредоносным программным обеспечением и другие киберугрозы;
- *Кибераналитика — Искусственный интеллект* также используется для анализа больших данных с целью выявления закономерностей и аномалий в системе кибербезопасности организации с целью обнаружения не только известных, но и еще неизвестных угроз;
- *безопасная разработка программного обеспечения — Искусственный интеллект* может помочь создать более безопасное программное обеспечение, предоставляя разработчикам обратную связь в режиме реального времени.

При этом *Искусственный интеллект* используется не только для защиты, но и для нападения, например для эмуляции акустических, видео и других образов с целью обмана механизмов аутентификации и дальнейшей имперсонации, обман проверки человек или робот capcha.

В настоящее время можно определить следующие классы систем, в которых применяется *Искусственный интеллект* (см. *Скрыпников А.В..РешенЗИБсИ-2021ст*):

- UEBA (User and Entity Behavior Analytics) — система анализа поведения субъектов (пользователей, программ, агентов) на предмет обнаружения нестандартного поведения и использования их для обнаружения потенциальных угроз с использованием шаблонов угроз (паттернов);
- TIP (Threat Intelligence Platform) — платформы раннего обнаружения угроз на основе сбора и анализа информации индикаторов компрометации и реагирования на них. Применение методов машинного обучения повышает эффективность обнаружения неизвестных угроз на ранних этапах;
- EDR (Endpoint Detection and Response) — системы обнаружения атак оперативного реагирования на конечных точках компьютерной сети. Могут обнаруживать вредоносные программы, автоматически классифицировать угрозы и самостоятельно реагировать на них;
- SIEM (Security Information and EventManagement) — системы сбора и анализа информации о событиях безопасности от сетевых устройств и приложений в реальном времени и оповещения;
- NDR (Network Detection and Response) — системы обнаружения атак на сетевом уровне и оперативного реагирования на них. *Искусственный интеллект* использует накопленную статистику и базу знаний об угрозах;
- SOAR (Security Orchestration and Automated Response) — системы, позволяющие выявлять угрозы информационной безопасности и автоматизировать реагирование на инциденты. В решениях данного типа, в отличие от SIEM-систем, *Искусственный интеллект* помогает не только проводить анализ, но и автоматически реагировать надлежащим образом на выявленные угрозы;
- Средства защиты приложений (Application Security) — системы, позволяющие определять угрозы безопасности прикладных приложений, управлять процессом мониторинга и устранения таких угроз;
- Антифрод (Antifraud) — платформы в режиме реального времени обнаруживают угрозы в бизнес-процессах и мошеннические операции. *Искусственный интеллект* используется для определения отклонений от идентифицированных бизнес-процессов с целью выявления вторжений или уязвимости процессов и повышает адаптивность к изменению логики и метрик бизнес-процессов.

В работе *Частикова В.А..МетодПСАИ-2022ст* предложена методика построения нейроиммунной системы анализа инцидентов информационной безопасности, объединяющей модули сбора и хранения (сжатия) данных, модуль анализа и корреляции событий информационной безопасности и подсистемы обнаружения сетевых атак на основе сверточных нейронных сетей. Использование технологий машинного обучения в информационной безопасности создает узкие места и системные уязвимости, которые можно использовать, и имеет следующие недостатки (см. *Абдурахман Д.Д.ИскусИиМОвК-2022ст*):

- наборы данных, которые должны быть сформированы из значительного количества входных выборок, что требует много времени и ресурсов;
- требуется огромное количество ресурсов, включая память, данные и вычислительную мощность;
- частые ложные срабатывания, которые нарушают работу и в целом снижают эффективность таких систем;
- организованные атаки на основе *Искусственного интеллекта* (семантические вирусы).

Организация информационной безопасности в интеллектуальных системах нового поколения

Определим цели обеспечения информационной безопасности систем нового поколения.

Из монографии *Остроух А.В.ИнтелСМ-2020кн* ***Целями обеспечения информационной безопасности традиционных интеллектуальных систем*** являются:

- обеспечение конфиденциальности информации в соответствии с проведенной классификацией;
- обеспечение целостности информации на всех этапах, связанных с нею процессов (создание, обработка, хранение, передача и уничтожение) при предоставлении публичных услуг;
- обеспечение своевременной доступности информации при предоставлении публичных услуг;
- обеспечение наблюдаемости, направленной на фиксирование любой деятельности пользователей и процессов;
- обеспечение аутентичности и невозможности отказа от транзакций и действий, производимых участниками предоставления публичных услуг;

- учет всех процессов и событий, связанных с вводом, обработкой, хранением, предоставлением и уничтожением данных.

Следует отметить так как интеллектуальные системы нового поколения будут взаимодействовать с подобными себе системами понимая при этом, о чем осуществляется запрос, то цели обеспечения будут выглядеть по-другому. *Целями обеспечения информационной безопасности интеллектуальных систем нового поколения* являются:

- обеспечение сохранности семантической совместимости информации;
- защита достоверности и целостности информации;
- обеспечение доступности информации на разных уровнях интеллектуальной системы;
- минимизация ущерба от событий, несущих угрозу информационной безопасности.

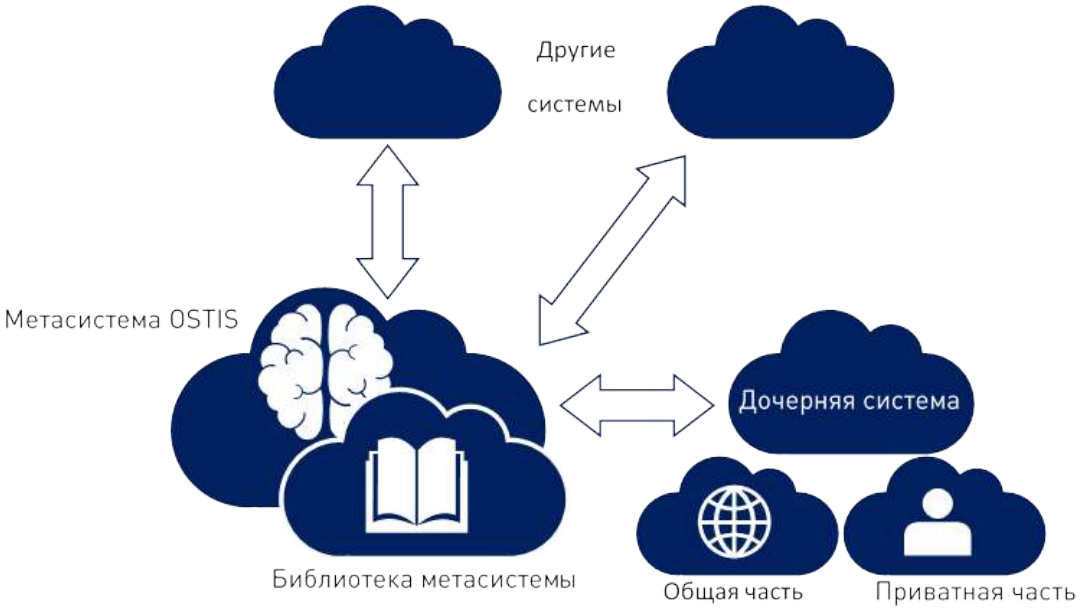
В настоящее время разработаны классические подходы и принципы обеспечения безопасности баз знаний (данных), интерфейсов связи (обмена информацией) между компонентами интеллектуальных систем такие, как шифрование передаваемых данных, фильтрация ненужного (избыточного) контента и политика разграничения доступа к данным.

Система обеспечения информационной безопасности должна создаваться на следующих принципах:

- **Принцип равнопрочности** — означает обеспечение защиты оборудования, программного обеспечения и системы управления от всех видов угроз;
- **Принцип непрерывности** — предусматривает непрерывное обеспечение безопасности информационных ресурсов системы для непрерывного предоставления публичных услуг;
- **Принцип разумной достаточности** — означает применение таких мер и средств защиты, которые являются разумными, рациональными и затраты на которые, не превышают стоимости последствий нарушения информационной безопасности;
- **Принцип комплексности** — для обеспечения безопасности во всем многообразии структурных элементов, угроз и каналов несанкционированного доступа должны применяться все виды и формы защиты в полном объеме;
- **Принцип комплексной проверки** — заключается в проведении специальных исследований и проверок, специального инженерного анализа оборудования, верификационных исследований программных средств. Должен осуществляться непрерывный мониторинг аварийных сообщений и параметров ошибок, постоянно должно выполняться тестирование аппаратного и программного оборудования, а также контроль целостности программных средств, как при загрузке программных средств, так и в процессе функционирования;
- **Принцип надежности** — методы, средства и формы защиты должны надежно перекрывать все пути проникновения и возможные каналы утечки информации, для этого допускается дублирование средств и мер безопасности;
- **Принцип универсальности** — меры безопасности должны перекрывать пути угроз независимо от места их возможного воздействия;
- **Принцип плановости** — планирование должно осуществляться путем разработки детальных планов действий по обеспечению информационной защищенности всех компонент системы предоставления публичных услуг;
- **Принцип централизованного управления** — в рамках определенной структуры должна обеспечиваться организованно-функциональная самостоятельность процесса обеспечения безопасности при предоставлении публичных услуг;
- **Принцип целенаправленности** — необходимо защищать то, что должно защищаться в интересах конкретной цели;
- **Принцип активности** — защитные меры обеспечения безопасности в работе процесса предоставления услуг должны претворяться в жизнь с достаточной степенью настойчивости;
- **Принцип квалификации обслуживающего персонала** — обслуживание оборудования должно осуществляться сотрудниками, подготовленными не только в вопросах эксплуатации техники, но и в технических вопросах обеспечения безопасности информации;
- **Принцип ответственности** — ответственность за обеспечение информационной безопасности должна быть ясно установлена, передана соответствующему персоналу и утверждена всеми участниками в рамках процесса обеспечения информационной безопасности.

§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем

Рисунок. Архитектура Экосистемы OSTIS
=



Структура *Экосистемы OSTIS* детально рассмотрена в *Главе 7.3. Структура Экосистемы OSTIS*. В *Экосистеме OSTIS* требуется организация обеспечения информационной безопасности на каждом из уровней: обмен данными, права доступа к данным, аутентификация клиентов Экосистемы, шифрование данных, получение данных из открытых источников, обеспечение достоверности и целостности хранимых и передаваемых данных, контроль за нарушением связей в базе знаний.

Важно отметить, что информационная безопасность тесно связана с архитектурой построенной системы: грамотно спроектированная и хорошо управляемая система сложнее поддается взлому. Поэтому очень важно разрабатывать систему информационной безопасности на этапе проектирования архитектуры и структуры будущей интеллектуальной системы нового поколения.

Экосистема OSTIS — это сообщество, где происходит взаимодействие *ostis-систем* и пользователей, где должны быть установлены правила, которые должны контролироваться. Нельзя допускать противоправные и дестабилизирующие действия со стороны всех участников сообщества. Пользователь не может на прямую осуществлять взаимодействия с другими *ostis-системами*, а только через персонального агента. Этот агент хранит все персональные данные пользователя и доступ к ним должен быть ограничен.

В *Экосистеме OSTIS* все агенты должны быть идентифицированы. Следует отметить, что персональный агент пользователя в Экосистеме решает проблему идентификации самого пользователя.

В рассмотренной *Экосистеме OSTIS* требуется организация обеспечения информационной безопасности на каждом из уровней взаимодействия: обмен данными, права доступа к данным, аутентификация клиентов Экосистемы, шифрование данных, получение данных из открытых источников, обеспечение достоверности и целостности хранимых и передаваемых данных, контроль за нарушением связей в базе знаний, отслеживание уязвимостей в системе.

угроза в ostis-системе
- ⊃ *угроза. нарушение конфиденциальности информации*
  - ⇒ *пояснение**:
    [несанкционированное получение доступа к чтению информации]
- ⊃ *угроза. нарушение целостности информации*
  - ⇒ *пояснение**:
    [несанкционированное или ошибочное изменение, искажение или уничтожение информации, а также несанкционированные воздействия на технические и программные средства обработки информации]
- ⊃ *угроза. нарушение доступности*
  - ⇒ *пояснение**:

[блокирование доступа к системе, отдельным ее компонентам, функциям или информации, а также невозможность своевременного получения информации (неприемлемые задержки в получении информации)]

⊃ *угроза. нарушение семантической совместимости*
⇒ *пояснение**:
[нарушение общности понятий и в общности базовых знаний]

⊃ *угроза. разрушение семантики баз знаний (семантические вирусы)*
⇒ *пояснение**:
[подмена или удаление узлов и связей между ними в базе знаний]

⊃ *угроза. избыточный объем входящей информации*

⊃ *угроза. нарушение неотказуемости*
⇒ *пояснение**:
[выдача несанкционированных действий за легальные, а также сокрытие или подмена информации о действиях субъектов]

⊃ *угроза. нарушение подотчетности*
⇒ *пояснение**:
[несанкционированное или ошибочное изменение, искажение или уничтожение информации о выполнении действий субъектом]

⊃ *угроза. нарушение подлинности (аутентичности)*
⇒ *пояснение**:
[выполнение действий в системе от имени другого лица или выдача недостоверных ресурсов (в том числе и данных) за подлинные]

⊃ *угроза. нарушение достоверности*
⇒ *пояснение**:
[преднамеренное или непреднамеренное предоставление и использование ошибочной (неправильной) или неактуальной (на конкретный момент времени) информации, а также выполнение процедур в нарушении регламента (протокола)]

Представим основные направления обеспечения информационной безопасности *ostis-систем* по предупреждению возникающих угроз:

- ограничение информационного трафика, анализируемого интеллектуальной системой;
- политика разграничении доступа к базе знаний;
- связность;
- введение семантической метрики;
- семантическая совместимость;
- активность.

Следует отметить, что на этапе проектирования самой *Технологии OSTIS* уже были заложены основные принципы обеспечения информационной безопасности, в рамках проектирования отдельных компонентов системы. Так уже изначально поддержка семантической совместимости и связности обеспечиваются в *ostis-системах* за счет способности системы обнаруживать вредоносные процессы в базе знаний.

Ограничение информационного трафика, анализируемого интеллектуальной системой

Экспоненциальный рост объема информации, циркулирующей в информационных потоках и ресурсах в условиях вполне определенных количественных ограничений на возможности средств ее восприятия, хранения, передачи и преобразования формирует новый класс угроз информационной безопасности, характеризуемых избыточностью совокупного входящего информационного трафика интеллектуальных систем.

В результате переполнение информационных ресурсов интеллектуальной системы избыточной информацией может спровоцировать распространения искаженной (деструктивной семантической) информации. Общая методология защиты интеллектуальных систем от избыточного информационного трафика осуществляется посредством использования аксиологических фильтров, реализующих функции численной оценки ценности поступающей информации, отбора наиболее ценной и отсеивания (фильтрации) менее ценной (бесполезной или вредной) с использованием вполне определенных критериев.

Следует также выделить в отдельную категорию угроз информационной безопасности активные средства разрушения семантики баз знаний (семантические вирусы) (см. *Баранович А.Е.СеманАИБКЗ-2011ст*).

Политика разграничении доступа к базе знаний

Мандатная политика безопасности (MAC — mandatory access control) основывается на мандатном (принудительном) разграничении доступа, определяющемся четырьмя условиями: все субъекты и объекты системы идентифицируются; задается решетка уровней безопасности информации; каждому объекту системы присваивается уровень безопасности, определяющий важность содержащейся в нем информации; каждому субъекту системы присваивается уровень доступа, определяющий уровень доверия к нему в интеллектуальной системе. Кроме того, мандатная

политика имеет более высокую степень надежности. Реализация данной политики основывается на разработанном алгоритме определения согласованных уровней безопасности всех элементов онтологии.

Так как семантические базы знаний в отличие от реляционной базы данных позволяют выполнять правила для получения логических выводов, то для обеспечения безопасности данных актуальным является разработка алгоритмов и методов, с помощью которых можно будет получать только данные, имеющие уровни безопасности меньше уровней доступа субъектов их запросивших (см. *Хоанг В.К..РешенОЗвРПППБРсСБД-2013ст*).

Связность

Вся информация, хранимая в семантической памяти интеллектуальной системы, систематизирована в виде единой базы знаний. К такой информации относятся непосредственно обрабатываемые знания, интерпретируемые программы, формулировки решаемых задач, планы и протоколы решения задач, информация о пользователях, описание синтаксиса и семантики внешних языков, описание пользовательского интерфейса и многое другое (см. *Голенков В.В..СеманМПиОБ-2017ст*). В информационной базе знаний между фрагментами информации (единицами информации) должна быть предусмотрена возможность установления связей различного типа. Прежде всего, эти связи могут характеризовать отношения между информационными единицами. Нарушение связей приводит к неправильному логическому выводу, либо к получению ложных знаний, либо к несовместимости знаний в базе. Данное направление более детально рассмотрено в *Главе 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*.

Введение семантической метрики

На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее семантическую близость информационных единиц, то есть силу ассоциативной связи между информационными единицами (см. *Дементьев А.В.МетриСД-2022ст*). Его можно было бы назвать отношением релевантности для информационных единиц. Такое отношение дает возможность выделять в информационной базе знаний некоторые типовые ситуации. Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

Семантическая совместимость

Внутренняя семантическая совместимость между компонентами интеллектуальной компьютерной системы (максимально возможное введение общих, совпадающих понятий для различных фрагментов хранимой базы знаний), являющаяся формой конвергенции и глубокой интеграции внутри интеллектуальной компьютерной системы для различного вида знаний и различных моделей решения задач, что обеспечивает эффективную реализацию мультимодальности интеллектуальной компьютерной системы. Внешняя семантическая совместимость между различными интеллектуальными компьютерными системами, выражающаяся не только в общности используемых понятий, но и в общности базовых знаний и являющаяся необходимым условием обеспечения высокого уровня социализации интеллектуальных компьютерных систем (см. *Golenkov V.V..Metho aTfECoC-2019art*). Более подробно рассмотрено в *Главе 1.2. Интеллектуальные компьютерные системы нового поколения*.

Активность

В интеллектуальной системе для актуализации тех или иных действий способствуют знания, имеющиеся в этой системе. Таким образом, выполнение активностей в интеллектуальной системе должно инициироваться текущим состоянием информационной базы знаний. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы (см. *Дружинин В.Н..КогниПУдВ-2002кн*). В том числе преднамеренное искажение информации и связей может стать источником преднамеренного искажения информации.

Для интеллектуальных систем нового поколения можно выделить ряд аспектов, в рамках которых требуется работка новых алгоритмов и методов обеспечения информационной безопасности в дополнении к существующим механизмам:

- многоуровневый доступ к отдельным частям базы знаний, так как информация бывает общедоступной, персональной, конфиденциальной;
- мониторинг изменений значений слов с течением времени, а также значений перевода с иностранного языка которые могут влиять на принимаемые решения;
- защиты от несанкционированного использования путем применения криптосемантических шифров;
- постоянный мониторинг уязвимостей в системе;
- протоколирование действий (взаимодействий) системы.

Для решения поставленных задач может быть применена экспертная *ostis-система*, способная обеспечить обнаружение злоупотреблений и аномалий в поведении всех участников *Экосистемы OSTIS* на основе постоянного мониторинга и введения протоколов взаимодействий участников.

Создание и применение экспертных систем является одним из важных этапов развития информационных технологий и информационной безопасности (см. *Созинова Е.Н.ПримеЭСдАиОИБ-2011ст*). Соответственно, решение задач обеспечения информационной безопасности может быть получено на базе использования экспертных систем:

- появляется возможность решения сложных задач с привлечением нового, специально разработанного для этих целей математического аппарата (семантических сетей, фреймов, нечеткой логики);
- применение экспертных систем позволяет значительно повысить эффективность, качество и оперативность решений за счет аккумуляции знаний;

Заключение к Главе 7.9.

Для эффективной информационной защиты системы на современном этапе необходим симбиоз традиционных технологий, и технологий, реализуемых в рамках *OSTIS*. Также следует отметить, что обеспечение информационной безопасности на базе *Технологии OSTIS* осуществляется значительно проще, потому что многие аспекты уже реализованы на этапе проектирования самой технологии. Важно отметить, что интеллектуальная информационная система нового поколения — это самостоятельный субъект, который может сам осознанно, целенаправленно и постоянно заботиться о себе, в том числе о своей собственной безопасности.

Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии

:= [Заключение к Монографии OSTIS-2023]

⇒ *автор**:
- *Голенков В. В.*
- *Гулякина Н. А.*

⇒ *подраздел**:
- *Особенность текущего состояния работ в области Искусственного интеллекта — переход к интеллектуальным компьютерным системам нового поколения*
- *Задачи текущего этапа разработки теории и технологии интеллектуальных компьютерных систем нового поколения*
- *Методологические проблемы текущего этапа работ в области Искусственного интеллекта*
- *Предпосылки перехода к интеллектуальным компьютерным системам нового поколения*
- *Предыстория и история разработки Технологии OSTIS*
- *Особенности, достоинства и новизна Технологии OSTIS*
- *Актуальные проекты текущего этапа работ по развитию Технологии OSTIS*

⇒ *ключевой знак**:
- *интеллектуальная компьютерная система нового поколения*
  := [компьютерная система нового поколения]
- *самообучаемая интеллектуальная компьютерная система*
- *интероперабельная интеллектуальная компьютерная система*
  := [интероперабельная компьютерная система]
- *индивидуальный субъект*
- *коллектив субъектов*
  := [коллекивный субъект]
  ⊂ *многоагентная система*
- *иерархический субъект*
- *социальная ответственность^*
- *интероперабельность^*
- *индивидуальная деятельность*
- *коллективная деятельность*
- *интеллект коллектива субъектов^*
  := [эффективность (качество) коллектива субъектов]
- *стратегическая задача субъекта**
  := [стратегическая цель данного субъекта*]
- *подзачада**
- *Общество*
  := [Человеческое сообщество]
- *Экосистема OSTIS*
  := [Вариант реализации Общества 5.0]
  := [Общество + надстройка в виде Глобальной иерархической системы взаимосвязанных ostis-систем]

⇒ *библиографическая ссылка**:
- *Golenkov V..otCurreSaCoAI-2022art*
- *Golenkov V.V..NextGICS-2022art*
- *Тарасов В.Б.оМногоСкИОФ-2002кн*
- *Golenkov V.V..MethoPotCSoW-2021art*
- *Golenkov V.V..tStand oICSaaK-2020art*
- *Golenkov V.V..Metho aTfECoC-2019art*

Особенность текущего состояния работ в области Искусственного интеллекта — переход к интеллектуальным компьютерным системам нового поколения

Эпицентром современного этапа автоматизации человеческой деятельности является низкий уровень автоматизации и большие накладные расходы

- на системную интеграцию различных компьютерных систем, то есть на создание сложных иерархических компьютерных комплексов;
- на модернизацию компьютерных систем в ходе их эксплуатации.

Для автоматизации этих аспектов человеческой деятельности у современных компьютерных систем явно не хватает *интеллекта* и *самостоятельности*.

Необходимость перехода от современных *компьютерных систем* (в том числе, и от современных *интеллектуальных компьютерных систем*) к *интеллектуальным компьютерных системам нового поколения* обусловлена необходимостью перехода к автоматизации все более и более сложных видов и областей *человеческой деятельности* требующих создания целых комплексов *интеллектуальных компьютерных систем*, способных самостоятельно эволюционировать и эффективно взаимодействовать между собой в процессе *коллективного решения сложных задач*.

Компьютерные системы, обладающие указанными способностями, и представляют собой *компьютерные системы нового поколения*. Поскольку указанные *компьютерные системы* не могут не иметь высокого уровня *интеллекта*, следует их также называть *интеллектуальными компьютерными системами нового поколения*. Высокий уровень *интеллекта* компьютерным системам нового поколения необходим:

- для адекватной оценки собственной компетенции и компетенции своих партнеров;
- для обеспечения взаимопонимания, договороспособности и координации (согласованности) своих действий с действиями партнеров в ходе *коллективного решения сложных задач* в условиях возможного возникновения непредсказуемых (нештатных) обстоятельств.

Очевидно, что для создания и эксплуатации *интеллектуальных компьютерных систем нового поколения* необходимо:

- разработать общую формальную теорию таких систем;
- разработать комплексную технологию проектирования и поддержки последующих этапов жизненного цикла этих систем;
- разработать общую формальную теорию всего многообразия видов и областей *человеческой деятельности*, которые целесообразно автоматизировать.

интеллектуальная компьютерная система нового поколения
$=$ (*самообучаемая интеллектуальная компьютерная система* $\cap$ *интероперабельная интеллектуальная компьютерная система*)

самообучаемая интеллектуальная компьютерная система
$:=$ **[***интеллектуальная компьютерная система*, имеющая высокие темпы самостоятельно реализуемой эволюции, следствием чего является существеное снижение трудоемкости (затрат, накладных расходов) на ее модернизацию**]***

самообучаемость интеллектуальной компьютерной системы
$\Rightarrow$ *предполагает**:

- способность мониторить состояние и динамику окружающей среды и корректировать свои действия при соответствующих изменениях окружающей среды (адаптивность);
- способность анализировать и повышать качество собственной базы знаний (структуризация и анализ противоречий, информационных дыр, информационного мусора);
- способность извлекать знания из внешних источников информации;
- способность анализировать и повышать качество собственной деятельности (в том числе способность учиться на собственных ошибках);
- способность анализировать качество деятельности других субъектов и извлекать из этого пользу для себя (учиться на чужих ошибках).

высокий уровень самообучаемости интеллектуальной компьютерной системы
$\Rightarrow$ *обеспечивается**:

- высоким уровнем гибкости интеллектуальной компьютерной системы
- высоким уровнем стратифицированности интеллектуальной компьютерной системы
- высоким уровнем рефлексивности интеллектуальной компьютерной системы

- высоким уровнем познавательной активности

интероперабельная интеллектуальная компьютерная система
:= [компьютерная система, способная к самостоятельному эффективному взаимодействию с другими системами]

интероперабельность интеллектуальной компьютерной системы
⇒ *предполагает**:
- способность к взаимопониманию с другими системами и ее пользователями
  ⇒ *предполагает**:
  семантическую совместимость с взаимодействующими системами и пользователями
- договороспособность
- способность к координации своих действий с действиями партнеров

В основе предлагаемого нами подхода к построению *интеллектуальных компьютерных систем нового поколения* лежат следующие принципы:
- смысловое представление знаний, хранимых в памяти *интеллектуальных компьютерных систем нового поколения*;
- онтологическая структуризация и систематизация хранимых в памяти знаний;
- децентрализованная ситуационная агенто-ориентированная организация *процессов решения задач*;
- конвергенция и глубокая (диффузная) интеграция различных моделей решения задач и, как следствие, гибридный характер *решателей задач*;
- смысловая интеграция входной информации, поступающей в индивидуальную интеллектуальную компьютерную систему извне по разным сенсорным каналам и на разных языках путем трансляции входной информации на общий универсальный *язык внутреннего смыслового представления знаний*.

следует отличать*
∋ {• *индивидуальная интеллектуальная компьютерная система нового поколения*
  • *коллективная интеллектуальная компьютерная система нового поколения*
  ⇒ *разбиение**:
  {• *коллектив индивидуальных интеллектуальных компьютерных систем нового поколения*
  • *иерархический коллектив интеллектуальных компьютерных систем нового поколения*
  := [коллектив интеллектуальных компьютерных систем нового поколения, членами которого могут быть как коллективные, так и *индивидуальные интеллектуальные компьютерные системы нового поколения*]
  }
}

индивидуальная интеллектуальная компьютерная система нового поколения
⇒ *особенности**:
- *индивидуальную интеллектуальную компьютерную систему нового поколения* невозможно декомпозировать на подсистемы, которые можно разрабатывать абсолютно независимо друг от друга и согласовывать только входам-выходам, реализуя принцип "черного ящика".
- В *индивидуальной интеллектуальной компьютерной системе нового поколения* необходима конвергенция, совместимость и "осмысленное" взаимодействие самых различных видов знаний и моделей решения задач. То есть *индивидуальная интеллектуальная компьютерная система нового поколения* должна быть *гибридной системой*.

Технология OSTIS
:= [Предложенная нами Технология разработки и сопровождения *интеллектуальных компьютерных систем нового поколения*]
:= [Open Semantic Technology for Intelligent Systems]
⇒ *предъявляемые требования**:
- комплексность — *Технология OSTIS* обеспечивает совместимость всех частных технологий *Искусственного интеллекта*; совместимость, самообучаемость и интероперабельность разрабатываемых *интеллектуальных компьютерных систем*, а также поддержку не только проектирования *интеллектуальных компьютерных систем*, но и всего их жизненного цикла
- универсальность — *Технология OSTIS* ориентирована на разработку и сопровождение *интеллектуальных компьютерных систем нового поколения* любого назначения
- самообучаемость — *Технология OSTIS* обеспечивает перманентную эволюцию самой *Технологии OSTIS* (самой себя) благодаря тому, что она реализована в виде *интеллектуальной компьютерной системы нового поколения*, которая "знает" *Технологию OSTIS* и "умеет" ее использовать

ostis-система
:= [*интеллектуальная компьютерная система*, построенная по *Технологии OSTIS*]

Экосистема OSTIS
:= [Основной продукт *Технологии OSTIS*, представляющий собой Глобальную сеть *ostis-систем*]
∈ *иерархический коллектив интеллектуальных компьютерных систем нового поколения*

Основными компонентами *Технологии OSTIS* являются:

- ***Стандарт OSTIS***
  := [Стандарт интеллектуальных компьютерных систем нового поколения, а также методик, методов и средств поддержки их жизненного цикла]
  := [*Стандарт Технологии OSTIS*]
- ***Метасистема OSTIS***
  := [Ядро Системы автоматизации поддержки жизненного цикла *ostis-систем*]
- ***Библиотека OSTIS***
  := [Распределенная библиотека типовых (многократно используемых) компонентов *ostis-систем*]

Задачи текущего этапа разработки теории и технологии интеллектуальных компьютерных систем нового поколения

Создание *интеллектуальных компьютерных систем нового поколения* требует получения ответов на следующие вопросы:

- Какие требования предъявляются к *интеллектуальным компьютерным системам*, обеспечивающим указанную выше комплексную автоматизацию *человеческой деятельности*;
- Почему современные интеллектуальные компьютерные системы указанным требованиям не удовлетворяют и, соответственно, почему необходим переход к принципиально новому поколению *интеллектуальных компьютерных систем*;
- Какие фундаментальные принципы должны лежать в основе *интеллектуальных компьютерных систем нового поколения*;
- Какие принципы должны лежать в основе максимально автоматизируемой технологии проектирования и поддержки всего жизненного цикла *интеллектуальной компьютерных систем нового поколения*;
- Какие принципы должны лежать в основе структуры и организации различных видов и областей *человеческой деятельности* для обеспечения ее комплексной и максимально возможной автоматизации с помощью *интеллектуальных компьютерных систем нового поколения* (как известно, прежде, чем автоматизировать какую-либо *человеческую деятельность*, необходимо привести ее в порядок — автоматизация беспорядка приводит к еще большему беспорядку).

К числу текущих фундаментальных задач по созданию теории и технологии *интеллектуальных компьютерных систем нового поколения* относятся:

- Разработка теории *иерархических многоагентных систем*, агентами в которых являются индивидуальные или коллективные *интероперабельные интеллектуальные компьютерные системы*.
- Унификация и стандартизация различных моделей представления и обработки знаний. Эффект от данной унификации будет виден не сразу. Но, если этого происходить не будет, мы никогда не придем к эффективной комплексной автоматизации *человеческой деятельности*. Эклектичное многообразие методов и средств автоматизации приводит не только к необоснованному дублированию разрабатываемых систем, но также и к повышению сложности их использования и сопровождения.
- Конвергенция и интеграция различных направлений *Искусственного интеллекта*.

  Сейчас различные направления *Искусственного интеллекта* имеют достаточно высокий уровень развития (signal processing, natural language processing, логические модели, искусственные нейронные сети, онтологические модели, многоагентные модели и многое другое). Интеграция всех этих направлений является пусть и достаточно трудоемкой задачей, но задачей вполне решаемой, в основе решения которой лежит согласование смежных понятий.
- Конвергенция таких видов деятельности в области *Искусственного интеллекта*, как:
  - подготовка специалистов в области *Искусственного интеллекта*;
  - инженерная деятельность по разработке прикладных *интеллектуальных компьютерных систем нового поколения*;
  - развитие *технологии* проектирования и поддержки жизненного цикла *интеллектуальных компьютерных систем нового поколения*;

- научно-исследовательская деятельность в области *Искусственного интеллекта*.

- Для развития Технологии *интеллектуальных компьютерных систем нового поколения* необходима также конвергенция этой Технологии со всеми видами и областями *человеческой деятельности*, которые не входят в состав деятельности в области *Искусственного интеллекта*. Развитие Технологии *интеллектуальных компьютерных систем нового поколения* носит ярко выраженный междисциплинарный характер. Это означает что все знания, накапливаемые *человеческим обществом* в самых различных областях должны быть представлены в составе Глобальной *базы знаний* Экосистемы *интеллектуальных компьютерных систем нового поколения* (с помощью порталов научно-технических, административных и прочих знаний), должны быть четко стратифицированы в виде иерархической системы семантически совместимых многократно используемых *онтологий* и превращены в иерархическую систему семантически совместимых *формальных* компонентов баз знаний *интеллектуальных компьютерных систем* различного прикладного назначения.

- Обеспечение семантической совместимости *интеллектуальных компьютерных систем нового поколения* не только на этапе их проектирования, но и на всех последующих этапах их жизненного цикла.

- Разработка модели коллективного поведения *интеллектуальных компьютерных систем нового поколения*, то есть модели децентрализованного коллективного решения задач на уровне:
  - *многоагентной системы*, агенты которой являются внутренними агентами *индивидуальной интеллектуальной компьютерной системы*, взаимодействующими через общую память (через общую *базу знаний*, хранимую в одной памяти);
  - *многоагентной системы*, агенты которой являются *интероперабельными интеллектуальными компьютерными системами*, взаимодействующими через общую *базу знаний*, хранимую в памяти *корпоративной интеллектуальной компьютерной системы* или в памяти координатора деятельности временного коллектива *интеллектуальных компьютерных систем*.

  В рамках теории *коллективного решения задач* можно выделить следующие задачные ситуации:
  - задача, которая может быть решена той *индивидуальной интеллектуальной компьютерной системой*, в которой эта задача инициирована;
  - задача, соответствующая компетенции того *коллектива интеллектуальных компьютерных систем*, в рамках которого эта задача инициирована;
  - задача, выходящая за пределы компетенции того *коллектива интеллектуальных компьютерных систем*, в рамках которого эта задача инициирована. Такая задача требует формирования временного коллектива, координатором (но не менеджером) которого становится та *интеллектуальная компьютерная система*, в рамках которой указанная задача инициировалась. Для этого необходимо найти те *интеллектуальные компьютерные системы*, которые в совокупности обеспечат необходимую компетенцию.

  Отметим при этом, что каждая *интероперабельная интеллектуальная компьютерная система* (как индивидуальная, так и коллективная) должна знать свою компетенцию для того, чтобы определить, сможет или не сможет она решить ту или иную заданную (возникшую) задачу. Это, в частности, необходимо для формирования временных коллективов *интеллектуальных компьютерных систем*.

- Разработка принципов, лежащих в основе мощной Библиотеки многократно используемых и совместимых компонентов *интеллектуальных компьютерных систем нового поколения*, которая обеспечивает полную автоматизацию интеграции этих компонентов в процессе сборки проектируемых систем.
- Разработка методик и средств перманентного расширения Библиотеки многократно используемых компонентов *интеллектуальных компьютерных систем нового поколения* в самых различных областях *человеческой деятельности*:
  - Научно-техническая деятельность в любой области должна сводиться к развитию *баз знаний* различных *интеллектуальных порталов научно-технических знаний*. При этом *база знаний* каждого такого портала должна декомпозироваться на фрагменты, включаемые в состав Библиотеки многократно используемых компонентов *баз знаний интеллектуальных компьютерных систем нового поколения*, которые могут иерархически входить друг в друга. Для этого указанные компоненты должны соответствующим образом специфицироваться.
  - Разработчики любой *интеллектуальной компьютерной системы* должны декомпозировать разработанную систему на множество компонентов, включаемых в состав Библиотеки компонентов *интеллектуальных компьютерных систем нового поколения* — так, чтобы разработка любой аналогичной системы свелась к сборке компонентов из этой Библиотеки.
  - Все(!) разработчики должны заботиться о расширении Библиотеки многократно используемых (типовых) компонентов *интеллектуальных компьютерных систем нового поколения*, что приведет к существенному снижению трудоемкости разработки новых *интеллектуальных компьютерных систем нового поколения* в рамках Экосистемы таких систем. При этом авторство компонентов указанной Библиотеки должно поощряться, что является фундаментальной основной развития рынка знаний, экономики знаний.

Если грамотно развивать и использовать Технологию *интеллектуальных компьютерных систем нового поколения*, то разработка любой новой *интеллектуальной компьютерной системы* будет в основном сводиться к ее автоматической сборке из указываемых разработчиком компонентов этой системы. Некоторые компоненты разрабатываемой *интеллектуальной компьютерной системы* могут входить в текущее состояние Библиотеки компонентов *интеллектуальных компьютерных систем нового поколения*, а некоторые из них будут требовать дополнительной разработки. Но при этом каждый такой новый компонент чаще всего является результатом модификации существующих компонентов из указанной Библиотеки и должен быть специфицирован и включен в эту Библиотеку. Таким образом разработчик прикладной *интеллектуальной компьютерной системы* должен разработать не только эту систему, но и внести вклад в развитие Библиотеки компонентов *интеллектуальных компьютерных систем нового поколения*, в результате которого разрабатываемая им следующая *интеллектуальная компьютерная система* может быть собрана без дополнительно разрабатываемых компонентов, а только из компонентов Библиотеки компонентов. Если все разработчики прикладных систем будут так действовать, то темпы повышения уровня автоматизации *человеческой деятельности* будут существенно возрастать.

Методологические проблемы текущего этапа работ в области Искусственного интеллекта

⇒ *эпиграф**:
[Самые сложные проблемы — это те, которые мы не осознаем и, особенно, те, причиной которых является наше несовершенство.]

⇒ *подраздел**:
- *Социальная ответственность специалистов в области Искусственного интеллекта*
- *Глобальная цель деятельности в области Искусственного интеллекта*
- *Общие требования, предъявляемые к специалистам в области Искусственного интеллекта*
- *Требования, предъявляемые к фундаментальной подготовке специалистов в области Искусственного интеллекта*
- *Проблемы текущего этапа разработки теории и технологии интеллектуальных компьютерных систем нового поколения*

Социальная ответственность специалистов в области Искусственного интеллекта

Современный этап развития теории и практики *Искусственного интеллекта* обнажает целый спектр проблем, препятствующих этому развитию. Дальнейшее развитие технологий *Искусственного интеллекта*

- с одной стороны, может и достаточно быстро осуществить переход современного общества на принципиально новый уровень его эволюции, обеспечивающий комплексную автоматизацию всех подлежащих автоматизации видов и областей *человеческой деятельности*, а также обеспечивающий максимально возможный комфорт и максимально возможное раскрытие творческого потенциала каждого человека;
- с другой стороны, может достаточно долго и весьма убедительно для неграмотного обывателя имитировать указанный прогресс автоматизации *человеческой деятельности* — любая даже весьма достойная цель может быть загублена имитацией ее достижения;
- с третьей стороны, может достаточно быстро привести *человеческое общество* к деградации и самоуничтожению.

Таким образом, на современном этапе развития технологий *Искусственного интеллекта*, уровень социальной ответственности специалистов в области *Искусственного интеллекта* является определяющим фактором развития *человеческого общества*. Опасность для *человеческого общества* исходит не от *интеллектуальных компьютерных систем*, а от мотивации специалистов, которые разрабатывают эти системы. Очевидно, что создание *интеллектуальных компьютерных систем*, предназначенных для осознанного нанесения любого ущерба *человеческому обществу*, и требующих создания соответствующих интеллектуальных средств обеспечения безопасности, является короткой дорогой к самоуничтожению.

Усилия специалистов в области *Искусственного интеллекта* должны быть направлены на существенное повышение уровня интеллекта *человеческого общества* в целом, основой чего является комплексная автоматизация всех тех видом и областей *человеческой деятельности*, которые принципиально имеет смысл автоматизировать.

Глобальная цель деятельности в области Искусственного интеллекта

Почему современный этап деятельности в области *Искусственного интеллекта* требует формулировки глобальной цели этой деятельности и перманентного ее уточнения.

Современное состояние *Искусственного интеллекта* можно охарактеризовать как глубокий методологический кризис, обусловленный:

- тем, что научные результаты в этой области вышли из научных лабораторий и стали оказывать реальное практическое воздействие;
- отсутствием понимания того, что получение серьезных научных результатов в той или иной области и создание технологий, обеспечивающих эффективное практическое использование этих результатов — это соизмеримые по значимости и сложности задачи. Особенно это касается *Искусственного интеллекта*.

Последнее обстоятельство приводит к неоправданной эйфории, иллюзии благоплучия и к бурно расцветающей эклектике, которая абсолютно игнорирует даже казалось бы очевидные законы общей теории систем.

К сожалению, локальное внедрение результатов научных исследований в области *Искусственного интеллекта*, локальная автоматизация бизнес-процессов какой-либо организации без учета системной организации всего комплекса методов и средств автоматизации различных видов и областей человеческой деятельности приводит к неоправданному дублированию результатов.

Если в ближайшее время не произойдет осознания глобальной (стратегической) цели работ в области *Искусственного интеллекта*, то деятельность в этой области в целом будет осуществляться в стиле "лебедя, рака и щуки". Трата усилий не приведет к целостному практически значимому результату. "Вектора" конкретных направлений этой деятельности, "вектора" наших усилий не будут иметь одинаковую направленность, что существенно снизит общую производительность всей этой деятельности и качество общего (суммарного) результата.

Какова же должна быть *стратегическая задача* (сверхзадача), которую должны решить специалисты в области *Искусственного интеллекта*. Очевидно, что такой сверхзадачей является переход всего комплекса *человеческой деятельности* на принципиально новый уровень максимально возможной его автоматизации, в рамках которого принципиально неавтоматизируемой частью человеческой деятельности остается *творческая* деятельность, в частности, научно-исследовательская деятельность, преподавательская и воспитательная деятельность, перманентное повышение уровня комплексной автоматизации *человеческой деятельности*. Основная цель комплексной автоматизации *человеческой деятельности* заключается не только в том, чтобы автоматизировать то, что можно эффективно автоматизировать с помощью методов *Искусственного интеллекта*, а в том, чтобы автоматизировать все(!) "узкие места" *человеческой деятельности*, которые определяют общую ее производительность в различных областях.

Таким образом, в настоящее время технологии *Искусственного интеллекта* находятся на пороге перехода к принципиально новому уровню развития — на пороге перехода от решения частных (локальных) задач к решению глобальной задачи комплексной автоматизации всех видов и областей *человеческой деятельности*, что требует автоматизации решения не только частных актуальных и важных задач, но и автоматизации решения задач все более и более высокого уровня, для которых автоматизируемые сейчас задачи становятся подзадачами. Другими словами, при автоматизации решения комплексных задач (надзадач) автоматизация фокусируется на разработке методов и средств взаимодействия между средствами решения локальных задач (частных задач).

Перенос акцента на автоматизацию решения не просто *интеллектуальных задач*, а на автоматизацию решения комплексных задач, подзадачами которых являются разнообразные *интеллектуальные задачи*, не только переводит технологии *Искусственного интеллекта* на принципиально новый уровень, но и окажет существенное влияние на все стороны *человеческой деятельности*:

- научно-исследовательские и научно-технические работы должны приобрести конвергентный взаимообогащающий характер;
- основой образования должна стать междисциплинарность;
- основой глобальной автоматизации *человеческой деятельности* должна стать общая комплексная формальная и перманентно совершенствуемая теория *человеческой деятельности*, в основе которой должна лежать междисциплинарная конвергентная методика, направленная на преодоление эклектичного подхода.

Следовательно, основной целью комплексной автоматизации всевозможных видов и областей человеческий деятельности с помощью *интероперабельных интеллектуальных компьютерных систем* является существенное повышение *уровня интеллекта* человеческого общества в целом.

Современное *человеческое общество* — это сложнейшая распределенная многоагентная *кибернетическая система*, развитие которой осуществляется, к сожалению, с нарушением многих законов Кибернетики и, в частности, с нарушением критериев, определяющих уровень *интеллекта* иерархических многоагентных систем. Уровень *интеллекта* таких систем определяется целым рядом казалось бы очевидных факторов:

- тем, каков объем и *качество знаний*, накопленных *многоагентной системой* и доступных всем агентам (субъектам), входящим в эту систему

  - насколько этих *знаний* достаточно для организации управления деятельностью этой системы;
  - насколько эти знания корректны (не противоречивы) и адекватны;
  - насколько велика конвергентность, компактность и чистота этих *знаний* (здесь учитывается наличие информационного мусора, информационного дублирования);
  - насколько хорошо структурированы (систематизированы) накапливаемые знания;
- тем, как осуществляется доступ каждого агента *многоагентной системы* к знаниям, хранимым в общей памяти всей *многоагентной системы*;
- тем, как эти *знания* накапливаются и эволюционируют, как многоагентная система самообучается
  - как *многоагентная система* учится на собственных ошибках,
  - как *многоагентная система* повышает качество своих *знаний*;
- тем, как *многоагентная система* в целом и каждый агент в частности используют накопленные в общей памяти *знания* для решения различных *задач*.

Таким образом, если рассматривать современное *человеческое общество* с позиций теории *многоагентных систем*, являющихся сообществами *интеллектуальных систем* (не только искусственных, но и естественных интеллектуальных систем), то очевидно, что следующий этап его эволюции требует:
- автоматизации накопления, анализа и перманентного повышения ***качества знаний***, накапливаемых человечеством;
- автоматизации эффективного использования накопленных человечеством знаний при решении задач самого различного уровня, требующих формирования различных кратковременных или долговременных ***сообществ из людей и интеллектуальных компьютерных систем***. Каждое такое сообщество предназначается либо для решения какой-либо одной конкретной задачи, либо для решений некоторого множества задач в некоторой области;
- повышения уровня ***конвергенции*** знаний, методов, действий, а также создаваемых новых технических систем;
- повышения уровня ***интероперабельности*** как для интеллектуальных компьютерных систем, так и для людей.

Общие требования, предъявляемые к специалистам в области Искусственного интеллекта

⇒ *эпиграф**:
  - Требования, предъявляемые к специалистам в области *Искусственного интеллекта* на новом этапе развития этой области, являются отражением требований, предъявляемых к *интеллектуальным компьютерным системам нового поколения* и соответствующим технологиям
  - Уровень *интеллекта* (в том числе коллективного интеллекта) разработчиков *интеллектуальной компьютерной системы* не может быть ниже уровня *интеллекта* создаваемых *интеллектуальных компьютерных систем*
  - Уровень *интеллекта* коллектива агентов далеко не всегда выше уровня *интеллекта* входящих в него агентов
  - Разработка *интероперабельных интеллектуальных компьютерных систем* может быть только коллективной
  - Коллектив неинтероперабельных разработчиков не может создавать *интероперабельные интеллектуальные компьютерные системы*

Высокий уровень *социальной ответственности*, требуемый от *специалистов в области Искусственного интеллекта*, предъявляет к ним целый ряд очевидных, но, к сожалению, часто не учитываемых общих требований, необходимых для качественного участия в сложных коллективных социально значимых проектах. К таким общим требованиям относятся:
- высокий уровень ***мотивации*** к участию в перманентной эволюции целостного технологического комплекса, обеспечивающего разработку эффективных *интероперабельных интеллектуальных компьютерных систем*. Комплексная и высококачественная технология разработки и сопровождения *интероперабельных интеллектуальных компьютерных систем* должна рассматриваться как ключевой продукт коллективной деятельности в области *Искусственного интеллекта*. Указанная мотивация предполагает соответствующую целеустремленность, отсутствие эгоизма, высокомерия, индивидуализма, изоляционизма, паразитизма;
- высокий уровень ***созидательной активности***, пассионарности, смелости;
- высокий уровень ***рефлексии*** — способности анализировать собственные цели и действия и исправлять собственные ошибки, а также анализировать цели, действия и ошибки, совершаемые коллективом, членом которого специалист является. Одно дело искренне признавать логичность и целесообразность соблюдения тех или иных правил (принципов, требований) и совсем другое дело уметь видеть и исправлять собственные нарушения этих правил. Без такой рефлексии прогресс коллективного творчества невозможен. Знать то, как надо делать и реально следовать этому — не одно и то же.

- высокий уровень ***собственной интероперабельности***:
  - способности к *взаимопониманию* и обеспечению *семантической совместимости*, требующей перманентного мониторинга текущего состояния и эволюции технологического комплекса;
  - *договороспособности* — способности оперативно согласовывать свои цели и планы, детонационную семантику понятий и терминов, а также децентрализовано распределять подзадачи коллективно решаемой задачи;
  - *способности координировать* и синхронизировать свои действия с коллегами в условиях возможного возникновения непредсказуемых обстоятельств.

Без высокого уровня *интероперабельности* разработчиков, невозможно обеспечить:

- ***конвергенцию***, унификацию, стандартизацию *интероперабельных интеллектуальных компьютерных систем*;
- формирование мощной ***Библиотеки типовых компонентов интеллектуальных компьютерных систем нового поколения***;
- существенное снижение трудоемкости и повышение уровня автоматизации разработки и сопровождения *интеллектуальных компьютерных систем нового поколения*;
- построение общей теории ***Экосистемы интеллектуальных компьютерных систем нового поколения*** и, соответственно, общей теории *человеческой деятельности*.

Таким образом, для создания *интероперабельных интеллектуальных компьютерных систем* необходимо. чтобы сами их создатели имели высокий уровень *интероперабельности*. Проблема обеспечить это является основным вызовом, который адресован специалистам в области *Искусственного интеллекта* на текущем этапе развития этой области.

Основной причиной, препятствующий формированию необходимого уровня *интероперабельности* у специалистов в области *Искусственного интеллекта*, является конкурентный стиль взаимоотношений между специалистами. Этот стиль взаимоотношений является широко распространенным способом стимулировать активность сотрудников. Но это не единственный способ стимулировать творческую активность в решении стратегически важных задач, каковой, в частности, является задача эффективной комплексной автоматизации всех видов и областей *человеческой деятельности* с помощью *интероперабельных интеллектуальных компьютерных систем*. Более того, конкуренция провоцирует эгоизм и игнорирование интересов иных субъектов (в том числе, и интересов того коллектива, членом которого субъект является). Таким образом конкуренция явно противоречит принципам *интероперабельности* и, соответственно, принципам организации *интеллектуальных сообществ*, интеллектуальных творческих коллективов и организаций.

От конкурентного стиля взаимоотношений необходимо переходить к взаимовыгодному взаимодействию между субъектами всех уровней иерархии. В этом заключается основная суть *интероперабельности* и перехода к *интеллектуальным коллективам* и интеллектуальному обществу.

Заметим, что перечисленные общие требования, предъявляемые к специалистам в области Искусственного интеллекта на современном этапе развития *технологий Искусственного интеллекта*, должны предъявляться не только к ним, но и ко всем людям, готовым способствовать технологическому прогрессу. Просто на данном этапе основная ответственность за это лежит именно на специалистах в области *Искусственного интеллекта*.

Требования, предъявляемые к фундаментальной подготовке специалистов в области Искусственного интеллекта

Необходимость существенного повышения уровня практической значимости и эффективности работ в области *Искусствтеного интеллекта*, требующего перехода к *интеллектуальным компьютерным системам нового поколения* и к принципиально новому технологическому комплексу, предъявляет к специалистам в области *Искусственного интеллекта* не только общие требования, необходимые для эффективного участия в сложных коллективных социально значимых проектах, но так же и высокие требования к их фундаментальной профессиональной подготовке:

- высокому уровню системной культуры, позволяющей "видеть" иерархию сложных систем, связи между различными уровнями и иерархии, разницу между тактическими и стратегическими задачами;
- высокому уровню математической культуры, культуры формализации;
- высокому уровню технологической культуры и технологической дисциплины;
- высокому уровню самообучаемости в условиях быстрого изменения технологической инфраструктуры

Проблемы текущего этапа разработки теории и технологии интеллектуальных компьютерных систем нового поколения

Перечислим основные методологические проблемы текущего этапа работ в области Искусственного интеллекта, которые препятствуют решению рассматриваемых выше фундаментальных задач:

- Недостаточно высокий уровень осознания специалистами в области *Искусственного интеллекта* своей социальной ответственности.
- Отсутствие согласованного осознания глобальной цели работ в области *Искусственного интеллекта*, которая заключается в поэтапном повышении ***уровня интеллекта*** *человеческого общества* путем комплексной автоматизации всех аспектов его деятельности с помощью сети взаимодействующих между собой *интеллектуальных компьютерных систем*.
- Недостаточно высокий уровень *интероперабельности* специалистов в области *Искусственного интеллекта* и преобладание конкурентного стиля взаимоотношений. Следствием этого является недостаточное количество мотивированных специалистов в области *Искусственного интеллекта*, способных к эффективному творческому взаимодействию. Для того, чтобы они появились в достаточном количестве, хорошей системы их профессиональной подготовки недостаточно. Следует также отметить, что хорошие человеческие отношения, психологическая атмосфера и Team Building в команде разработчиков, к которому серьезно относятся многие компании, является необходимым, но далеко не достаточным условием результативности коллективной разработки сложных компьютерных систем (особенно это касается *интеллектуальных компьютерных систем нового поколения*).
- Недостаточно высокий уровень комплексной фундаментальной подготовки специалистов в области *Искусственного интеллекта*.
- Ярко выраженный междисциплинарный характер *Искусственного интеллекта* как области человеческой деятельности, требующий от специалистов умения работать на стыках наук.
- Отсутствие осознания необходимости глубокой *конвергенции* между различными направлениями *Искусственного интеллекта* и формализации всего комплекса знаний в области *Искусственного интеллекта* для их использования в *базах знаний* интеллектуальных компьютерных систем (прежде всего, инструментальных *интеллектуальных компьютерных систем*, входящих в состав технологического комплекса разработки и сопровождения *интеллектуальных компьютерных систем* различного назначения).
- Высокий уровень сложности комплексной формализации всех накапливаемых человечеством знаний (прежде всего, в области математики и общей теории систем) и их конвергенции с комплексом знаний, накапливаемых и формализуемых в области *Искусственного интеллекта*. Это необходимо для непосредственного использования накапливаемых человечеством знаний в *интеллектуальных компьютерных системах* различного назначения.
- Отсутствие осознания необходимости глубокой *конвергенции* и согласованности между
  - научно-исследовательской деятельностью в области *Искусственного интеллекта*;
  - деятельностью, направленной на развитие частных технологий *Искусственного интеллекта*, а также комплексной технологии проектирования и поддержки жизненного цикла *интеллектуальных компьютерных систем*;
  - инженерной деятельностью, направленной на разработку конкретных *интеллектуальных компьютерных систем* различного назначения;
  - образовательной деятельностью, направленной на подготовку специалистов в области *Искусственного интеллекта*.
- Проблема обеспечения ***семантической совместимости*** *интеллектуальных компьютерных систем нового поколения* не только на этапе их проектирования, но и на протяжении всего их жизненного цикла в условиях перманентной эволюции самих *интеллектуальных компьютерных систем* в ходе их эксплуатации, а также перманентной эволюции комплексной технологии их разработки.

Основная часть указанных проблем заключается в необходимости перехода к принципиально новому стилю и организации взаимодействия специалистов в области *Искусственного интеллекта*, без чего невозможен переход от частных теорий *Искусственного интеллекта* к ***Общей теории интеллектуальных компьютерных систем***, обеспечивающей совместимость всех частных теорий *Искусственного интеллекта*, а также переход от частных технологий *Искусственного интеллекта* к ***Комплексной технологии Искусственного интеллекта***, обеспечивающей совместимость всех частных технологий *Искусственного интеллекта*. В основе перехода к новому стилю взаимодействия специалистов в области *Искусственного интеллекта* лежит переход от конкуренции к синергетическому взаимовыгодному взаимодействию, направленному на конвергенцию и глубокую интеграцию частных (локальных) результатов, что приведет к преобразованию современного сообщества специалистов в области *Искусственного интеллекта* в ***интеллектуальное сообщество*** (см. *Тарасов В.Б.оМногоСкИОФ-2002кн*).

Предпосылки перехода к интеллектуальным компьютерным системам нового поколения

- Активно расширяющееся многообразие информационных ресурсов и сервисов, эффективность использования которых имеет низкий уровень из-за отсутствия их систематизации и совместимости
- Появление формальных онтологий как средства обеспечения семантической совместимости накапливаемых человечеством информационных ресурсов, Semantic Web
- Активное развитие теории многоагентных систем, их самоорганизации, эмерджентности, синергии, теории интеллектуальных сообществ и организаций
- Развитие теории децентрализованного ситуационного управления ("оркестр играет без дирижера")
- Умный дом, умная больница, умный город
- Industry 4.0, University 4.0
- Появление работ, направленных на уточнение кибернетических принципов, лежащих в основе Общества 5.0

Предыстория и история разработки Технологии OSTIS

- 1981 – Японский и американский проекты ЭВМ пятого поколения
- 1984 – Защита В.В. Голенковым кандидатской диссертации “Структурная организация и переработка информации в электронных математических машинах, управляемых потоком сложноструктурированных данных”
- Совет Д.А. Поспелова: «Прежде, чем проектировать компьютеры, ориентированные на реализацию интеллектуальных компьютерных систем, необходимо:
  - разработать базовое математическое и программное обеспечение таких компьютеров;
  - разработать основы технологии проектирования интеллектуальных компьютерных систем, реализуемых на базе указанных компьютеров;
  - разработать на современных компьютерах программную модель (эмулятор) создаваемого компьютера нового поколения;
  - реализовать несколько конкретных *интеллектуальных компьютерных систем* на базе указанной выше технологии и указанной программной модели будущего компьютера.

  Если всего этого не сделать, то разработанный компьютер нового поколения будет талантливо сделанным "железом", которое непонятно как использовать и которое, следовательно, быстро морально устареет. Именно поэтому все проекты ЭВМ пятого поколения были обречены.»
- 1992 – Прототип семантического компьютера на транспьютерах
- 1995 – Открытие в БГУИР учебной специальности “Искусственный интеллект” и создание соответствующей выпускающей кафедры
- 1996 – Защита В.В. Голенковым докторской диссертации “Графодинамические модели и методы параллельной асинхронной переработки информации в интеллектуальных системах”
- 2010 – Создание открытого *Проекта OSTIS*, направленного на создание открытой комплексной технологии проектирования *интеллектуальных компьютерных систем*, реализация которых ориентируется на использование *компьютеров нового поколения*
- 2011 – Начало проведения ежегодных конференций OSTIS, направленных на развитие открытого *Проекта OSTIS*
- 2019 – На базе учреждения образования “Белорусский государственный университет информатики и радиоэлектроники” создано Учебно-научное объединение по направлению “Искусственный интеллект”;
- 2021 – Издание прототипа *Стандарта Технологии OSTIS*, представленного в виде формализованного текста, являющегося исходным текстом базы знаний Метасистемы поддержки проектирования интеллектуальных компьютерных систем, разрабатываемых по *Технологии OSTIS*;
- 2023 – Издание коллективной монографии по *Технологии OSTIS*, которая рассматривается как основа дальнейшего развития и официального признания формализованного *Стандарта Технологии OSTIS* и существенного расширения соответствующего авторского коллектива

Резюмируя наш опыт работ в области *Искусственного интеллекта*, можно сказать следующее:
- Требования, предъявляемые к *интеллектуальным компьютерным системам* следующего поколения (высокий уровень *самообучаемости*, *интероперабельности*, *самостоятельности*, *универсальности*), предполагает создание *принципиально новой комплексной* технологии, которая интегрирует, обеспечивает совместимость всего многообразия существующих *частных технологий Искусственного интеллекта* и которая поддерживает все этапы жизненного цикла разрабатываемых *интеллектуальных компьютерных систем*;
- Сложность реализации *интеллектуальных компьютерных систем нового поколения* (из-за несоответствия базовых принципов обработки информации в таких системах и принципов машины фон-Неймана, лежащих в основе современных компьютеров) требует создания компьютеров, специально ориентированных на реализацию *интеллектуальных компьютерных систем нового поколения*. Но создавать указанные компьютеры нового поколения необходимо на основе (а, точнее, в рамках) указанной выше комплексной технологии про-

ектирования и поддержки последующих этапов жизненного цикла интеллектуальных *компьютерных систем нового поколения*.

- Эпицентром создания и последующей эволюции указанной комплексной технологии для *интеллектуальных компьютерных систем нового поколения* является:
  - подготовка нового поколения специалистов в области *Искусственного интеллекта*, которые изначально ориентированы на конвергенцию, на обеспечение совместимости своих результатов с результатами своих коллег и на спецификацию своих результатов в рамках Библиотеки типовых (многократно используемых) компонентов;
  - перманентное развитие *Стандарта Технологии OSTIS*, представленного в виде формализованного текста *базы знаний* Метасистемы поддержки проектирования *интеллектуальных компьютерных систем*, разрабатываемых по *Технологии OSTIS*.

Особенности, достоинства и новизна Технологии OSTIS

Новизна *Технологии OSTIS* прежде всего заключается:

- в требованиях, предъявляемых к системам, создаваемым и сопровождаемым с помощью этой Технологии (к *интеллектуальным компьютерным системам нового поколения*) – гибридность, интероперабельность, самообучаемость.
- в требованиях, предъявляемых к самой *Технологии OSTIS* (к используемым ею методикам, автоматизируемым методам и средствам) – комплексность технологии, ее универсальность и самообучаемость

Дополнительные факторы новизны *Технологии OSTIS* заключаются:

- в том, что интенсивная эволюция самой *Технологии OSTIS* (переход на новые ее версии) не приводит к моральному старению уже эксплуатируемых *интеллектуальных компьютерных систем* (*ostis-систем*), поскольку в процессе эксплуатации этих систем возможна автоматическая их модификация (модернизация) в направлении их приведения в соответствие с текущей версией *Технологии OSTIS*;
- в том, что обеспечивается перманентная поддержка семантической совместимости эксплуатируемых *интеллектуальных компьютерных систем* (*ostis-систем*) в ходе их собственной эволюции, а также в ходе эволюции самой *Технологии OSTIS*;
- в том, что основой деятельности (функционирования) иерархических коллективов *ostis-систем* является децентрализованное планирование, инициирование и ситуационное управление коллективно выполняемыми действиями (процессами), осуществляемыми в рамках как долговременно существующих, так и временно существующих *коллективов ostis-систем*;
- в существенном повышении эффективности расширения и использования Библиотеки типовых (многократно используемых) компонентов *ostis-систем* (*Библиотеки OSTIS*) благодаря:
  - исключению семантической эквивалентности компонентов;
  - существенному сокращению многообразия логически и функционально эквивалентных компонентов;
  - наличию простой и достаточно легко автоматизируемой процедуры интеграции компонентов указанной библиотеки и, соответственно, процедуры сборки *ostis-систем* из готовых компонентов *Библиотеки OSTIS*;
- в ориентации на создание комплексной модели, обеспечивающий согласование всего многообразия видов и областей *человеческой деятельности* и на разработку архитектуры глобального комплекса *ostis-систем*, обеспечивающего автоматизацию указанного многообразия (*Экосистемы OSTIS*).

Актуальные проекты текущего этапа работ по развитию Технологии OSTIS

Перечислим некоторые актуальные на данном этапе проекты прикладных *интеллектуальных компьютерных систем нового поколения* и средства их разработки:

- Разработка формализованного ***Стандарта интеллектуальных компьютерных систем нового поколения***, представленного в составе *базы знаний* интеллектуального портала научно-технических знаний по теории *интеллектуальных компьютерных систем нового поколения* и обеспечивающего *семантическую совместимость компьютерных систем* этого класса.
- Разработка формализованного ***Стандарта методов и средств поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения***, представленного в составе *базы знаний* интеллектуальной *Метасистемы автоматизации поддержки жизненного цикла компьютерных систем нового поколения* (*Метасистемы OSTIS*).

- Разработка комплексной ***Библиотеки типовых компонентов интеллектуальных компьютерных систем нового поколения*** (*Библиотеки OSTIS*), обеспечивающей совместимость типовых (многократно используемых) компонентов и полную автоматизацию их интеграции (соединения) в процессе сборочного (компонентного) проектирования семантически совместимых *интеллектуальных компьютерных систем нового поколения*.

- В рамках ***Метасистемы OSTIS*** обеспечение широкого доступа к текущему состоянию *Стандарта OSTIS* и разработка соответствующих средств семантической визуализации и навигации.

- В рамках ***Метасистемы OSTIS*** разработка средств автоматизации и управления процессом коллективного совершенствования (модернизации, реинжиниринга) *Стандарта OSTIS*.

- Разработка ***программной платформы*** для реализации *интеллектуальных компьютерных систем нового поколения*.

- Разработка ***ассоциативного семантического компьютера*** для реализации *интеллектуальных компьютерных систем нового поколения*. Это универсальный компьютер, в котором осуществляется аппаратная реализация ассоциативной реконфигурируемой (структурно перестраиваемой) памяти, в которой переработка информации сводится к реконфигурации связей между элементами памяти.

- Разработка архитектуры *интеллектуальной компьютерной системы нового поколения*, которая является ***персональным интеллектуальным ассистентом*** (секретарем, референтом) для каждого пользователя, обеспечивающим максимально возможную автоматизацию процесса взаимодействия пользователя со всей Глобальной экосистемой *интеллектуальных компьютерных систем нового поколения* (*Экосистемой OSTIS*). *База знаний* каждого такого *персонального интеллектуального ассистента* включает в себя:
  - персональную информацию соответствующего пользователя, доступ к которой другим *интеллектуальным компьютерным системам* предоставляет персональный интеллектуальный ассистент этого пользователя, но обязательно с разрешения этого пользователя и с сообщением пользователю соответствующих факторов риска. Персональная информация пользователя — это его медицинские данные, биографические данные, личные фотографии, неопубликованная интеллектуальная собственность, формируемые или отправленные сообщения, адресуемые другим пользователям или различным сообществам.
  - информацию о различных сообществах *Глобальной экосистемы интеллектуальных компьютерных систем нового поколения*, членом которых является соответствующий (ассистируемый) пользователь, с указанием роли (должности, обязанности), которую выполняет указанный пользователь в рамках каждого такого сообщества. Указанных сообществ может быть много — профессиональные сообщества, друзья, родственники, сообщества потребителей-производителей, административно-гражданские сообщества, банки, сообщества медицинского обслуживания и другие.
  - информацию о собственных планах и намерениях (как о стратегических, так и о ближайших, включая встречи, переговоры, совещания)

  Решатель задач персонального интеллектуального ассистента
  - обеспечивает максимально возможную автоматизацию различных видов профессиональной индивидуальной деятельности соответствующего (обслуживаемого) пользователя;
  - обеспечивает интеллектуальное посредничество (представление интересов) обслуживаемого пользователя в рамках всех сообществ, в состав которых он входит.

  Пользовательский интерфейс персонального интеллектуального ассистента
  - предоставляет пользователю средства управления его индивидуальной деятельностью, осуществляемой совместно с соответствующим ему персональным интеллектуальным ассистентом;
  - обеспечивает унифицированный характер взаимодействия пользователей в рамках различных сообществ, в которые он входит. Простейшим видом сообществ является разовый диалог двух пользователей.

- Разработка унифицированного ***комплекса средств автоматизации индивидуального проектирования фрагментов баз знаний***, входящего в состав *персонального интеллектуального ассистента* каждого пользователя и обеспечивающего поддержку индивидуального вклада в развитие как собственной (персональной) *базы знаний*, так и *базы знаний* других систем, входящих в состав *Экосистемы интеллектуальных компьютерных систем*. В состав указанного комплекса средств автоматизации входят
  - редактор внутреннего представления знаний (редактор *sc-текстов*);
  - редакторы различных внешних форм представления знаний (*sc.g-текстов*, *sc.n-текстов*);
  - трансляторы с внутреннего представления знаний на различные внешние формы представления;
  - трансляторы с каждой формы внешнего представления знаний во внутреннее их представление;
  - средства синтаксического и семантического анализа проектируемого фрагмента *базы знаний*;
  - транслятор, обеспечивающий преобразование внутреннего представления знаний (в *SC-коде*) в естественно-языковое представление в формате языка разметки LaTeX, удовлетворяющее требованиям, предъявляемым к оформлению статей в сборниках научно-технических материалов. Данный транслятор позволит

сконцентрировать усилия разработчиков различных интеллектуальных компьютерных систем на формализацию научно-технических знаний, используемых в *интеллектуальных компьютерных системах*, и существенно снизить трудоемкость подготовки и оформления публикаций соответствующих научно-технических результатов.

В перспективе различные научно-технические журналы должны быть преобразованы в интеллектуальные порталы коллективно разрабатываемых научно-технических знаний в различных областях.

- Разработка в рамках персонального интеллектуального ассистента набора ***средств индивидуального комплексного перманентного медицинского контроля и мониторинга*** соответствующего (обслуживаемого) пользователя
- Разработка для каждого сообщества *интеллектуальных компьютерных систем нового поколения* унифицированного комплекса ***средств коллективной разработки общей базы знаний*** этого сообщества (*базы знаний* корпоративной системы указанного сообщества), в состав которого входят:
  - средства сборки (интеграции) разрабатываемой *базы знаний* из индивидуально разрабатываемых ее фрагментов;
  - средства согласования индивидуально разработанных фрагментов (персональных точек зрения, эпицентром чего является согласование используемых *понятий*);
  - средства взаимного рецензирования;
  - средства согласованной корректировки *базы знаний*;
  - средства формирования и согласования плана совершенствования коллективно разрабатываемой *базы знаний*;
  - средства контроля и управления процессом совершенствования коллективно разрабатываемой *базы знаний*.
- Расширение набора ***средств автоматизации проектирования*** различных видов компонентов *интеллектуальных компьютерных систем нового поколения* (*ostis-систем*) и различных классов таких систем.
- Разработка формальной структуры глобального комплекса автоматизируемой человеческой деятельности и соответствующей этому архитектуры ***Экосистемы OSTIS***. Существенное расширение направлений применения *Технологии OSTIS* (медицина, промышленность, строительство, юриспруденция и так далее).
- Разработка в рамках *Экосистемы OSTIS* ***комплекса средств и методик подготовки специалистов в области Искусственного интеллекта*** (на уровне обучения студентов, магистрантов и аспирантов).
- Разработка в рамках *Экосистемы OSTIS* ***комплекса средств информатизации среднего образования*** с помощью семантически совместимых *интеллектуальных компьютерных систем нового поколения*.
- Разработка в рамках *Экосистемы OSTIS* ***комплекса средств информатизации высшего технического образования*** с помощью семантически совместимых *интеллектуальных компьютерных систем нового поколения*.

Используемые сокращения и предметный указатель OSTIS

Предлагаемый вашему вниманию предметный указатель представляет собой алфавитный перечень основных терминов, используемых в данной монографии, которые взаимно-однозначно соответствуют элементам рафинированной семантической сети, представляющей собой базу знаний *Метасистемы OSTIS*, основная часть которой семантически эквивалентна тексту данной монографии.

В рамках внутреннего языка *ostis-систем* (в рамках *SC-кода*) указанные основные термины называются *основными sc-идентификаторами* (основными внешними идентификаторами sc-элементов — элементов рафинированных семантических сетей).

В данном предметном указателе в алфавитном порядке перечислены:

- все русскоязычные основные термины описываемых в монографии сущностей с указанием
  - их англоязычных эквивалентов;
  - тех разделов монографии, в которых эти термины являются ключевыми знаками;
- все интернациональные основные термины (например, названия различных программных систем, такие как *Neo4j*, *MySQL* и так далее), используемые в монографии с указанием
  - соответствующих разделов монографии, где эти термины являются ключевыми знаками.

При этом по умолчанию считается, что русскоязычный термин является основным русскоязычным идентификатором, а соответствующий ему англоязычный — основным англоязычным идентификатором.

Для каждого неосновного, но часто используемого русскоязычного термина указывается синонимичный ему основной русскоязычный термин. Кроме того, для каждого основного русскоязычного термина указывается ссылка на соответствующую главу, параграф или пункт монографии, где сущность, обозначаемая этим термином, является ключевым знаком.

абстрактная сущность
⇒ *трактовка**:
[трактовка этого термина имеет два аспекта:
- вымышленная (придуманная, реально несуществующая) сущность в отличие оо реальной (материальной) сущности, например, множество, пространственная точка;
- сущность, имеющая неоднозначную спецификацию описывающую только те свойства, которые важны (существенны) только для некоторой точки зрения. Например, абстрактная (виртуальная) машина, абстрактный (виртуальный) пациент.

]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

Абстрактная scp-машина
:= [*Abstract scp-machine*]
⇐ *ключевой знак**:
- *§ 3.3.5. Базовый язык программирования ostis-систем*

абстрактный sc-агент
:= [*abstract sc-agent*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

абстрактный sc-агент, не реализуемый на Языке SCP
:= [*abstract sc-agent not implemented in the SCP Language*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

абстрактный sc-агент, реализуемый на Языке SCP
:= [*abstract sc-agent implemented in the SCP Language*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

А/Б тестирование пользовательских интерфейсов
:= [*A/B testing of user interfaces*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

автоматизированная система управления технологическим процессом
:= [*automated technological process control system*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

агент Экосистемы OSTIS
:= [*OSTIS Ecosystem agent*]
⇐ *ключевое понятие**:
- *Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS*

адаптивное управление
:= [*adaptive control*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

адаптивный интерфейс
:= [*adaptive interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

адъюнкт
:= [*adjunct*]
⇐ *ключевое понятие**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

активный навык
:= [*active skill*]
⇐ *ключевое понятие**:
- *§ 3.1.5. Спецификация методов и понятие навыка*

алфавит
:= [*alphabet*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

Алфавит SC-кода^
:= [*SC-code Alphabet^*]
⇐ *ключевой параметр**:
- *Пункт 2.3.2.1. Синтаксис SCg-кода*

Алфавит SCg-кода^
:= [*SCg-code Alphabet^*]
⇐ *ключевой параметр**:
- *Пункт 2.3.2.1. Синтаксис SCg-кода*

Алфавит SCn-кода^
:= [*SCn-code Alphabet^*]
⇐ *ключевой параметр**:
- *Пункт 2.3.4.1. Синтаксис SCn-кода*

Алфавит SCs-кода^
:= [*SCs-code Alphabet^*]
⇐ *ключевой параметр**:
- *Пункт 2.3.3.1. Синтаксис SCs-кода*

анализ
:= [*analysis*]
⇐ *ключевое отношение**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

арифметическая операция*
:= [*arithmetic operation**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

арность
:= [*arity*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

архитектура вычислительной системы
:= [*computing system architecture*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

ассоциативный семантический компьютер
:= [*associative semantic computer*]
⇐ *ключевой знак**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

атомарное действие
:= [*atomic action*]
⇐ *ключевое отношение**:

- *§ 3.1.1. Понятие действия*

атомарное существование
:= [*atomic existence*]
⇐ *ключевое понятие**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

атомарный абстрактный sc-агент
:= [*atomic abstract sc-agent*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

атрибут отношения*
:= [*relation attribute**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

аудиоинтерфейс
:= [*audio interface*]
⇐ *ключевое понятие**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*

аудиосигнал
:= [*audio signal*]
⇐ *ключевое понятие**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*

база знаний
:= [*knowledge base*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*

база знаний ostis-системы
:= [*ostis-system knowledge base*]
⇐ *ключевое понятие**:
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

базовая ostis-платформа
:= [*basic ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

базовый класс описываемых сущностей
:= [*basic entity class*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

библиотека многократно используемых компонентов пользовательских интерфейсов ostis-систем
:= *сокращение основного sc-идентификатора**:
[библиотека м.и.к. п.и. ostis-систем]
:= [*ostis-systems reusable user interface components library*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

библиотека многократно используемых компонентов ostis-систем
:= *сокращение основного sc-идентификатора**:
[библиотека м.и.к. ostis-систем]
:= [*ostis-systems reusable components library*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Библиотека Метасистемы OSTIS
:= [*OSTIS Metasystem library*]
⇐ *ключевой знак**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Библиотека Экосистемы OSTIS
:= [*OSTIS Ecosystem library*]
⇐ *ключевой знак**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

бинарное отношение
:= [*binary relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

блокировка*
:= [*lock**]
⇐ *ключевое отношение**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

веб вещей
:= [*web of things*]
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

величина
:= [*value*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

верное высказывание*
:= [*correct statement**]
⇐ *ключевое отношение**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

вероятностный технологический процесс
:= [*probabilistic technological process*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

вершина
:= [*head*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

вид деятельности
:= [*activity type*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

вид знаний
:= [*knowledge type*]
⇐ *ключевое понятие**:
- *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*

включение*
:= [*inclusion**]
⇐ *ключевое отношение**:

- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

внешняя сущность
:= [*external entity*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

воздействие
:= [*manipulation*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

волновая микропрограмма
:= [*wave microprogram*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

волновой язык программирования
:= [*wave programming language*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

вопрос
:= [*question*]
⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*
- *Глава 3.4. Язык вопросов для ostis-систем*

временная связь
:= [*temporary connection*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

временная сущность
:= [*temporary entity*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

встроенная ostis-система
:= [*built-in ostis-system*]
⇐ *ключевое понятие**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

выводимое множество
:= [*inferrable set*]
⇐ *ключевое понятие**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

высказывание
:= [*statement*]
⇐ *ключевое понятие**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

высказывание*
:= [*statement**]
⇐ *ключевое отношение**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

геоинформационная система
:= [*geoinformation system*]
⇐ *ключевое понятие**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

геоонтология
:= [*geoontology*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

геосемантическая характеристика объекта местности
:= [*geosemantic characteristic of terrain object*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

глобальный нетранслируемый sc-идентификатор
:= [*global non-translatable sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента*

грамматическая категория
:= [*grammatical category*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

действие
:= [*action*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

действие в sc-памяти
:= [*action in sc-memory*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

действие в sc-памяти, инициируемое вопросом
:= [*action in sc-memory initiated by a question*]
⇐ *ключевое понятие**:
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

действие по построению искусственных нейронных сетей
:= [*artificial neural networks building action*]
⇐ *ключевое понятие**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

действие редактирования базы знаний
:= [*knowledge base editing action*]
⇐ *ключевое понятие**:
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

декартово произведение*
:= [*cartesian product**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

декларативная формулировка задачи
:= [*declarative problem definition*]
⇐ *ключевое понятие**:

- *§ 3.1.2. Понятие задачи*

декларативная формулировка задачи*
:= [*declarative problem definition**]
⇐ *ключевое отношение**:
- *§ 3.1.2. Понятие задачи*

декларативно-процедурная формулировка задачи
:= [*declarative-procedural problem definition*]
⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*

денотационная семантика метода*
:= [*denotational semantics of method**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.1.5. Спецификация методов и понятие навыка*

денотационная семантика языка
:= [*denotational semantics of language*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

денотационная семантика языка представления методов*
:= [*denotational semantics of method representation language**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Денотационная семантика SCg-кода
:= [*Denotational semantics of SCg-code*]
⇐ *ключевой знак**:
- *Пункт 2.3.2.2. Денотационная семантика SCg-кода*

Денотационная семантика SCn-кода
:= [*Denotational semantics of SCn-code*]
⇐ *ключевой знак**:
- *Пункт 2.3.4.2. Денотационная семантика SCn-кода*

Денотационная семантика SCs-кода
:= [*Denotational semantics of SCs-code*]
⇐ *ключевой знак**:
- *Пункт 2.3.3.2. Денотационная семантика SCs-кода*

деятельность
:= [*activity*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

дидактическая информация
:= [*didactic information*]
⇐ *ключевое понятие**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

дискретная информационная конструкция
:= [*discrete information construction*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

длительность^
:= [*duration^*]
⇐ *ключевой параметр**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

домен*
:= [*domain**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

древовидное тестирование пользовательских интерфейсов
:= [*tree testing of user interfaces*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

единица измерения*
:= [*measurement unit**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

естественно-языковой интерфейс
:= [*natural language interface*]
⇐ *ключевое понятие**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

жизненный цикл
:= [*life cycle*]
⇐ *ключевое понятие**:
- *Часть 1. Введение в интеллектуальные компьютерные системы нового поколения и технологию комплексной поддержки их жизненного цикла*

завершение^
:= [*completion^*]
⇐ *ключевой параметр**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

задача
:= [*problem*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

задача геоинформационной системы
:= [*geoinformation system problem*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

задача, решаемая в sc-памяти
:= [*problem solved in sc-memory*]
⇐ *ключевое понятие**:
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

знак
:= [*sign*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

знаковая конструкция
:= [*sign construction*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

знание
:= [*knowledge*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*

идентификатор
:= [*identifier*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

измерение*
:= [*measurement**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

измерение с фиксированной единицей измерения
:= [*measurement with fixed measurement unit*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

имя нарицательное
:= [*common name*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента*

имя собственное
:= [*proper name*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента*

индивидуальная кибернетическая система
:= [*individual cybernetic system*]
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

Индустрия 4.0
:= [*Industry 4.0*]
⇐ *ключевой знак**:
- *Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0*

инициируемое пользовательским интерфейсом действие*
:= [*action initiated by the user interface*]
⇐ *ключевое отношение**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интеграция
:= [*integration*]
⇐ *ключевое понятие**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

интеллект^
:= [*intelligence^*]
:= [уровень интеллекта^]
∈ *параметр*
⇒ *область определения**:
кибернетическая система
⇐ *ключевой параметр**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

интеллектуальная геоинформационная система
:= [*intelligent geoinformation system*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

интеллектуальная геоинформационная ostis-система

≔ [*intelligent geoinformation ostis-system*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

интеллектуальная кибернетическая система
≔ [*intelligent cybernetic system*]
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

интеллектуальная компьютерная система нового поколения
≔ *сокращение основного sc-идентификатора**:
[и.к.с. нового поколения]
≔ [*next-generation intelligent computer system*]
⇒ *определение**:
[интеллектуальная компьютерная система, обладающая:
- высоким уровнем самообучаемости, обеспечивающим высокий уровень автоматизации собственной эволюции и, соответственно, высокие темпы этой эволюции;
- высоким уровнем интероперабельности.

]
⊂ *самообучаемая интеллектуальная компьютерная система*
⊂ *интероперабельная интеллектуальная компьютерная система*
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

интеллектуальная обучающая система
≔ [*intelligent learning system*]
⇐ *ключевое понятие**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*

интеллектуальная справочная система
≔ [*intelligent help system*]
⇐ *ключевое понятие**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*

интеллектуальный интерфейс
≔ [*intelligent interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интервальная величина
≔ [*interval value*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

интернет вещей
≔ [*internet of things*]
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

интерпретатор пользовательских действий
≔ [*user action interpreter*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интерпретатор sc-моделей пользовательских интерфейсов
≔ [*user interface sc-model interpreter*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интерфейс
:= [*interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интерфейсное действие пользователя
:= [*interface action by user*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

интерфейс ostis-систем
:= [*интерфейс интеллектуальных компьютерных систем нового поколения*]
:= [*ostis-system interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

информационное действие
:= [*information action*]
⇐ *ключевое понятие**:
- *§ 3.1.1. Понятие действия*

информационная задача
:= [*information problem*]
⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*

информационная конструкция
:= [*information construction*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

информационный ресурс
:= [*information resource*]
⇐ *ключевое понятие**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

Искусственный интеллект
:= [*Artificial intelligence*]
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

картографический интерфейс
:= [*cartographic interface*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

качественный метод оценки пользовательских интерфейсов
:= [*qualitative method of user interface evaluation*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

квазибинарное отношение
:= [*quasybinary relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

квантор*
:= [*quantifier**]
⇐ *ключевое отношение**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

кибернетическая система
:= [*cybernetic system*]
⊃ *компьютерная система*
:= [искусственная кибернетическая система]
⊃ *интеллектуальная компьютерная система*
⊃ *интеллектуальная компьютерная система нового поколения*
= (*самообучаемая компьютерная система* ∩ *интероперабельная компьютерная система*)
⊃ *ostis-система*
:= [предлагаемое уточнение (вариант реализации) интеллектуальной компьютерной системы нового поколения]
⇐ *ключевое понятие**:
- *Часть 1. Введение в интеллектуальные компьютерные системы нового поколения и технологию комплексной поддержки их жизненного цикла*
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

класс
:= [*class*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

класс атомарных действий
:= [*atomic action class*]
⇐ *ключевое понятие**:
- *§ 3.1.3. Понятие класса действий и класса задач*

класс действий
:= [*action class*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

класс задач
:= [*problem class*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

класс легко выполнимых неатомарных действий
:= [*class of trivial non-atomic actions*]
⇐ *ключевое понятие**:
- *§ 3.1.3. Понятие класса действий и класса задач*

класс логически атомарных действий
:= [*class of logically atomic actions*]
⇐ *ключевой знак**:
- *§ 3.3.2. Действия, задачи, планы, протоколы и методы, реализуемые ostis-системой*

класс методов
:= [*method class*]
⇐ *ключевое понятие**:
- *§ 3.1.6. Понятие класса методов и языка представления методов*

количественный метод оценки пользовательских интерфейсов
:= [*quantitative method of user interface evaluation*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

коллектив ostis-систем
:= [*ostis-systems group*]
⇐ *ключевое понятие**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

команда
:= [*command*]

⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*

комплемент
:= [*complement*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

компонент пользовательского интерфейса
:= [*user interface component*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

компонентное проектирование
:= [*component design*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

компонентное проектирование баз знаний интеллектуальных систем
:= [*intelligent system knowledge base component design*]
⇐ *ключевое понятие**:
- *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*

компонентное проектирование интеллектуальных систем
:= [*intelligent system component design*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

компонент ostis-системы
:= [*ostis-system component*]
⇐ *ключевое понятие**:
- *Глава § 5.1.3. Понятие многократно используемого компонента ostis-систем*

компьютерная алгебра
:= [*computer algebra*]
⇐ *ключевое понятие**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

компьютерное зрение
:= [*computer vision*]
⇐ *ключевое понятие**:
- *§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции*

конструктивно истинное высказывание*
:= [*constructively true statement**]
⇐ *ключевое отношение**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

контекст
:= [*context*]
⇐ *ключевое понятие**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

контекст диалога
:= [*dialog context*]
⇐ *ключевое понятие**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

контекстно-зависимая система

:= [*context-dependent system*]
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

контекст*
:= [*context**]
⇐ *ключевое отношение**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

корпоративная система
:= [*corporate system*]
⇐ *ключевое понятие**:
- *§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения*

Корпоративная система Экосистемы OSTIS
:= [*OSTIS Ecosystem corporate system*]
⇐ *ключевой знак**:
- *Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS*

корпоративная ostis-система
:= [*corporate ostis-system*]
⇐ *ключевое понятие**:
- *§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения*

лексема
:= [*lexeme*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

линия разметки sc.n-текста
:= [*sc.n-text markup line*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

логическая формула
:= [*logical formula*]
⇐ *ключевое понятие**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

логическая связка*
:= [*logical sheaf**]
⇐ *ключевое отношение**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

логический канал связи
:= [*logical link*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

локальный нетранслируемый sc-идентификатор
:= [*local non-translatable sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента*

локальный признак изображения
:= [*local image feature*]
⇐ *ключевое понятие**:

- *§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции*

материнская ostis-система
:= [*maternal ostis-system*]
:= [*parent ostis-system*]
⇐ *ключевое понятие**:
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

машина обработки знаний
:= [*knowledge processing machine*]
⇐ *ключевое понятие**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

машина фон-Неймана
:= [*von Neumann machine*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

менеджер многократно используемых компонентов ostis-систем
:= *сокращение основного sc-идентификатора**:
[менеджер компонентов]
:= [*ostis-systems reusable component manager*]
:= [*sc-component-manager*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

метаметод
:= [*метапрограмма*]
:= [*meta-method*]
⇐ *ключевое понятие**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Метасистема OSTIS
:= [*OSTIS Metasystem*]
:= *сокращение основного sc-идентификатора**:
[Intelligent MetaSystem]
:= [интеллектуальная метасистема поддержки проектирования интеллектуальных систем]
⇐ *ключевой знак**:
- *Глава 7.2. Метасистема OSTIS*

метаструктура
:= [*meta-structure*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

метод
:= [*программа*]
:= [*method*]
⇐ *ключевое понятие**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

метод заданного языка представления методов
:= [*specified method representation language method*]
⇐ *ключевое понятие**:
- *§ 3.1.6. Понятие класса методов и языка представления методов*

метод оценки пользовательских интерфейсов
:= [*user interface evaluation method*]
⇐ *ключевое понятие**:

- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

метрика
:= [*metric*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

метрическое пространство
:= [*metric space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

метрическое конечное семантическое пространство
:= [*metric finite semantic space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

метрическое конечное синтаксическое пространство
:= [*metric finite syntactic space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

микротехнологическая операция
:= [*microtechnological operation*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

минимальная конфигурация ostis-системы
:= [*ostis-system minimal configuration*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

многоагентная модель решения задач
:= [*multiagent problem solving model*]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

многоагентная система
:= [*multiagent system*]
:= [кибернетическая система, представляющая собой коллектив взаимодействующих кибернетических систем, являющихся агентами (членами) этого коллектива]
:= [коллективная кибернетическая система]
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

многоагентная система обработки информации в общей памяти
:= [*multiagent system of information processing in common memory*]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

многоагентный интерфейс интеллектуальной компьютерной системы нового поколения
:= [*next-generation intelligent computer system multiagent interface*]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

многоагентный решатель задач
:= [*multiagent problem solver*]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

многократно используемый компонент ostis-систем
:= *сокращение основного sc-идентификатора**:

[м.и.к. ostis-систем]
:= [*ostis-systems reusable component*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

многократно используемый компонент решателей задач
:= [*reusable problem solver component*]
⇐ *ключевое понятие**:
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

множество
:= [*set*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

модальный оператор
:= [*modal operator*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

модальное правило вывода
:= [*modal inference rule*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

модель решения задач
:= [*problem solving model*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

монотонное бинарное отношение*
:= [*monotonic binary relation**]
⇐ *ключевой отношение**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

мощность множества
:= [*set power*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

мультимножество
:= [*multiset*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

мультимодальный интерфейс
:= [*multimodal interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

навык
:= [*skill*]
⇐ *ключевое понятие**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

навык решения задач с помощью искусственных нейронных сетей
:= [*skill of problem solving using artificial neural networks*]
⇐ *ключевое понятие**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

накопительный модуль
:= [*storage module*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

начало^
:= [*begin^*]
⇐ *ключевой параметр**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

неатомарное действие
:= [*non-atomic action*]
⇐ *ключевое отношение**:
- *§ 3.1.1. Понятие действия*

неатомарный абстрактный sc-агент
:= [*non-atomic abstract sc-agent*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

небинарная связь
:= [*nonbinary sheaf*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

небинарное отношение
:= [*nonbinary relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

неискаженное высказывание*
:= [*undistorted statement**]
⇐ *ключевое отношение**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

нейросетевая модель решения задач
:= [*neural network model of problem solving*]
⇐ *ключевое понятие**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

нейросетевой метод решения задач
:= [*и.н.с.*]
:= [*искусственная нейронная сеть*]
:= [*artificial neural network*]
:= [*artificial neural network method of problem solving*]
⇐ *ключевое понятие**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

некорректность в scp-программе
:= [*inaccuracy in scp-program*]
⇐ *ключевое понятие**:
- *5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

немонотонный вывод на конечном sc-множестве посылок
:= [*non-monotonic inference on a finite sc-set of premises*]
⇐ *ключевое понятие**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

неориентированное множество
:= [*non-oriented set*]

⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

неосновной sc-идентификатор
:= [*auxiliary sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.1. Понятие sc-идентификатора sc-элемента*

непроцедурный язык представления методов
:= [*non-procedural method representation language*]
⇐ *ключевое понятие**:
- *§ 3.1.7. Общая классификация языков представления методов*

неролевое отношение
:= [*norole relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

неслотовое бинарное отношение
:= [*non-slot binary relation*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

нестроковый sc-идентификатор
:= [*non-string sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

нетранслируемый sc-идентификатор
:= [*non-translatable sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента*

неточная величина
:= [*inexact value*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

нефактографическое высказывание
:= [*non-factual statement*]
⇐ *ключевое понятие**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

нечеткая истинность*
:= [*fuzzy truth**]
⇐ *ключевое отношение**:
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

область определения′
:= [*definitional domain′*]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

область прибытия′
:= [*output set′*]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

Обобщенный жизненный цикл ostis-систем
:= [*Generalized ostis-systems life cycle*]
⇐ *ключевой знак**:

- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

обозначение внешней сущности
:= [*external entity designation*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

обозначение sc-класса
:= [*sc-class designation*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

обозначение sc-множества
:= [*sc-set designation*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

обозначение sc-связки
:= [*sc-sheaf designation*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

обозначение sc-структуры
:= [*sc-structure designation*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

объединение*
:= [*union**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

объект′
:= [*object′*]
⇐ *ключевое отношение**:
- *§ 3.1.1. Понятие действия*

объект местности
:= [*terrain object*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

онтология
:= [*ontology*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *§ 2.5.5. Формализация понятия онтологии*

онтология верхнего уровня
:= [*top-level ontology*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *§ 2.5.6. Онтологии верхнего уровня*

операционная семантика метода*

:= [*operational semantics of method**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.1.5. Спецификация методов и понятие навыка*

операционная семантика языка представления методов*
:= [*operational semantics of method representation language**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

оптическая система компьютерного зрения
:= [*optical computer vision system*]
⇐ *ключевое понятие**:
- *§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции*

ориентированное множество
:= [*oriented set*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

основной sc-идентификатор
:= [*main sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

ответ на вопрос
:= [*question answer*]
⇐ *ключевое понятие**:
- *Глава 3.4. Язык вопросов для ostis-систем*

отношение
:= [*relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

отношение порядка
:= [*order relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

отслеживание движения глаз
:= [*eye tracking*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

Отношение выводимости
:= [*Inferability relation*]
⇐ *ключевой знак**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

Отношение выводимости на конечных множествах
:= [*Inferability relation on finite sets*]
⇐ *ключевой знак**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

Отношение выводимости на конечных множествах полносвязно представленных множеств
:= [*Inferability relation on finite sets of fully connected sets*]
⇐ *ключевой знак**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

Отношение выводимости на секвенциях
:= [*Inferability relation on sequences*]

⇐ *ключевой знак**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

отношение, заданное на множестве знаний
:= [*relation defined on a set of knowledge*]
⇐ *ключевое понятие**:
- *§ 2.5.1. Формализация понятия знания и формальные модели баз знаний ostis-систем*

отношение становления структур
:= [*relation of structure formation*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

ошибка в scp-программе
:= [*error in scp-program*]
⇐ *ключевое понятие**:
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

пакетный менеджер
:= [*package manager*]
⇐ *ключевое понятие**:
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

параметр
:= [*parameter*]
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

параметр, заданный на множестве кибернетических систем
:= [*parameter defined on cybernetic systems set*]
∋ *качество физической оболочки кибернетической системы^*
∋ *качество хранимой информации^*
:= [качество информации, хранимой в памяти кибернетической системы^]
∋ *качество решателя задач^*
∋ *качество интерфейса^*
∋ *обучаемость^*
∋ *гибкость кибернетической системы^*
∋ *стратифицированность кибернетической системы^*
∋ *рефлексивность кибернетической системы^*
∋ *уровень эволюционных ограничений^*
⇐ *ключевое понятие**:
- *Глава 1.1. Факторы, определяющие уровень интеллекта кибернетических систем*

параметр scp-программы'
:= [*scp-program parameter'*]
⇐ *ключевое отношение**:
- *§ 3.3.5. Базовый язык программирования ostis-систем*

пассивный навык
:= [*passive skill*]
⇐ *ключевое понятие**:
- *§ 3.1.5. Спецификация методов и понятие навыка*

пересечение*
:= [*intersection**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

персональный ассистент
:= [*personal assistant*]
⇐ *ключевое понятие**:
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

персональный ostis-ассистент
:= [*personal ostis-assistant*]
⇐ *ключевое понятие**:
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

планируемая блокировка*
:= [*planned lock**]
⇐ *ключевое отношение**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

поведенческая задача
:= [*behavioral problem*]
⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*

поведенческое действие
:= [*behavioral action*]
⇐ *ключевое понятие**:
- *§ 3.1.1. Понятие действия*

подметод*
:= [*submethod**]
⇐ *ключевое отношение**:
- *§ 3.1.4. Понятие метода*

подформула*
:= [*subformula**]
⇐ *ключевое отношение**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

полная семантическая окрестность элемента*
:= [*full semantic neighborhood of element**]
⇐ *ключевое отношение**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

пользовательский интерфейс
:= [*user interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

пользователь Экосистемы OSTIS
:= [*OSTIS Ecosystem user*]
⇐ *ключевое понятие**:
- *Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS*

понятие, переходящее из основного в неосновное
:= [*concept that transitions from main to auxiliary*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

понятие, переходящее из неосновного в основное
:= [*concept that transitions from auxiliary to main*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

последовательность мышления
:= [*thinking sequence*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

портал знаний
:= [*knowledge portal*]

⇐ *ключевое понятие**:
- *§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний*

Правила построения нетранслируемых sc-идентификаторов
:= [*Non-translatable sc-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.3. Понятие нетранслируемого sc-идентификатора sc-элемента*

Правила построения основных sc-идентификаторов
:= [*Main sc-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

Правила построения простых sc-идентификаторов
:= [*Simple sc-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента*

Правила построения системных sc-идентификаторов
:= [*System sc-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

Правила построения сложных sc-идентификаторов
:= [*Complex sc-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.5. Понятие сложного sc-идентификатора sc-элемента*

Правила построения sc-идентификаторов
:= [*SC-identifier construction rules*]
⇐ *ключевое знание**:
- *Пункт 2.3.1.1. Понятие sc-идентификатора sc-элемента*

предметная область
:= [*subject domain*]
⇐ *ключевое понятие**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *§ 2.5.4. Формализация понятия предметной области*

приложение умного дома
:= [*smart home application*]
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

принадлежность*
:= [*belonging**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

Принципы, лежащие в основе Технологии OSTIS
:= [*OSTIS Technology principles*]
⇐ *ключевое знание**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

приоритет блокировки*
:= [*lock priority**]
⇐ *ключевое отношение**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

Программный вариант реализации ostis-платформы
:= [*Программная платформа для ostis-систем*]
:= [*Software version of ostis-platform implementation*]
:= [*Software platform for ostis-systems*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

программный вариант ostis-платформы
:= [*software version of ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

программный интерфейс
:= [*application interface*]
:= [*API*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

Программный интерфейс Реализации sc-памяти в ostis-платформе
:= [*Application interface of SC-memory implementation in ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

проектирование
:= [*design*]
⇐ *ключевое понятие**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

производство
:= [*production*]
⇐ *ключевое понятие**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

произведение*
:= [*multiplication**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

простой sc-идентификатор
:= [*simple sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.4. Понятие простого sc-идентификатора sc-элемента*

пространственное отношение
:= [*space relation*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

процедурная формулировка задачи
:= [*procedural problem definition*]
⇐ *ключевое понятие**:
- *§ 3.1.2. Понятие задачи*

процедурная формулировка задачи*
:= [*procedural problem definition**]
⇐ *ключевое отношение**:
- *§ 3.1.2. Понятие задачи*

процедурный язык представления методов
:= [*procedural method representation language*]
⇐ *ключевое понятие**:

- *§ 3.1.7. Общая классификация языков представления методов*

процесс
:= [*process*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

процессорный модуль
:= [*processor module*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

процессорный элемент
:= [*processor element*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

псевдометрика
:= [*pseudometric*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

псевдометрическое пространство
:= [*pseudometric space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

псевдометрическое конечное семантическое пространство
:= [*pseudometric finite semantic space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

разбиение*
:= [*partition**]
:= [*subdividing**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

разработка плана производства
:= [*production plan development*]
⇐ *ключевое понятие**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

распределенная система
:= [*distributed system*]
⇐ *ключевое понятие**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

расширенная ostis-платформа
:= [*extended ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

Реализация интерпретатора sc-моделей пользовательских интерфейсов
:= [*Implementation of user interfaces sc-models*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

Реализация менеджера многократно используемых компонентов ostis-систем в ostis-платформе
:= [*Implementation of ostis-systems reusable component manager in ostis-platform*]
⇐ *ключевое понятие**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 6.3. Программная платформа ostis-систем*

Реализация памяти в ostis-платформе
:= [*Memory implementation in ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

Реализация подсистемы взаимодействия ostis-платформы с внешней средой
:= [*Implementation of subsystem of interaction between ostis-platform and external environment*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

Реализация sc-памяти в ostis-платформе
:= [*SC-memory implementation in ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

Реализация файловой памяти в ostis-платформе
:= [*File memory implementation in ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.3. Программная платформа ostis-систем*

реинжиниринг
:= [*reengineering*]
⇐ *ключевое понятие**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

рецепторное действие
:= [*receptor action*]
⇐ *ключевое понятие**:
- *§ 3.1.1. Понятие действия*

рецептурное производство
:= [*product formulation*]
⇐ *ключевое понятие**:
- *§ 7.7.2. Построение умных предприятий рецептурного производства с помощью ostis-систем*

речевой интерфейс
:= [*speech interface*]
⇐ *ключевое понятие**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*

речевой сигнал
:= [*speech signal*]
⇐ *ключевое понятие**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*

решатель задач пользовательского интерфейса ostis-систем
:= [*ostis-system user interface problem solver*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

решатель задач ostis-системы
:= [*ostis-system problem solver*]
⇐ *ключевое понятие**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

решатель задач

:= *сокращение основного sc-идентификатора**:
[р.з.]
:= [*problem solver*]
⇐ *ключевое понятие**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

ролевое отношение
:= [*role relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

самостоятельная ostis-система
:= [*independent ostis-system*]
⇐ *ключевое понятие**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

связь
:= [*sheaf*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

секвенция
:= [*sequence*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

семантическая метрика
:= [*semantic metric*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

семантическая окрестность
:= [*semantic neighborhood*]
⇐ *ключевое понятие**:
- *§ 2.5.3. Формализация понятия семантической окрестности*

семантическая сеть
:= [*semantic network*]
⊃ *рафинированная семантическая сеть*
⊃ *иерархическая семантическая сеть*
:= [семантическая сеть, являющаяся метаграфовой]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

Семантическая теория программ для ostis-систем
:= [*Semantic program theory for ostis-systems*]
⇐ *ключевой знак**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

семантический электронный учебник
:= [*semantic electronic handbook*]
⇐ *ключевое понятие**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*

семейство множеств
:= [*set of sets*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

сервис
:= [*service*]
⇐ *ключевое понятие**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

сигнал
:= [*signal*]
⇐ *ключевое понятие**:
- *4.3. Аудиоинтерфейс ostis-систем*

Синтаксис SCg-кода
:= [*SCg-code syntax*]
⇐ *ключевой знак**:
- *Пункт 2.3.2.1. Синтаксис SCg-кода*

Синтаксис SCn-кода
:= [*SCn-code syntax*]
⇐ *ключевой знак**:
- *Пункт 2.3.4.1. Синтаксис SCn-кода*

Синтаксис SCs-кода
:= [*SCs-code syntax*]
⇐ *ключевой знак**:
- *Пункт 2.3.3.1. Синтаксис SCs-кода*

синтаксис языка
:= [*language syntax*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

синтаксис языка представления методов*
:= [*syntax of method representation language**]
⇐ *ключевое отношение**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *Глава 3.2. Семантическая теория программ для ostis-систем*

синтаксическая группа
:= [*phrase*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

система компьютерной алгебры
:= [*с.к.а.*]
:= [*computer algebra system*]
⇐ *ключевое понятие**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

система компьютерной математики
:= [*с.к.м.*]
:= [*computer mathematics system*]
⇐ *ключевое понятие**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

система локального позиционирования
:= [*real-time locating system*]
⇐ *ключевое понятие**:
- *§ 4.4.7. Системы локального позиционирования, использующиеся в задачах трехмерной реконструкции*

системный sc-идентификатор
:= [*system sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

ситуация

:= [*situation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

словоформа
:= [*word form*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

слотовое бинарное отношение
:= [*slot binary relation*]
⇐ *ключевое понятие**:
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

смысловое представление информации
:= [*semantic representation of information*]
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *Глава 2.1. Информационные конструкции и языки*

событие в sc-памяти
:= [*event in sc-memory*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

сообщение
:= [*message*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

соответствие*
:= [*mapping**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

составляющая
:= [*constituent*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

специализированная ostis-платформа
:= [*specialized ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

спецификатор
:= [*specifier*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

спецификация*
:= [*specification**]
⇐ *ключевое отношение**:
- *§ 3.1.2. Понятие задачи*

спецификация метода*
:= [*method specification**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

спецификация языка представления методов*
:= [*method representation language specification**]
⇐ *ключевое отношение**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Стандарт Технологии OSTIS
:= [*Стандарт ostis-систем*]
:= [*Стандарт OSTIS*]
:= [*OSTIS Technology Standard*]
:= [*Standard of ostis-systems*]
⇐ *ключевой знак**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS*

страница sc.n-текста
:= [*sc.n-text page*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

стратегия решения задач
:= [*problem solving strategy*]
⇐ *ключевое понятие**:
- *§ 3.1.4. Понятие метода*

стратегия решения информационных задач
:= [*information problem solving strategy*]
⇐ *ключевое понятие**:
- *§ 3.1.4. Понятие метода*

строка sc.n-текста
:= [*sc.n-text string*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

строковый sc-идентификатор
:= [*string sc-identifier*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.2. Понятие основного и системного sc-идентификаторов sc-элемента*

структура
:= [*structure*]
⇐ *ключевое понятие**:
- *§ 2.5.2. Формализация понятия структуры*

субъект
:= [*subject*]
⇐ *ключевое понятие**:
- *§ 3.1.1. Понятие действия*

субъект'
:= [*subject'*]
⇐ *ключевое отношение**:
- *§ 3.1.1. Понятие действия*

сумма*
:= [*sum**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

сцена в трехмерном представлении
:= [*scene in 3D representation*]
⇐ *ключевое понятие**:
- *§ 4.4.4. Семантическое представление объектов и сцены*

темпоральное отношение
:= [*temporal relation*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

терминальный модуль
:= [*terminal module*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

тестирование удобства использования пользовательских интерфейсов
:= [*usability testing of user interfaces*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

технология
:= [*technology*]
:= [комплекс моделей, методик, методов и средств, обеспечивающих выполнение соответствующего вида деятельности]
⊃ *технология поддержки жизненного цикла*
- ⊃ *технология комплексной поддержки жизненного цикла*
  - ⊃ *технология комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
    - ∋ *Технология OSTIS*
      - := [Технология комплексной поддержки жизненного цикла ostis-систем]

⇐ *ключевое понятие**:
- *Часть 1. Введение в интеллектуальные компьютерные системы нового поколения и технологию комплексной поддержки их жизненного цикла*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

технология проектирования интеллектуальных систем
:= [*intelligent systems design technology*]
⇐ *ключевое понятие**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Технология OSTIS
:= [*OSTIS Technology*]
⇐ *ключевой знак**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

технология*
:= [*technology**]
⇐ *ключевое отношение**:
- *§ 3.1.9. Понятие деятельности, вида деятельности и технологии*

технологическая операция
:= [*technological operation*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

технологический процесс
:= [*technological process*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

тип блокировки
:= [*lock type*]
⇐ *ключевое понятие**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

топологическое пространство
:= [*topological space*]
⇐ *ключевое понятие**:
- *§ 2.2.3. Смысловое пространство ostis-систем*

точка останова
:= [*breakpoint*]
⇐ *ключевое понятие**:
- *5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

точная величина
:= [*exact value*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

точность*
:= [*accuracy**]
⇐ *ключевое отношение**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

транзакция в sc-памяти
:= [*transaction in sc-memory*]
⇐ *ключевое понятие**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

трехмерная модель объекта
:= [*object 3D model*]
⇐ *ключевое понятие**:
- *§ 4.4.5. Трехмерное представление объектов в сцене*

трехмерная реконструкция
:= [*3D reconstruction*]
⇐ *ключевое понятие**:
- *§ 4.4.6. Трехмерная реконструкция объектов окружающего мира*

трехмерное представление объекта
:= [*object 3D representation*]
⇐ *ключевое понятие**:
- *§ 4.4.5. Трехмерное представление объектов в сцене*

удаляемые sc-элементы*
:= [*sc-elements to be deleted**]
⇐ *ключевое отношение**:
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

умная больница
:= [*smart-больница*]
:= [*smart-hospital*]
⇐ *ключевое понятие**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

Умное общество
:= [*Smart-общество*]
:= [*Общество 5.0*]
:= [*Интеллектуальное общество*]
:= [*Smart-society*]

⇐ *ключевой знак**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

умное предприятие
:= [*smart-предприятие*]
:= [*предприятие 5.0*]
:= [*smart-enterprise*]
⇐ *ключевое понятие**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*

умный город
:= [*smart-город*]
:= [*smart-city*]
⇐ *ключевое понятие**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

умный дом
:= [*smart-дом*]
:= [*smart-home*]
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

УСК
:= *сокращение основного sc-идентификатора**:
[*Универсальный семантический код*]
:= [*Универсальный семантический код, разработанный В. В. Мартыновым*]
:= [*Universal semantic code*]
:= [*USC*]
⇐ *ключевой знак**:
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

файл
:= [*file*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

файл ostis-системы
:= [*ostis-system file*]
:= [*file of ostis-system*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

фактографическое высказывание
:= [*factual statement*]
⇐ *ключевое понятие**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*

физический интерфейс
:= [*physical interface*]
⇐ *ключевое понятие**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

физический канал связи
:= [*physical link*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

цель*
:= [*goal**]
⇐ *ключевое отношение**:
- *§ 3.1.1. Понятие действия*

цифра
:= [*digit*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

цифровая экосистема
:= [*digital ecosystem*]
⇐ *ключевое понятие**:
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

цифровой двойник
:= [*digital twin*]
⇐ *ключевое понятие**:
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

часть речи
:= [*part of speech*]
⇐ *ключевое понятие**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

число
:= [*number*]
⇐ *ключевое понятие**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

эквивалентность задач*
:= [*problem equivalence**]
⇐ *ключевое отношение**:
- *§ 3.1.4. Понятие метода*

Экосистема OSTIS
:= [*OSTIS Ecosystem*]
⇐ *ключевой знак**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *Глава 7.3. Структура Экосистемы OSTIS*
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

экспертная оценка пользовательских интерфейсов
:= [*expert evaluation of user interfaces*]
⇐ *ключевое понятие**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*

эффекторное действие
:= [*effector action*]
⇐ *ключевое понятие**:
- *§ 3.1.1. Понятие действия*

эффективность метода
:= [*method efficiency*]
⇐ *ключевое понятие**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Ядро базы знаний ostis-системы
:= [*Knowledge base core of ostis-system*]
⇐ *ключевой знак**:

- *Глава 3.4. Язык вопросов для ostis-систем*

язык
:= [*language*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Язык вопросов для ostis-систем
:= [*Questions language for ostis-systems*]
⇐ *ключевой знак**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Язык карт для ostis-систем
:= [*Map language for ostis-systems*]
⇐ *ключевое понятие**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

язык представления методов
:= [*я.п.м.*]
:= [*язык программирования*]
:= [*method representation language*]
⇐ *ключевое понятие**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

Язык представления нейросетевых методов решения задач в базах знаний
:= [*Neural network methods problem solving representation language*]
⇐ *ключевой знак**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

язык ostis-системы
:= [*ostis-system language*]
⇐ *ключевое понятие**:
- *Глава 2.1. Информационные конструкции и языки*

Язык SCL
:= [*SCL language*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code Logic]
⇐ *ключевой знак**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*

Язык SCP
:= [*SCP language*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code Programming]
:= [Графовый процедурный язык программирования, построенный на базе *SC-кода*]
⇐ *ключевой знак**:
- *§ 3.3.5. Базовый язык программирования ostis-систем*

Язык SCPD
:= [*SCPD language*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code Distributed]
⇐ *ключевой знак**:
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

ambient assisted living
⇐ *ключевое понятие**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

ISA-88
:= [*ISA-88*]
⇐ *ключевой знак**:
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

ISA-95
:= [*ISA-95*]
⇐ *ключевой знак**:
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

Maple
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Mathematica
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

MATLAB
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Maxima
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

MQTT
⇐ *ключевой знак**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

Node-RED
⇐ *ключевой знак**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

OSTIS
:= *сокращение основного sc-идентификатора**:
[Open Semantic Technology for Intelligent Systems]
:= *сокращение основного sc-идентификатора**:
[Открытая семантическая технология проектирования интеллектуальных систем]

ostis-
:= *сокращение основного sc-идентификатора**:
[for open semantic technology for intelligent systems]
⇒ *пример применения**:
[
- ostis-система
- ostis-платформа
- ostis-сообщество

]

ostis-ассистент
:= [*ostis-assistant*]
⇐ *ключевое понятие**:

- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

ostis-платформа
:= [*ostis-platform*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

ostis-портал знаний
:= [*ostis-portal of knowledge*]
⇐ *ключевое понятие**:
- *§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний*

ostis-система
:= [*ostis-system*]
⇐ *ключевое понятие**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

ostis-сообщество
:= [*ostis-community*]
⇐ *ключевое понятие**:
- *Пункт 7.3.1.2. Описание структуры Экосистемы OSTIS*

ostis-технология
:= [*частная технология, входящая в состав комплексной Технологии OSTIS*]
:= [*ostis-technology*]
⇐ *ключевой знак**:
- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*

OWL
:= *сокращение основного sc-идентификатора**:
[*Web Ontology Language*]
:= [*Язык описания онтологий для семантической паутины*]
⇐ *ключевой знак**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

RDF
:= *сокращение основного sc-идентификатора**:
[*Resource Description Framework*]
:= [*Разработанная Консорциумом Всемирной паутины модель для представления данных*]
⇐ *ключевой знак**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

REST API
⇐ *ключевой знак**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

SC-
:= *сокращение основного sc-идентификатора**:
[Semantic Code]
:= *сокращение основного sc-идентификатора**:
[Semantic Computer]
⇒ *пример применения**:
[
- SC-код
- sc-элемент

- sc-множество
- sc-модель

]

sc-агент
:= [*sc-agent*]
⇐ *ключевое понятие**:
- *§ 3.3.3. Внутренние агенты, выполняющие действия в sc-памяти – sc-агенты*

sc-выражение
:= [сложный sc-идентификатор]
:= [*complex sc-identifier*]
:= [*sc-expression*]
⇐ *ключевое понятие**:
- *Пункт 2.3.1.5. Понятие сложного sc-идентификатора sc-элемента*

sc-идентификатор
:= [*sc-identifier*]
⇒ *часто используемый sc-идентификатор**:
[внешний идентификатор sc-элемента]
⇐ *ключевое понятие**:
- *§ 2.3.1. Внешние идентификаторы sc-элементов — знаков, входящих в конструкции SC-кода*

sc-класс
:= [*sc-class*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

SC-код
:= [*SC-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code]
:= [Универсальный базовый способ смыслового представления знаний в виде семантических сетей с базовой теоретико-множественной интерпретацией]
⇐ *часто используемый sc-идентификатор**:
sc-структура
∈ *часто используемый sc-идентификатор*
∈ *имя собственное*
:= [sc-конструкция]
⇐ *ключевой знак**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

⇐ *ключевой термин**:
- *Параграф § 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*

sc-машина
:= [*sc-machine*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

sc-множество
:= [*sc-set*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

sc-память
:= [*sc-memory*]
⇐ *ключевое понятие**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*

sc-связка
:= [*sc-sheaf*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

sc-структура
:= [*sc-structure*]
⇐ *ключевое понятие**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

sc-элемент
:= [*sc-element*]
⇐ *часто используемый sc-идентификатор**:
сущность
⇐ *ключевое понятие**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

sc-язык
:= [*sc-language*]
:= [подъязык *SC-кода*]
⇐ *ключевое понятие**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

SCD-код
:= [*SCD-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code Distributed]
⇐ *ключевой знак**:
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

SCg-код
:= [*SCg-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code graphic]
:= [Графический нелинейный вариант визуализации текстов SC-кода]
⇐ *ключевой знак**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

SCfin-код
:= [*SCfin-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code file interior]
⇐ *ключевой знак**:
- *Глава 6.3. Программная платформа ostis-систем*

SCin-код
:= [*SCin-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code interior]
⇐ *ключевой знак**:
- *Глава 6.3. Программная платформа ostis-систем*

SC-JSON-код

⇐ *ключевой знак**:
- *Глава 6.3. Программная платформа ostis-систем*

sc.g-дуга
:= [*sc.g-arc*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

sc.g-коннектор
:= [*sc.g-connector*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

sc.g-рамка
:= [*sc.g-frame*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

sc.g-ребро
:= [*sc.g-edge*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

sc.g-элемент
:= [*sc.g-element*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

sc.g-шина
:= [*sc.g-bus*]
⇐ *ключевое понятие**:
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*

SCn-код
:= [*SCn-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code natural]
:= [Гипертекстовый вариант визуализации текстов *SC-кода*]
⇐ *ключевой знак**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-дуга
:= [*sc.n-arc*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-коннектор
:= [*sc.n-connector*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-контур
:= [*sc.n-contour*]

⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-рамка
:= [*sc.n-frame*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-ребро
:= [*sc.n-edge*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

sc.n-элемент
:= [*sc.n-element*]
⇐ *ключевое понятие**:
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*

scp-компьютер
:= [*scp-computer*]
⇐ *ключевое понятие**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

scp-операнд'
:= [*scp-operand'*]
⇐ *ключевое отношение**:
- *§ 3.3.5. Базовый язык программирования ostis-систем*

scp-оператор
:= [*scp-operator*]
⇐ *ключевое понятие**:
- *§ 3.3.5. Базовый язык программирования ostis-систем*

SCs-код
:= [*SCs-code*]
:= *сокращение основного sc-идентификатора**:
[Semantic Code string]
:= [Линейный вариант визуализации текстов SC-кода]
⇐ *ключевой знак**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-дуга
:= [*sc.s-arc*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-коннектор
:= [*sc.s-connector*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-модификатор
:= [*sc.s-modifier*]
⇐ *ключевое понятие**:

- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-ограничитель
:= [*sc.s-limiter*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-предложение
:= [*sc.s-sentence*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-разделитель
:= [*sc.s-separator*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-ребро
:= [*sc.s-edge*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

sc.s-элемент
:= [*sc.s-element*]
⇐ *ключевое понятие**:
- *§ 2.3.3. Язык внешнего линейного представления конструкций SC-кода — SCs-код (Semantic Code string)*

Yandex Cloud
⇐ *ключевой знак**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

Yandex IoT Core
⇐ *ключевой знак**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

WebSocket
⇐ *ключевой знак**:
- *Глава 6.3. Программная платформа ostis-систем*

Wolfram
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Wolfram Alpha
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Wolfram Mathematica
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Wolfram Language
⇐ *ключевой знак**:
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Библиография OSTIS

В *Библиографии OSTIS* приводится список библиографических источников в алфавитном порядке их идентификаторов, при этом вначале перечисляются русскоязычные источники, а затем — англоязычные.

Каждый библиографический источник имеет соответствующий уникальный идентификатор, а также формальную спецификацию.

Идентификаторы статей, книг и других печатных работ строятся следующим образом:
- Пишется фамилия первого автора на том языке, на котором опубликована данная работа. Затем через пробел ставятся инициал(-ы) первого автора
- Если работа опубликована с участием только одного автора, то ставится одна точка, если нескольких — две точки. Если работа представляет собой коллективную публикацию под редакцией кого-либо, то вместо фамилии автора указывается фамилия и инициалы главного редактора и вместо двух точек ставится сочетание символов ".ред." или ".ed." в зависимости от языка.
- Пишется первых пять букв первого слова из названия работы на том языке, на котором опубликована данная работа. Если первое слово названия работы служебное (предлог, частица, артикль и так далее), то ставится первая строчная буква этого слова, а первых пять букв берется из следующего (первого значимого) слова.
- Перечисляются заглавные (прописные) первые буквы всех остальных слов названия работы за исключением служебных слов, таких как предлоги, частицы, артикли и тому подобное. Для всех служебных слов указываются строчные первые буквы. Если название содержит очень много слов, то допускается использовать только первые 5-7 слов. Если служебное слово идет сразу же после первого значимого слова, то перед соответствующей строчной буквой ставится пробел.
- Ставится дефис
- Указывается год издания работы
- Указывается 2-3 буквенный код, обозначающий тип работы, на том языке, на котором она опубликована, например:
  - *кн* или *bk* — книга
  - *ст* или *art* — статья

Идентификаторы электронных и прочих ресурсов формируются аналогичным образом, с учетом того, что опускается год издания и фамилии авторов, а также ставится буквенный код *эл* для обозначения электронного ресурса. Например, *МетасOSTIS-2022эл*, *Cypher-2022эл*.

При спецификации конкретного библиографического источника в разделе библиографии указываются:
- Идентификатор библиографического источника, составленный в соответствии с указанными выше правилами.
- Его библиографическое описание (библиографический идентификатор), составленное в соответствии с каким-либо общепринятым стандартом (ГОСТ, IEEE, ACM и так далее).
- Указывается аннотация данного библиографического источника.
- Перечисляются ключевые знаки (понятия или конкретные сущности) для описываемого библиографического источника. При этом подразумевается, что и имя (термин), и трактовка данного ключевого знака в описываемом источнике и в рамках Стандарта OSTIS совпадают. В противном случае имя такой ключевой сущности описывается как ключевой термин.
- Перечисляются ключевые термины для описываемого библиографического источника. При этом для каждого термина желательно при возможности указывать соответствующие понятие или конкретную сущность (ключевой знак), описываемые в рамках Стандарта OSTIS, для которых данный термин может рассматриваться как синонимичный идентификатор. Примеры использования ключевых терминов и ключевых знаков можно найти в спецификации источников *Поспелов Д.А.СитуаУТиП-1986кн* и *Голенков В.В..оОбучеИСкОС-2018ст*.
- Указываются разделы Стандарта OSTIS (Монографии OSTIS), для которых данный библиографический источник является важным.
- Приводятся цитаты из данного библиографического источника. При необходимости указывается часть источника, из которой непосредственно взята цитата. Примеры указания цитат можно найти в спецификации источников *Баринов И.И..ФормиСРКпИИ-2021ст* и *Поспелов Д.А.СитуаУТиП-1986кн*.
- Указывается любая другая информация об описываемом библиографическом источнике.

Для ссылки на библиографический источник в естественно-языковом или формальном тексте используется его идентификатор, составленный по рассмотренным выше правилам. Например:
- "Работа *Голенков В.В..оОбучеИСкОС-2018ст* посвящена вопросам обучения и обучаемости в интеллектуальных системах."

- “Основные принципы многоагентных систем рассматриваются в ряде работ (*Wooldridge M.aIntro tMS-2009bk*, *Тарасов В.Б.оМногоСкИОФ-2002кн*).”

Абдурахман Д.Д.ИскусИиМОвК-2022ст

:= *стандартное библиографическое описание**:
[Д. Д. Абдурахман, “Искусственный интеллект и машинное обучение в кибербезопасности,” *Современные проблемы лингвистики и методики преподавания русского языка в вузе и школе*, № 34, с. 916—921, 2022]

⇐ *библиографическая ссылка**:
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*

Абламейко С.В..ГеогрИССЦК-2000ст

:= *стандартное библиографическое описание**:
[С. В. Абламейко, Г. Апарин и А. Крючков, “Географические информационные системы. Создание цифровых карт,” 2000]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*

Абрамский М.М..СравнАИРиГБ-2018ст

:= *стандартное библиографическое описание**:
[М. М. Абрамский и Т. И. Тимерханов, “Сравнительный анализ использования реляционных и графовых баз данных в разработке цифровых образовательных систем,” в *Вестник НГУ*, 2018, с. 5—11]

⇒ *аннотация**:
[Рассмотрены вопросы выбора варианта реализации базы данных при разработке цифровых образовательных систем. Проведен сравнительный анализ реляционного и графового подходов к хранению базовых сущностей. Продемонстрирован ряд преимуществ графовой модели в контексте удобства разработки и эффективности реализации запросов.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Аверьянов Л.Я.ПочемЛЗВ-1993ст

:= *стандартное библиографическое описание**:
[Л. Я. Аверьянов, “Почему люди задают вопросы,” *М.: Социолог*, с. 152, 1993]

⇒ *аннотация**:
[Работа посвящена логико-философским проблемам вопроса. Дается краткий экскурс в историю изучения вопроса, современное его исследование, предлагается концептуально-гипотетическая модель вопроса и вопросно-ответных отношений. Книга рассчитана на научных работников, преподавателей философских и социологических факультетов, специалистов в области логики вопроса и ответа.]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Айзерман М.А..ДинамПкАСОГОГ-1977ст

:= *стандартное библиографическое описание**:
[М. Айзерман и др., “Динамический подход к анализу структур, описываемых графами (основы графодинамики),” *Автоматика и телемеханика*, № 7/8, с. 135–151/123—136, 1977]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Айлиф Дж.ПринцПБМ-1973кн

:= *стандартное библиографическое описание**:
[Д. Айлиф, *Принципы построения базовой машины*. Мир, 1973, с. 119]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Аладьев В.З..Введе вСПМ22-1999кн

:= *стандартное библиографическое описание**:
[В. З. Аладьев и М. Л. Шишаков, *Введение в среду пакета Mathematica 2.2*. 1999, с. 368]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*

Аладьев В.З..МодулПМvMaVV-2011кн

:= *стандартное библиографическое описание**:
[В. З. Аладьев и В. А. Ваганов, *Модульное программирование: Maple vs Mathematica, and vice versa.* 2011, с. 417]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*

Аладьев В.З..СистеКАМИП-2006кн

:= *стандартное библиографическое описание**:
[В. З. Аладьев, *Системы компьютерной алгебры: Maple: Искусство программирования*. Лаб. Базовых Знаний, 2006]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Альтшуллер Г.С..НайтиИВвТРИ-2010кн

:= *стандартное библиографическое описание**:
[Г. С. Альтшуллер, *Найти идею: Введение в ТРИЗ – теорию решения изобретательских задач, 3-е изд.* М.: Альпина Паблишер, 2010]

⇒ *аннотация**:
[Книга посвящена Теории решения изобретательских задач – ТРИЗ.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Амосов Н.М..Автом иРП-1973кн

:= *стандартное библиографическое описание**:
[Н. Амосов и др., *Автоматы и разумное поведение*. Наукова думка, 1973]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Анцыферов С.С.ОценкКИС-2013art

:= *стандартное библиографическое описание**:
[С. С. Анцыферов, “Оценка уровня качества интеллектуальных систем,” *Искусственный интеллект*, с. 316—323, 2013]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

Аршинский Л.В..КомплВПБЗсИ-2020ст

:= *стандартное библиографическое описание**:
[Л. В. Аршинский, А. А. Ермаков и М. С. Нитежук, “Комплексная верификация продукционных баз знаний с использованием VTF-логик,” *Ontology of Designing*, т. 10, № 1, с. 112—120, апр. 2020. DOI: 10.18287/2223-9537-2020-10-1-112-120. url: https://doi.org/10.18287/2223-9537-2020-10-1-112-120]

⇐ *библиографическая ссылка**:
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*

- *§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы*

Атаева О.М..СредаИПДГ-2011ст

:= *стандартное библиографическое описание**:
[О. М. Атаева и др., "Среда интеграции пространственных данных «ГеоМета»," в *Интернет и современное общество (IMS-2011) : труды XIV Всероссийской объединенной конференции, Санкт-Петербург, Россия, 2011.*, Санкт-Петербург, 2011, с. 11—16]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Баклавски К..ОнтолС2020-2020ст

:= *стандартное библиографическое описание**:
[К. Баклавски и др., "Онтологический Саммит 2020. Коммюнике: Графы знаний," *Онтология проектирования*, т. 10, № 4 (38), с. 540—555, 2020, Перевод с англ. Д. Боргест. DOI: 10.18287/2223-9537-2020-10-4-540-555.]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Баранович А.Е.СеманАИБКЗ-2011ст

:= *стандартное библиографическое описание**:
[А. Баранович, "Семантические аспекты информационной безопасности: концентрация знаний," *История и архивы*, № 13(75), с. 38—58, 2011]

⇒ *аннотация**:
[С позиции информационно-эволюционного подхода продолжается исследование основных направлений информационной безопасности интеллектуальных систем. Основное внимание в настоящей работе сконцентрировано на направлении их защиты «от информации». Статья продолжает цикл работ посвященных семантико-прагматическим аспектам обеспечения информационной безопасности.]

⇐ *библиографическая ссылка**:
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

Баринов И.И..ФормиСРКпИИ-2021ст

:= *стандартное библиографическое описание**:
[И. Баринов и др., "Формирование стратегии развития Комитета по искусственному интеллекту в Научно-образовательном центре «Инженерия будущего»," *Онтология проектирования*, т. 11, № 3, с. 260—293, сент. 2021. DOI: 10.18287/2223-9537-2021-11-3-260-293. url: https://doi.org/10.18287/2223-9537-2021-11-3-260-293]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Библиография OSTIS*
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*

Баталов Р.Н..ГеопрИДЗвИК-2021ст

:= *стандартное библиографическое описание**:
[Р. Н. Баталов и Л. К. Радченко, "Геопространственные информация, данные, знания в историко-картографических исследованиях," *Регулирование земельно-имущественных отношений в России: правовое и геопространственное обеспечение, оценка недвижимости, экология, технологические решения*, № 1, с. 112—118, 2021]

⇒ *аннотация**:
[Понятия «геопространственная информация»,«геопространственные данные» и «геопространственные знания» имеют много общего, но при этом они заметно различаются при конкретном применении в описании того или иного процесса. Возникает необходимость выявить смысловое содержание этих понятий и определить значение и применение геопространственных информации, данных и знаний в историко-картографических исследованиях, что и является целью данной статьи.]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Батыршин И.З.ОсновОНЛииО-2001кн

:= *стандартное библиографическое описание**:

[И. З. Батыршин, *Основные операции нечеткой логики и их обобщения*. Казань: Отечество, 2001]

⇒ *аннотация**:

[В книге рассматриваются свойства операций конъюнкции, дизъюнкции и отрицания нечеткой логики и определяемых ими операций пересечения, объединения и дополнения нечетких множеств]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Батыршин И.З..НечетГСТиП-2007кн

:= *стандартное библиографическое описание**:

[И. З. Батыршин и др., *Нечеткие гибридные системы. Теория и практика*, под ред. Н. Г. Ярушиной, ред. М.: Физматлит, 2007]

⇒ *аннотация**:

[Книга посвящена современным подходам к математическому и компьютерному моделированию нечеткости. Рассмотрены операции нечеткой логики и их обобщения. Широко представлен контекст современной теории нечетких систем: синергетический искусственный интеллект, нечеткие нейронные и нечеткие эволюционные гибриды, возможностное программирование]

⇐ *библиографическая ссылка**:

- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Башлыков А.А.МетодПСУБЗ-2013ст

:= *стандартное библиографическое описание**:

[А. А. Башлыков, “Методология построения систем управления базами знаний для интеллектуальных систем,” в *Программные продукты и системы*, 2013, с. 131—137]

⇒ *аннотация**:

[Рассматривается методология построения систем управления БЗ для интеллектуальных систем поддержки принятия решений. Определяются архитектура систем управления БЗ и все ее компоненты. Предлагаемая архитектура систем управления БЗ рассматривается как дальнейшее развитие архитектуры систем управления БД. Излагаются вопросы ориентации предложенной архитектуры на существующие формализмы построения моделей знаний и методы автоматического поиска решений. Описания базируются на формализме семиотической модели знаний, который трактуется как мультимодельная структура представления знаний. Мультимодельность относится к составляющим компонентам семиотической модели и включает семантические сети, фреймы, формальные модели, продукции, вычислительные модели. Выделяются индуктивная и дедуктивная составляющие знаний, описываемых семиотической моделью. Индуктивная составляющая включает средства описания знаний, с помощью которых пользователи системы могут определять проблемные ситуации, описывать мультимодельную структуру знаний предметной области, логику решения задач оперативного принятия решений. В качестве таких средств предлагается формализм языка определения знаний. Основная особенность дедуктивной составляющей семиотической модели - возможность представлять в ней знания о средствах достижения цели управления. Эти знания имеют вид упорядоченного множества процедур, моделей и способов их использования (управления на моделях) для решения возникающих задач в мультимодельной структуре знаний. В основу дедуктивной составляющей положен формализм языка манипулирования знаниями как средства постановки и поиска решений для разрешимых задач поддержки принятия решений. Вводится понятие разрешимости задачи в модели знаний. Приводится описание методов определения разрешимости задачи и планирования ее процессов поиска решений в семиотической модели знаний. Определяются этапы планирования процессов поиска решений: нахождение решающего задачу алгоритма, интерпретация алгоритма для получения результата. Описываются области применения систем управления БЗ.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Башлыков А.А.СистеУБЗ-2010ст

:= *стандартное библиографическое описание**:

[А. А. Башлыков, “Системы управления базами знаний,” в *Автоматизация, телемеханизация и связь в нефтяной промышленности*, 2010, с. 33—39]

⇒ *аннотация**:

[В статье рассмотрены принципы построения систем управления базами знаний. Описание мультимодельной среды хранения знаний предлагается реализовать средствами семиотического моделирования. Приведено описание языков определения и манипулирования знаниями. Описана процедура поиска решений.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Белкин Е.Л.ДидакОУПДвУ-1982кн

:= *стандартное библиографическое описание**:

[Е. Л. Белкин, *Дидактические основы управления познавательной деятельностью в условиях применения технических средств обучения*. Ярославль: Верхне-Волжское книжное издательство, 1982, с. 107]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Белнап Н..ЛогикВиО-1981кн

:= *стандартное библиографическое описание**:

[Н. Белнап и др., “Логика вопросов и ответов,” 1981]

⇐ *библиографическая ссылка**:

- *Глава 3.4. Язык вопросов для ostis-систем*

Белякова М.Л.ИнтелГСдУИТ-2016кн

:= *стандартное библиографическое описание**:

[М. Белякова, *Интеллектуальные геоинформационные системы для управления инфраструктурой транспортных комплексов*. Таганрог: Издательство Южного федерального университета, 2016, 190 л. — Режим доступа: по подписке. - `https://hub.sfedu.ru/repository/material/800749244`. — Дата доступа: 31.03.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*

Березко А..ИнтелГИС-2009ст

:= *стандартное библиографическое описание**:

[А. Березко и др., “Интеллектуальная ГИС,” *Вестник ОНЗ РАН*, т. 1, 2009]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Берестнева В.Г..СетевКМПКМ-2022ст

:= *стандартное библиографическое описание**:

[О. Берестнева и др., “Сетевая концепция метадисциплинарной платформы конструирования множетвенных реальных и виртуальных миров,” *Онтология проектирования*, т. 12, № 2, с. 218—230, июль 2022. DOI: `10.18287/2223-9537-2022-12-2-218-230`. url: `https://doi.org/10.18287/2223-9537-2022-12-2-218-230`]

⇒ *аннотация**:

[В работе предлагается онтологический подход к формированию связей между знаниями, окружающим миром и сценами дополненной и виртуальной реальности.]

⇐ *библиографическая ссылка**:

- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.4. Семантическое представление объектов и сцены*

Беркинблит М.Б.НейроСЭУП-1993кн

:= *стандартное библиографическое описание**:

[М. Б. Беркинблит, *Нейронные сети : эксперим. учеб. пособие*. М.: МИРОС : Всерос. заоч. многопредмет. шк. Рос. акад. наук, 1993]

⇒ *аннотация**:

[Книга представляет собой учебник про искусственным нейронным сетям.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Беркович С.Я..оЭффекПАПвВП-1975ст

:= *стандартное библиографическое описание**:

[С. Беркович, Ю. Кочин и В. Молчанов, “Об эффективности применения ассоциативной памяти в вычислительных процедурах,” *Вычислительные системы. Вып. 62.*, с. 97—105, 1975]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Бир С.Кибер иМ-2006кн

:= *стандартное библиографическое описание**:

[С. Бир, *Кибернетика и менеджмент*, 2-е изд., А. Б. Челюсткин, ред. М.: Комкнига, 2006, с. 280]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Блискавицкий А.А.КонцеПГИСиУ-2012кн

:= *стандартное библиографическое описание**:

[А. А. Блискавицкий, *Концептуальное проектирование ГИС и управление геоинформацией. Технологии интеграции, картографического представления, веб-поиска и распространения геоинформации*. LAP LAMBERT Academic Publishing, 2012, 484 л.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах*

Блискавицкий А.А.СеманГОФГиИ-2014ст

:= *стандартное библиографическое описание**:

[А. А. Блискавицкий, “Семантика геопространственных объектов, функциональная грамматика и интеллектуальные ГИС,” в *Известия высших учебных заведений. Геология и разведка*, 2014, с. 62—69]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах*

Боргест Н.М.СтратИиЕОПР-2019ст

:= *стандартное библиографическое описание**:

[Н. М. Боргест, “Стратегии интеллекта и его онтологии: попытка разобраться,” *Онтология проектирования*, т. 9, № 4 (34), с. 407—428, 2019. DOI: 10.18287/2223-9537-2019-9-4-407-428]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Борисов А.Н.ПострИСОнЗсП-2014ст

:= *стандартное библиографическое описание**:

[А. Н. Борисов, “Построение интеллектуальных систем, основанных на знаниях, с повторным использованием компонентов,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2014) : материалы IV Междунар. науч.-техн. конф., Минск, 20–22 февр. 2014 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2014, с. 97—102]

⇒ *аннотация**:

[В работе рассмотрены теоретические основы и средства построения интеллектуальных систем с многократным использованием компонентов на основе методов инженерной онтологии. Приведены два основных семейства инструментальных средств: модельно-ориентированный подход и подход, использующий декомпозицию решаемой проблемы. Описаны основные языки представления онтологий, а также программные средства построения онтологий. Рассмотрены средства представления декларативных знаний и механизмы обработки процедурных знаний, а также средства взаимодействия онтологических баз знаний и декларативных правил. Проанализированы дискуссионные вопросы развития реляционных языков и онтологической семантики.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Брукс Ф.МифичЧМиКСП-2020кн

:= *стандартное библиографическое описание**:
[Б. Фредерик, *Мифический человеко-месяц, или Как создаются программные системы*. Питер, 2020]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*
- *§ 3.2.6. Комплекс свойств, определяющих эффективность программ в ostis-системах*

Вагин В.Н..Досто иПВвИС-2008кн

:= *стандартное библиографическое описание**:
[В. Н. Вагин и др., *Достоверный и правдоподобный вывод в интеллектуальных системах*, 2-е изд., испр. и доп, под ред. В. Н. Вагина и Д. А. Поспелова, ред. М.: Физматлит, 2008]

⇒ *аннотация**:
[Рассматриваются методы достоверного (дедуктивного) и правдоподобного (абдуктивного, индуктивного) выводов в интеллектуальных системах различного назначения. Приводятся методы дедуктивного вывода на графовых структурах. Описываются как классические, так и немонотонные модальные логики. Приводятся основы теории аргументации и методы абдуктивного вывода.]

⇐ *библиографическая ссылка**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Вайнцвайг М.Н..МеханМиМЕРвРВ-1987ст

:= *стандартное библиографическое описание**:
[М. Вайнцвайг и М. Полякова, “Механизм мышления и моделирование его работы в реальном времени,” *Интеллектуальные процессы и их моделирование*, с. 208—229, 1987]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Вайнцвайг М.Н.ОбучаСИИсАПП-1980кн

:= *стандартное библиографическое описание**:
[М. Вайнцвайг, *Обучающаяся система искусственного интеллекта с ассоциативной памятью-процессором*. АН СССР, Научн. сов. по комплекс. пробл. "Кибернетика", 1980, с. 26]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Валеев С.С..ПострАИвСР-2018ст

:= *стандартное библиографическое описание**:
[С. С. Валеев и И. М. Исмагилова, “Построение адаптивных интерфейсов в сложных распределенных технических системах с применением статистических методов,” *Вестник Уфимского государственного авиационного технического университета*, т. 9, с. 122—130, май 2018]

⇒ *аннотация**:
[Рассматривается задача обеспечения взаимодействия человекаоператора и информационной системы. Проводится анализ используемых методик построения адаптивного интерфейса. Предлагается агентная модель взаимодействия элементов системы с применением сценарного подхода. Применяются статистические критерии для оценки квалификации операторов при построении адаптивных интерфейсов.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Варшавский В.А..ОркесИБДР-1984кн

:= *стандартное библиографическое описание**:
[В. И. Варшавский и Д. А. Поспелов, *Оркестр играет без дирижера: размышления об эволюции некоторых технических систем и управлении ими*. Москва : Наука. Главная редакция физико-математической литературы, 1984]
⇒ *аннотация**:
[Мир, создаваемый человеком в технических системах во многом похож на тот, который окружает человека в природе. И в искусственном мире техники могут происходить процессы, подобные эволюции живых организмов. Возникают колонии и сообщества технических систем, формируются «сверхорганизмы» типа муравейника, возникают «коллективы», живущие по своим законам. Авторы книги анализируют эти аналогии и рассматривают принципы построения управления в таких технических системах, которые во многом отличаются от привычных схем управления. Для чтения книги не требуется никакой специальной подготовки, хотя она обращена не только к так называемому широкому читателю, но и к специалистам, работающим в области управления и кибернетики.]
⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Васильев В.В..ЭлектМЗнГ-1987кн
:= *стандартное библиографическое описание**:
[В. Васильев и Е. Ралдугин, *Электронные модели задач на графах*. Наукова думка, 1987, с. 152]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Владимиров А.Н..ПрогрКУПР-2010ст
:= *стандартное библиографическое описание**:
[А. Н. Владимиров и др., “Программный комплекс "УДАВ": практическая реализация активного обучаемого логического ввода с линейной вычислительной сложностью на основе миварной сети правил,” в *Труды НИИР : сб. науч ст.*, Науч.-исслед. ин-т радио, М., 2010, с. 108—116]
⇒ *аннотация**:
[В статье описан программный комплекс "УДАВ".]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

Волков А.И..ПрофеСвОИТ-2015ст
:= *стандартное библиографическое описание**:
[А. И. Волков, Л. А. Рейнгольд и Е. А. Рейнгольд, “Профессиональные стандарты в области ИТ как фактор технологического и социального развития,” *Прикладная информатика*, т. 10, № 2 (56), с. 37—48, 2015. url: `https://cyberleninka.ru/article/n/professionalnye-standarty-v-oblasti-it-kak-faktor-tehnologicheskogo-i-sotsialnogo-razvitiya`]
⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*

Волченскова Н.И.ТехноМРиЖБ-1984ст
:= *стандартное библиографическое описание**:
[Н. И. Волченскова, “Технология многомашинной реализации и жизнеобеспечения библиотек подпрограмм вычислительной математики на языке Фортран,” 1984]
⇒ *аннотация**:
[Библиотеки подпрограмм создаются и используются уже много лет, однако, и сейчас в этой области существуют еще не решенные проблемы. В качестве одной из форм организации вычислений на ЭВМ авторами предлагалось использование библиотеки подпрограмм, и уже тогда была показана эффективность такого метода программирования.]
⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Гаврилова Т.А..ВизуаМРсЗП-2008ст

:= *стандартное библиографическое описание**:
[Т. А. Гаврилова и Н. А. Гулякина, "Визуальные методы работы со знаниями: попытка обзора," *Искусств. интеллект и принятие решений*, № 1, с. 15—21, 2008]

⇒ *аннотация**:
[Работа содержит описание визуальных методов работы со знаниями: попытка обзора.]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Гаврилова Т.А..БазыЗИСУ-2000кн

:= *стандартное библиографическое описание**:
[Т. А. Гаврилова и В. Ф. Хорошевский, *Базы знаний интеллектуальных систем : учеб. пособие*. СПб. [и др.] : Питер, 2000]

⇒ *аннотация**:
[Учебник для технических вузов по входящим в различные дисциплины вопросам разработки интеллектуальных систем - развивающейся области информатики.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Галушкин А.СовреНРНТвР-1997ст

:= *стандартное библиографическое описание**:
[А. Галушкин, "Современные направления развития нейрокомпьютерных технологий в России," *Открытые системы*, № 4, с. 25—28, 1997]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Гапонов П.А.Модел иМПАП-2000дс

:= *стандартное библиографическое описание**:
[П. А. Гапонов, "Модели и методы параллельной асинхронной переработки информации в графодинамической ассоциативной памяти," 114 л., дис. ... канд. техн. наук : 05.13.11, Минск, 2000]

⇒ *аннотация**:
[Работа содержит описание моделей и методов параллельной асинхронной переработки информации в графодинамической ассоциативной памяти]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Гладков Л.А..ГенетАУП-2006кн

:= *стандартное библиографическое описание**:
[Л. А. Гладков, В. В. Курейчик и В. М. Курейчик, *Генетические алгоритмы : учеб. пособие*, 2-е испр. и доп. М.: Физматлит, 2006]

⇒ *аннотация**:
[Книга содержит описание фундаментальных основ генетических алгоритмов и эволюционного моделирования.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Гладун В.П.ПланиР-1987кн

:= *стандартное библиографическое описание**:
[В. Гладун, *Планирование решений*. Наукова думка, 1987, с. 168]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Гладун В.П.ЭврисПвСС-1977кн

:= *стандартное библиографическое описание**:
[В. Гладун, *Эвристический поиск в сложных средах*. Наукова думка, 1977, с. 164]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Глотов А.А.ИнтелГСПиН-2015ст

:= *стандартное библиографическое описание**:
[А. Глотов, "Интеллектуализация геоинформационных систем: подходы и направления," *ГЕОМАТИКА*, № 4, с. 18—24, 2015]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Глотов А.А.ПримеГТдФС-2014ст

:= *стандартное библиографическое описание**:
[А. Глотов, "Применение геоинформационных технологий для формирования систем поддержки принятия решений в области здравоохранения," *ГЕОМАТИКА*, № 3, с. 38—42, 2014]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Глушков В.М..оРазвитСМЭВМсИЯВУ-1978ст

:= *стандартное библиографическое описание**:
[В. Глушков, С. Погребинский и З. Рабинович, "О развитии структур мультипроцессорных ЭВМ с интерпретацией языков высокого уровня," *Управляющие машины и системы*, № 6, с. 61—66, 1978]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Берштейн Л.С..ОднорПСдРКЛЗнГиГ-1975ст

:= *стандартное библиографическое описание**:
[Л. Берштейн, В. Лисяк и В. Рабинович, "Однородная программируемая структура для решения комбинаторно-логических задач на графах и гиперграфах," *Методы расчета и автоматизация проектирования устройств в микроэлектронных ЦВМ*, с. 39—52, 1975]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Суворов Е.В..ПроцесБЗ-1985ст

:= *стандартное библиографическое описание**:
[Е. Суворов и Я. Фет, "Процессоры баз данных," *Изв. АН СССР. Техн. кибернет*, № 6, с. 63—75, 1985]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Глушков В.М.Кибер-1979ст

:= *стандартное библиографическое описание**:
[В. Глушков, "Кибернетика," 1979, с. 850—856]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*

Глушков В.М.РекурМиВТ-1974кн

:= *стандартное библиографическое описание**:
[В. [д. Глушков, *Рекурсивные машины и вычислительная техника*. ИК АН УССР, 1974, с. 26]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Глушков В.М.ФундаИиТП-1980ст
:= *стандартное библиографическое описание**:
[В. Глушков, "Фундаментальные исследования и технология программирования," *Программирование*, № 2, с. 3—13, 1980]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Голенков В.В..БазовПТЯS-1996кн
:= *стандартное библиографическое описание**:
[В. Голенков и В. Королев, *Базовые преобразования текстов языка SCL для реализации механизмов дедуктивного логического вывода*. Минск: ИТК АН Беларуси, 1996]
⇒ *аннотация**:
[Рассмотрены графодинамические ассоциативные модели обработки информации и соответствующие им абстрактные графодинамические ассоциативные машины, осуществляющие параллельную асинхронную обработку семантических сетей в графодинамической ассоциативной памяти. Одним из перспективных направлений применения предложенных в данной работе моделей являются интеллектуальные обучающие системы и распределенные интеллектуальные системы.]
⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Голенков В.В..ВиртуК-2002ст
:= *стандартное библиографическое описание**:
[В. В. Голенков, Н. А. Гулякина, О. Е. Елисеева и др., "Вирутальная кафедра," *Вышэйшая школа: навукова-метадычны і публіцыстычны часопіс*, № 2, с. 11—14, 2002]
⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Голенков В.В..ВиртуКиИОС-2001ст
:= *стандартное библиографическое описание**:
[В. В. Голенков, В. В. Емельянов и В. Б. Тарасов, "Виртуальные кафедры и интеллектуальные обучающие системы," *Новости искусственного интеллекта*, № 4, с. 3—13, 2001]
⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Голенков В.В..ГрафоАМиСП-2004ст
:= *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, "Графодинамические ассоциативные модели и средства параллельной обработки информации в системах искусственного интеллекта," *Доклады БГУИР*, с. 92—101, 2004]
⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Голенков В.В..ИнтелОСиВУ-2001кн
:= *стандартное библиографическое описание**:
[В. Голенков, В. Тарасов и др., *Интеллектуальные обучающие системы и виртуальные учебные организации: Монография*, В. Голенков и В. Тарасов, ред. Мн.: БГУИР, 2001]
⇒ *аннотация**:
[Рассмотрены графодинамические ассоциативные модели представления и обработки знаний в системах искусственного интеллекта. В основе этих моделей лежит, во-первых, представление всевозможных знаний в виде однородных семантических сетей, имеющих базовую теоретико-множественную интерпретацию, и, во-вторых, трактовка обработки знаний как графодинамического процесса, т.е. как процесса изменения конфигурации семантических сетей.]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

Голенков В.В..ОнтолПГСС-2019ст

≔ *стандартное библиографическое описание**:
[В. В. Голенков, Н. А. Гулякина, И. Т. ДАавыденко и др., "Онтологическое проектирование гибридных семантически совместимых интеллектуальных систем на основе смыслового представления знаний," в *Онтология проектирования*, 2019, с. 132—151]

⇐ *библиографическая ссылка**:
- *Глава 2.1. Информационные конструкции и языки*
- *§ 2.1.1. Формализация понятия информационной конструкции*

Голенков В.В..оОбучеИСкОС-2018ст

≔ *стандартное библиографическое описание**:
[В. В. Голенков, Н. А. Гулякина, Н. В. Гракова и др., "От обучения интеллектуальных систем к обучению средств их разработки," в *Открытые семантические технологии проектирования интеллектуальных систем*, сер. Вып. 2, Минск : БГУИР, 2018, с. 81—98]

⇒ *аннотация**:
[Работа содержит описание обучения интеллектуальных систем и их средств разработки]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Библиография OSTIS*
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.1 Проблемы адаптивного управления производственной деятельностью при разработке интеллектуальных компьютерных систем нового поколения*

Голенков В.В..ПроекОСТКПИСЧ2-2014ст

≔ *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, "Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 2: Унифицированные модели проектирования," *Онтология проектирования*, № 4, с. 34—53, 2014]

⇒ *аннотация**:
[Работа содержит описание проекта открытой семантической технологии компонентного проектирования интеллектуальных систем, унифицированных моделей проектирования, обработки знаний, интерфейсов и семантического компьютера]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.2.1 Предлагаемый подход к автоматической генерации тестовых вопросов*
- *Пункт 7.5.3.4. Семантическая модель решателя задач подсистемы контроля знаний*
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *§ 2.3.2. Язык внешнего графического представления конструкций SC-кода — SCg-код (Semantic Code graphical)*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Голенков В.В..ПроекПРПнОО-2017ст

≔ *стандартное библиографическое описание**:
[В. Таберко и др., "Проектирование предприятий рецептурного производства на основе онтологий," т. 7, № 24, с. 123—144, 2017. url: `https://libeldoc.bsuir.by/handle/123456789/29428`]

⇒ *аннотация**:
[Статья содержит описание подхода к проектированию предприятий на основе онтологий]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.1 Проблемы разработки систем комплексной автоматизации*
- *Подпункт 7.7.2.2.1 Подходы к автоматизации предприятий*

Голенков В.В..СеманМПиОБ-2017ст

:= *стандартное библиографическое описание**:

[В. В. Голенков, Н. А. Гулякина, И. Т. Давыденко и др., "Семантическая модель представления и обработки баз знаний," Л. А. Калиниченко и др., ред., ФИЦ ИУ РАН, окт. 2017, с. 412—419]

⇒ *аннотация**:

[Предложен подход к созданию интеллектуальных систем, ориентированных на решение комплексных задач, в основе которого лежат семантические модели баз знаний и согласованные с ними семантические модели машин обработки базы знаний. Основой для построения указанных моделей является унифицированное смысловое представление знаний на основе универсального языка семантических сетей с теоретико-множественной интерпретацией. На базе указанного языка построено открытое семейство совместимых языков, семантика каждого из которых задается соответствующей онтологией. Семантическая модель машины обработки базы знаний построена на базе многоагентного подхода, предполагающего, что агенты взаимодействуют между собой через общую для них семантическую память.]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

Голенков В.В..СеманУиВК-2004ст

:= *стандартное библиографическое описание**:

[В. В. Голенков и Н. А. Гулякина, "Семантические учебники и виртуальные кафедры," *Статистика и экономика*, № 3, с. 2—5, 2004]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Голенков В.В..СтандОТОП-2021кн

:= *стандартное библиографическое описание**:

[В. Голенков, Н. Гулякина и Д. Шункевич, *Стандарт открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем*, В. Голенков, ред. Минск: Бестпринт, 2021, с. 690]

⇒ *аннотация**:

[В книге представлена документация текущей версии стандарта открытой технологии онтологического проектирования, производства и эксплуатации семантически совместимых гибридных интеллектуальных компьютерных систем (Технологии OSTIS). Описание Технологии OSTIS (Open Semantic Technology for Intelligent Systems) представлено в виде раздела базы знаний ostis-системы (системы, построенной по Технологии OSTIS) на внутреннем языке ostis-системы и обладает достаточной полнотой для использования этой технологии разработчиками интеллектуальных компьютерных систем. Предложена стандартизация интеллектуальных компьютерных систем, а также стандартизация методов и средств их проектирования, что является важнейшим фактором, обеспечивающим семантическую совместимость интеллектуальных компьютерных систем и их компонентов]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.6.3. Грамматика SC-JSON-кода*
- *§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS*
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.2. Предлагаемый подход к созданию умных домов*
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.1.5. Понятие сложного sc-идентификатора sc-элемента*
- *Пункт 2.3.2.3. Иерархическое семейство подъязыков, семантически эквивалентных SCg-коду*
- *Пункт 2.3.3.1. Синтаксис SCs-кода*
- *§ 2.3.4. Язык внешнего форматированного представления конструкций SC-кода — SCn-код (Semantic Code natural)*
- *Глава 2.1. Информационные конструкции и языки*
- *§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

- *Глава 1.3. Принципы, лежащие в основе технологии комплексной поддержки жизненного цикла интеллектуальных компьютерных систем нового поколения*
- *§ 1.3.1. Технология OSTIS (Open Semantic Technology for Intelligent Systems)*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *§ 3.1.4. Понятие метода*
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Голенков В.В..СтрукСП-2014ст

:= *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, “Структуризация смыслового пространства,” в *Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS-2014) : материалы IV междунар. науч.-техн. конф. (Минск, 20-22 февраля 2014 года)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2014, с. 65—78]

⇒ *аннотация**:
[Работа содержит описание структуризации смыслового пространства]

⇐ *библиографическая ссылка**:
- *Глава 2.1. Информационные конструкции и языки*
- *§ 2.1.1. Формализация понятия информационной конструкции*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Голенков В.В..ЭлектУНПО-2006ст

:= *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, “Электронные учебники нового поколения, основанные на применении технологий искусственного интеллекта,” *Известия Белорусской инженерной академии*, № 1 (21)/3, с. 75—95, 2006]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Голенков В.В.ГрафоМиМПА-1996дс

:= *стандартное библиографическое описание**:
[В. В. Голенков, “Графодинамические модели и методы параллельной асинхронной переработки информации в интеллектуальных системах,” 396 л., дис. ... д-ра техн. наук : 05.13.11 ; 05.13.17, Минск, 1996]

⇒ *аннотация**:
[Докторская диссертация, посвященная графодинамическим моделям и методам параллельной асихронной переработки информации в интеллектуальных системах]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *§ 6.2.3. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для ostis-систем*
- *§ 6.2.4. Архитектура ассоциативных семантических компьютеров для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*
- *Пункт 6.2.4.2. Вариант крупнозернистой архитектуры ассоциативных семантических компьютеров*

Голенков В.В.ПаралГКОнРЗИИиеП-1994кн

:= *стандартное библиографическое описание**:
[В. В. Голенков, *Параллельный графовый компьютер (PGC), ориентированный на решение задач искусственного интеллекта, и его применение (Препринт; No 2)*. Институт технической кибернетики, 1994, с. 59]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Голенков В.В.СтрукОиПИвЭММУПСД-1996дс

:= *стандартное библиографическое описание**:

[В. В. Голенков, “Структурная организация и переработка информации в электронных математических машинах, управляемых потоком сложноструктурированных данных,” дис. ... канд. техн. наук : 05.13.01 ; 05.13.13, Минск, 1996]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Голенков В.В.ТермиМПГКИсПиПМСЛОМпМОЭВМ-1994кн

:= *стандартное библиографическое описание**:

[В. [д. Голенков, *Терминальный модуль параллельного графового компьютера (PGC): Интерфейс с пользователем и процессорными модулями, структура, логическая организация: материалы по математическому обеспечению ЭВМ*. Институт технической кибернетики, 1994, с. 43]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *Пункт 6.2.4.2. Вариант крупнозернистой архитектуры ассоциативных семантических компьютеров*

Голенков В.В..ГрафоМПОЗ-2012ст

:= *стандартное библиографическое описание**:

[В. В. Голенков и Н. А. Гулякина, “Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2012) : материалы II Междунар. науч.-техн. конф., Минск, 16–18 февр. 2012 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2012, с. 23—52]

⇒ *аннотация**:

[Работа содержит описание графодинамических моделей параллельной обработки знаний, их принципов построения, реализации и проектирования]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*
- *§ 3.2.2. Существующие онтологии языков программирования*

Голенков В.В..ОткрыПНнСТ-2013ст

:= *стандартное библиографическое описание**:

[В. В. Голенков и Н. А. Гулякина, “Открытый проект, направленный на создание технологии компонентного проектирования интеллектуальных систем,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 55—78]

⇒ *аннотация**:

[Работа содержит описание открытого проекта, направленного на создание технологии компонентного проектирования интеллектуальных систем]

⇐ *библиографическая ссылка**:

- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*

Голенков В.В..ПринцПМСТ-2011ст

:= *стандартное библиографическое описание**:

[В. В. Голенков и Н. А. Гулякина, “Принципы построения массовой семантической технологии компонентного проектирования интеллектуальных систем,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12 февраля 2011г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 21—58]

⇒ *аннотация**:

[Работа содержит описание принципов построения массовой семантической технологии компонентного проектирования интеллектуальных систем]

⇐ *библиографическая ссылка**:

- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Голенков В.В..ПроекОСТКПИСЧ1-2014ст

:= *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, "Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 1: Принципы создания," *Онтология проектирования*, № 1, с. 42—64, 2014]

⇒ *аннотация**:
[Работа содержит описание проекта открытой семантической технологии компонентного проектирования интеллектуальных систем, его принципы создания, общие положения и представление знаний]

⇐ *библиографическая ссылка**:
- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*

Голенков В.В..СеманТКПС-2015ст

:= *стандартное библиографическое описание**:
[В. В. Голенков и Н. А. Гулякина, "Семантическая технология компонентного проектирования систем, управляемых знаниями," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 57—78]

⇒ *аннотация**:
[Работа содержит описание семантической технологии компонентного проектирования систем, управляемых знаниями]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

Головатая Е.А.МоделФИдТРСпДВИ-2019ст

:= *стандартное библиографическое описание**:
[Е. Головатая и В. Садов, "Модель формирования изображений для трехмерной реконструкции сцен по данным видеоэндоскопических исследований," *Вестник Полоцкого государственного университета*, № 12, с. 43—49, 2019]

⇒ *аннотация**:
[В работе рассматриваются геометрические модели формирования изображений для последующей трехмерной реконструкции, а также предлагается модель широкоугольной сферической проекции для обработки изображений, полученных с использованием широкоугольных линз, например, из видеоэндоскопических систем.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.1. Оптические системы компьютерного зрения*

Головко В.А..НейроТОД-2017кн

:= *стандартное библиографическое описание**:
[В. А. Головко и В. В. Краснопрошин, *Нейросетевые технологии обработки данных*. Минск: Издательство БГУ, 2017, ISBN: 9789855664674]

⇒ *аннотация**:
[Изложены математические и алгоритмические аспекты функционирования искусственных нейронных сетей. Детально проанализированы как конвенциальные модели нейтронных сетей, так и глубокие нейтронные сети. Рассмотрены парадигмы обучения нейтронных сетей. Большое внимание уделено применению нейтронных сетей для решения различного рода задач]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *Пункт 3.6.1.1. Денотационная семантика моделей искусственных нейронных сетей, используемых в ostis-системах*
- *Пункт 3.6.1.2. Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах*

Головко В.А..ПринцПСПР-2019ст

:= *стандартное библиографическое описание**:

[В. А. Головко, В. А. Крощенко и др., “Принципы построения систем принятия решений на основе интеграции нейросетевых и семантических моделей,” в *Открытые семантические технологии проектирования интеллектуальных систем*, сер. Вып. 3, Минск : БГУИР, 2019, с. 91—102]

⇒ *аннотация**:

[В работе описываются принципы построения систем принятия решений на основе интеграции нейросетевых и семантических моделей]

⇐ *библиографическая ссылка**:

- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Головко В.А.НейроСООиП-2001кн

:= *стандартное библиографическое описание**:

[В. А. Головко, *Нейронные сети: обучение, организация и применение : учеб. пособие*. М.: Журн. «Радиотехника», 2001, (Нейрокомпьютеры и применение ; кн. 4)]

⇒ *аннотация**:

[В книге изложены математические и алгоритмические аспекты функционирования нейронных сетей с прямыми и обратными связями; отражены вопросы самоорганизации, отказоустойчивости и реализации нейронных сетей на систолических процессорах; большое внимание уделено применению и проектированию нейронных сетей для решения различного рода задач]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Горбань А.Н..НейроСнПК-1996кн

:= *стандартное библиографическое описание**:

[А. Н. Горбань и Д. А. Россиев, *Нейронные сети на персональном компьютере*. Новосибирск: Наука, 1996]

⇒ *аннотация**:

[В книге описаны элементы нейронных сетей, способы их соединения и настройки для решения различных задач, набор дополнительных элементов, необходимых для создания полноценного нейрокомпьютера, и различные алгоритмы обучения нейронных сетей]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Гордей А.Н.ТеориАПАЗ-2014ст

:= *стандартное библиографическое описание**:

[А. Гордей, “Теория автоматического порождения архитектуры знаний (ТАПАЗ-2) и дальнейшая минимизация семантических исчислений / А. Н. Гордей,” в *Открытые семантические технологии проектирования интеллектуальных систем = Open Semantic Technologies for Intelligent Systems (OSTIS-2014) : материалы IV междунар. науч.-техн. конф. (Минск, 20-22 февраля 2014 года)*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2014, с. 49—64]

⇒ *аннотация**:

[При помощи геометрического метода объясняется приоритет модели мира над языковой картиной мира, излагается новая версия теории автоматического порождения архитектуры знаний (ТАПАЗ-2), предлагается алгебраический аппарат для исчисления семантики предметных областей и процедурального представления и преобразования знаний в искусственных интеллектуальных системах, приводятся примеры формального описания русской глагольной семантики в зависимости от суперпозиции процессов познания.]

⇐ *библиографическая ссылка**:

- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Городецкий В.И..БазовОКПА-2015ст

:= *стандартное библиографическое описание**:

[В. И. Городецкий, В. В. Самойлов и Д. В. Троцкий, “Базовая онтология коллективного поведения автономных агентов и ее расширения,” *Изв. Рос. акад. наук. Теория и системы упр.*, № 5, с. 102—121, 2015]

⇒ *аннотация**:

[В работе рассмотрен онтологический подход к проектированию многоагентных систем.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Горшков С.В.Введе вОМ-2016кн

≔ *стандартное библиографическое описание**:
[С. В. Горшков, *Введение в онтологическое моделирование*. Екатеринбург : ТриниДата, 2016]

⇒ *аннотация**:
[Методическое пособие предназначено для аналитиков, ИТспециалистов, менеджеров, участвующих в создании информационных систем, использующих семантические технологии для решения прикладных задач, студентов соответствующих специальностей.]

⇐ *библиографическая ссылка**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

Гракова Н.В..МоделСОКиД-2015ст

≔ *стандартное библиографическое описание**:
[Н. В. Гракова, Д. И. Коновал и В. С. Семенов, "Модель структуры описания курсового и дипломного проектирования, представленная на языке семантических сетей," в *Дистанционное обучение – образовательная среда XXI века : материалы IX международной научно-методической конференции (Минск, 3-4 декабря 2015 года)*, БГУИР, Минск, 2015, с. 155—156]

⇒ *аннотация**:
[В статье рассматривается модель организации управления курсовым и дипломным проектированием студентов вуза.]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Грибова В.В..АвтомРПИсД-2011ст

≔ *стандартное библиографическое описание**:
[В. В. Грибова и А. В. Тарасов, "Автоматизация разработки пользовательских интерфейсов с динамическими данными," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12 февраля 2011г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 287—292]

⇒ *аннотация**:
[Работа содержит описание принципов автоматического построения пользовательских интерфейсов]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Грибова В.В..БазовТРИС-2015ст

≔ *стандартное библиографическое описание**:
[В. В. Грибова, А. С. Клещев, Д. А. Крылов, Ф. М. Москаленко, В. А. Тимченко и др., "Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Ч. 1. Разработка базы знаний и решателя задач," *Програм. инженерия*, № 12, с. 3—11, 2015]

⇒ *аннотация**:
[В данной работе описана технология разработки базы знаний и решателя задач в рамках платформы IACPaaS.]

⇐ *библиографическая ссылка**:
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Грибова В.В..ГенерКПИУ-2005ст

≔ *стандартное библиографическое описание**:
[В. В. Грибова и А. В. Тарасов, "Генератор кода пользовательского интерфейса, управляемый онтологией," в *Искусственный интеллект, Т.4*, 2005, с. 457—464]

⇒ *аннотация**:
[В данной работе представлен новый подход к реализации инструментария для разработки пользовательского интерфейса, управляемого онтологиями. Цель данного подхода - обеспечить разработчика расширяемым инструментарием. Это связано с быстрым развитием области знаний, связанной с интерфейсом, и, соответственно, быстрым устареванием инструментария. В работе описана основная идея подхода, метод генерации кода пользовательского интерфейса, управляемый онтологиями.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Грибова В.В..Онтол дРиГАП-2022ст

:= *стандартное библиографическое описание**:
[В. В. Грибова, С. В. Паршкова и Л. Федорищев, "Онтология для разработки и генерации адаптивных пользовательских интерфейсов редакторов баз знаний," т. 12, с. 200—217, 2022]

⇒ *аннотация**:
[В статье рассматривается метод создания автоматически генерируемых адаптивных пользовательских интерфейсов редакторов баз знаний, построенных на основе онтологического подхода, с целью улучшения качества работы по формированию и редактированию баз знаний с учетом специфики предметной области, характеристик пользователя-эксперта и других параметров. Приведено описание концепции авторского подхода к генерации адаптивных интерфейсов используемых онтологий, баз знаний и моделей, как ключевых элементов предложенного подхода.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Грибова В.В..ПроекIACP-2011ст

:= *стандартное библиографическое описание**:
[В. В. Грибова, А. С. Клещев, Д. А. Крылов, Ф. М. Москаленко, С. В. Смагин и др., "Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений," *Искусств. интеллект и принятие решений*, № 1, с. 27—35, 2011]

⇒ *аннотация**:
[В данной работе описан проект IACPaaS и комплекс интеллектуальных систем на основе облачных вычислений.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Грибова В.В..Платф дРОИС-2016ст

:= *стандартное библиографическое описание**:
[В. В. Грибова, А. С. Клещев, Ф. М. Москаленко и др., "Платформа для разработки облачных интеллектуальных сервисов," в *XV Национальная конференция по искусственному интеллекту с международным участием (КИИ-2016), Смоленск, 3–7 октября 2016 г. : труды : в 3 т.*, Рос. ассоц. искусств. интеллекта, Федер. исслед. центр «Информатика и управление» Рос. акад. наук, т. 1, Смоленск, 2016, с. 24—33]

⇒ *аннотация**:
[В работе описана платформа для разработки облачных интеллектуальных сервисов.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*

Губаревич А.В..ОнтолПИСвОИ-2017ст

:= *стандартное библиографическое описание**:
[А. В. Губаревич, О. Л. Моросин и Д. В. Ланде, "Онтологическое проектирование интеллектуальных систем в области истории," в *Открытые семантические технологии проектирования интеллектуальных систем*, сер. Вып. 1, Минск : БГУИР, 2017, с. 245—250]

⇒ *аннотация**:
[В работе описывается онтологическое проектирование интеллектуальных систем в области истории.]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*

Губаревич А.В..СтрукБЗвИСпИ-2018ст

:= *стандартное библиографическое описание**:
[А. В. Губаревич, С. П. Витязь и Р. Б. Григянец, "Структура баз знаний в интеллектуальных системах по истории," в *Открытые семантические технологии проектирования интеллектуальных систем*, сер. Вып. 2, Минск : БГУИР, 2018, с. 347—350]

⇒ *аннотация**:
[В работе рассматривается структура баз знаний в интеллектуальных системах по истории.]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*

Гудфеллоу Я..ГлубоО-2017кн

≔ *стандартное библиографическое описание**:
[Я. Гудфеллоу, И. Бенджио и А. Курвилль, *Глубокое обучение*. Москва: ДМК Пресс, 2017, ISBN: 9785970605547]

⇒ *аннотация**:
[Рассмотрены основы дисциплин, необходимых для понимания глубокого обучения. Описываются различные приложения глубокого обучения, а также основные приемы глубокого обучения]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

Гузаревич Я.В..ВнедрСУДнПМ-2023эл

≔ *стандартное библиографическое описание**:
[Я. В. Гузаревич и др. “Внедрение системы умного дома на примере многоквартирного жилого дома.” (), url: `https://rep.bntu.by/handle/data/100103`]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Гулякина Н.А..МетодПСМИ-2013ст

≔ *стандартное библиографическое описание**:
[Н. А. Гулякина, И. Т. и Д. В. Шункевич, “Методика проектирования семантической модели интеллектуальной справочной системы, основанные на семантических сетях,” *Программные системы и вычислительные методы*, № 1, с. 56—68, 2013]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Гулякина Н.А..ФормаОМПИ-2011ст

≔ *стандартное библиографическое описание**:
[Н. А. Гулякина, Т. А. Гаврилова и В. В. Голенков, “Формальная основа модульного проектирования интеллекутальных обучающих систем,” в *Дистанционное обучение — образовательная среда XXI века: Материалы VII Международной научно-методической конференции, 1-2 дек. 2011 г.*, 2011, с. 224—226]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Гулякина Н.А..ЭлектУНПО-2006ст

≔ *стандартное библиографическое описание**:
[Н. А. Гулякина и В. В. Голенков, “Электронные учебники нового поколения, основанные на применении технологий искусственного интеллекта,” *Известия Белорусской анженерной академии*, № 1 (21)/3, с. 75—95, 2006]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Гулякина Н.А..Языки иТПОнО-2012ст

≔ *стандартное библиографическое описание**:
[Н. А. Гулякина, О. В. Пивоварчик и Д. А. Лазуркин, “Языки и технологии программирования, ориентированные на обработку семантических сетей,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2012) : материалы II междунар. науч.-техн. конф., Минск, 16–18 февр. 2012 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2012, с. 221—228]

⇒ *аннотация**:
[В работе рассматриваются языки и технологии программирования, ориентированные на обработку семантических сетей]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*

Гулякина Н.А.ИнтелТвЭО-2004ст

:= *стандартное библиографическое описание**:

[Н. А. Гулякина, “Интеллектуальные технологии в электронном образовании,” в *Дистанционное обучение — образовательная среда XXI века: Материалы IV Международной научно-методической конференции, 10-12 нояб. 2004 г.*, Мн.: БГУИР, 2004, с. 13—19]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Гулякина Н.А.КорпоСВК-2003ст

:= *стандартное библиографическое описание**:

[Н. А. Гулякина, “Корпоративные системы виртуальных кафедр,” *Доклады БГУИР*, т. 1, № 2/2, с. 15—25, 2003]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Давыденко И.Т..СеманСБЗИ-2015ст

:= *стандартное библиографическое описание**:

[И. Т. Давыденко и Е. А. Дюбина, “Семантическая структуризация базы знаний интеллектуальной справочной системы по геометрии,” в *Дистанционное обучение – образовательная среда XXI века : материалы IX международной научно-методической конференции (Минск, 3-4 декабря 2015 года)*, БГУИР, Минск, 2015, с. 153—154]

⇒ *аннотация**:

[В статье рассматривается семантический подход к структурированию базы знаний интеллектуальных систем на примере интеллектуальной системы по геометрии.]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Давыденко И.Т.РазраИОСнОТ-2013ст

:= *стандартное библиографическое описание**:

[И. Давыденко, “Разработка интеллектуальных обучающих систем на основе технологии OSTIS,” в *Дистанционное обучение – образовательная среда XXI века: материалы VIII международной научно-методической конференции. (Минск, 5–6 декабря 2013 года)*, БГУИР, Минск, 2013, с. 196—197]

⇒ *аннотация**:

[В работе рассматривается принципы разработки интеллектуальных обучающих систем на основе открытой семантической технологии проектирования интеллектуальных систем OSTIS (Open Semantic Technology for intelligent systems), использующей в качестве способа кодирования информации семантическую сеть с базовой теоретико-множественной интерпретацией. Основными принципами данного подхода являются поэтапное эволюционное проектирование баз знаний на основе быстрого прототипирования, модульное проектирование на основе библиотек типовых многократно используемых компонентов, ориентация на семантическое представление знаний.]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Давыденко И.Т.ТехноКПБЗнО-2013ст

:= *стандартное библиографическое описание**:

[И. Давыденко, “Технология компонентного проектирования баз знаний на основе унифицированных семантических сетей,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 511—516]

⇒ *аннотация**:

[В работе рассматривается состав семантической технологии проектирования баз знаний интеллектуальных систем. Данная технология ориентирована на семантическое представление знаний, расширение контингента разработчиков баз знаний и сокращение сроков проектирования.]

⇐ *библиографическая ссылка**:

- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*

- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*

Давыденко И.Т..ИнтелССГ-2011ст

:= *стандартное библиографическое описание**:
[И. Т. Давыденко, В. А. Житко и др., “Интеллектуальная справочная система по геометрии,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. науч.-техн. конф., Минск, 10–12 февр. 2011 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 463—482]

⇒ *аннотация**:
[В работе приводится описание интеллектуальной справочной системы по геометрии, спроектированной на основе открытой семантической технологии компонентного проектирования интеллектуальных систем.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Давыденко.И.Т..СредсССМБЗ-2016ст

:= *стандартное библиографическое описание**:
[И. Т. Давыденко, Н. В. Гракова и др., “Средства структуризации семантических моделей баз знаний,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 93—106]

⇒ *аннотация**:
[Работа содержит описание средств структуризации семантических моделей баз знаний]

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

Давыденко И.Т.МоделМиСРГБ-2017дс

:= *стандартное библиографическое описание**:
[И. Т. Давыденко, “Модели, методика и средства разработки гибридных баз знаний на основе семантической совместимости многократно используемых компонентов,” 284 л., дис. ... канд. техн. наук : 05.13.17, Минск, 2017]

⇒ *аннотация**:
[Работа содержит описание моделей, методики и средст разработки гибридных баз знаний на основе семантической совместимости многократно используемых компонентов]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *§ 4.2.3. Методика и средства разработки естественно-языковых интерфейсов*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем*

Давыденко.И.Т.СеманМКПБЗ-2016ст

:= *стандартное библиографическое описание**:
[И. Т. Давыденко, “Семантическая модель коллективного проектирования баз знаний,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 107—114]

⇒ *аннотация**:
[Работа содержит описание семантической модели коллективного проектирования баз знаний]

⇐ *библиографическая ссылка**:
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*

Деменков Н.П.НечетУвТСУП-2005кн

:= *стандартное библиографическое описание**:
[Н. П. Деменков, *Нечеткое управление в технических системах : учеб. пособие*. М.: Изд-во Моск. гос. техн. ун-та, 2005]

⇒ *аннотация**:

[Рассматриваются вопросы, связанные с использованием теории нечеткого управления для решения неформализорванных задач оптимизации в технических системах]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Дементьев А.В.МетриСД-2022ст

≔ *стандартное библиографическое описание**:
[А. В. Дементьев, "Метрики семантических данных," *Молодой ученый*, № 24(419), с. 48—51, июнь 2022]

⇒ *аннотация**:
[Данная статья повествует о семантической метрике извлечения перечня понятий из текстов на соответствующую тематику. Онтологический анализ представляет основу для данной метрики. Для обнаружения семантической метрики применяется два показателя – показатель вложенных взаимодействий и тезаурусный подход.]

⇐ *библиографическая ссылка**:
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

Добров Б.В..ОнтолТМИП-2009кн

≔ *стандартное библиографическое описание**:
[Б. В. Добров и др., *Онтологии и тезаурусы: модели, инструменты, приложения : учеб. пособие*. М. : Интернет-Ун-т Информ. технологий : БИНОМ. Лаб. знаний, 2009]

⇒ *аннотация**:
[В книге впервые на русском языке в систематизированной форме излагаются теоретические и практические вопросы использования онтологий и тезаурусов как способов организации информации и знаний в современных информационных системах.]

⇐ *библиографическая ссылка**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

Драгалин А.Г.КонстТДиНА-2003кн

≔ *стандартное библиографическое описание**:
[А.Г.Драгалин, *Конструктивная теория доказательств и нестандартный анализ*, russian,english, Г. Е. Минц и др., ред. Едиториал УРСС, 2003, ISBN: 5-354-00388-1]

⇒ *аннотация**:
[В данной книге представлены обсуждения бизнес потенциала обработки знаний и рассматриваются аспекты прикладной технологии искусственого интеллекта.]

⇐ *библиографическая ссылка**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Дружинин В.Н..КогниПУдВ-2002кн

≔ *стандартное библиографическое описание**:
[В. Н. Дружинин и Д. В. Ушаков, *Когнитивная психология. Учебник для вузов*. М.: ПЕР СЭ, 2002]

⇒ *аннотация**:
[Учебник подготовлен авторским коллективом преподавателей Института психологии Государственного университета гуманитарных наук, Московской гуманитарно-социальной академии и сотрудников Института психологии РАН. В нем подробно изложены основы психологии познания как составляющей курса «Общей психологии» с учетом последних научных результатов, полученных отечественными и зарубежными исследователями.]

⇐ *библиографическая ссылка**:
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

Дулин С.К..ПроблОСГиСП-2016ст

≔ *стандартное библиографическое описание**:
[С. К. Дулин, Н. Г. Дулина и Д. А. Никишин, "Проблемы обеспечения семантической геоинтероперабельности и согласования понимания семантики геоданных," *Системы и средства информатики*, т. 26, № 1, с. 86—108, 2016]

⇒ *аннотация**:

[Основной задачей работы ставилась выработка целостного и по возможности всестороннего взгляда на проблему интероперабельности (ИО, англ. Interoperability) взаимодействующих информационных систем (ИС) вообще и семантического аспекта этой проблемы в частности. Рассматриваются основные причины возникновения проблемы межсистемной ИО, ее различные определения и подходы к ее структуризации, на основании чего сделана попытка структурировать проблему ИО, выявляются неоднозначные моменты и предлагаются возможные варианты их устранения. Анализируется состояние дел в области стандартизации метаданных, на основании чего сделан вывод о необходимости стандартизации форм и способов выполнения поисковых запросов и межсистемных функциональных вызовов. Рассматриваются основные моменты проблемы понимания смысла данных, существующие подходы к ее структурированию и решению.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Дьяконов В.Maple вМР-2022кн

:= *стандартное библиографическое описание**:

[В. Дьяконов, *Maple 10/11/12/13/14 в математических расчетах*. Litres, 2022]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Дьяконов В.MatheПР-2022кн

:= *стандартное библиографическое описание**:

[В. Дьяконов, *Mathematica 5/6/7. Полное руководство*. Litres, 2022]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Дьяконов В.MatlaПС-2022кн

:= *стандартное библиографическое описание**:

[В. Дьяконов, *MATLAB. Полный самоучитель*. Litres, 2022]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Дьяконов В.ЭнцикКА-2022кн

:= *стандартное библиографическое описание**:

[В. Дьяконов, *Энциклопедия компьютерной алгебры*. Litres, 2022]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*

Елисеева О.Е..КомпоПИОС-2013ст

:= *стандартное библиографическое описание**:

[О. Е. Елисеева и К. В. Русецкий, “Компонентное проектирование интеллектуальной обучающей системы для подготовки школьников к централизованному тестированию по иностранным языкам,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 511—516]

⇒ *аннотация**:

[В работе представлено компонентное проектирование интеллектуальной обучающей системы для подготовки к централизованному тестированию по иностранному языку. Выделенные компоненты ИОС позволяют выполнять разработку отдельных ее частей почти автономно с участием специалистов различного профиля.]

⇐ *библиографическая ссылка**:

- *§ 7.5.2. Автоматизация среднего образования с помощью ostis-систем*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Емельянов В.В..Теори иПЭМ-2003кн

:= *стандартное библиографическое описание**:
[В. В. Емельянов, В. В. Курейчик и В. М. Курейчик, *Теория и практика эволюционного моделирования*. М.: Физматлит, 2003]

⇒ *аннотация**:
[Книга содержит описание генетических и синергетических подходов, а также средства эволюционного моделирования.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Еремеев А.П.ПострРФнБТ-1997ст

:= *стандартное библиографическое описание**:
[А. П. Еремеев, "Построение решающих функций на базе тернарной логики в системах принятия решений в условиях неопределенности," *Изв. Рос. акад. наук. Теория и системы упр.*, № 2, с. 138—143, 1997]

⇒ *аннотация**:
[В данной работе рассматриваются проблемы существующих методов, средств и технологий построения машин обработки знаний и ставится проблема отсутствия средств, позволяющих относительно неподготовленному разработчику в удовлетворительные сроки проектировать машины обработки знаний для прикладных интеллектуальных систем различного назначения]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Ершов А.П.АлгорМОиАМВС-1982кн

:= *стандартное библиографическое описание**:
[А. Ершов, *Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем*. Наука, 1982, с. 336]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Ефименко И.В..УСКМТЛС-2014ст

:= *стандартное библиографическое описание**:
[И. В. Ефименко и В. Ф. Хорошевский, "УСК Мартынова — тридцать лет спустя," в *Открытые семантические технологии проектирования интеллектуальныхсистем = Open Semantic Technologies for Intelligent Systems (OSTIS-2014) :материалы IV междунар. науч.-техн. конф. (Минск, 20-22 февраля 2014 года)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2014, с. 29—38]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Ефимов Е.И.РешатИЗ-1982кн

:= *стандартное библиографическое описание**:
[Е. И. Ефимов, *Решатели интеллектуальных задач*. М.: Наука, 1982]

⇒ *аннотация**:
[Книга посвящана рассмотрению элементов теории интеллектуальных решателей и ее практических приложений]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Житко В.А..СеманТКПЕЯИИВОС-2011ст

:= *стандартное библиографическое описание**:
[В. А. Житко и др., "Семантическая технология компонентного проектирования естественно-языкового интерфейса интеллектуальных вопросно-ответных систем," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12*

февраля 2011г., Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 395—408]

⇒ *аннотация**:

[В данной работе рассматривается семантическая технология проектирования естественно-языковоыхх интерфейсов для интеллектуальных вопросно-ответных систем. Также рассматривается библиотека компонентов проектирования естественно-языкового интерфейса, её пополнение, как сторонними компонентами, так и создание новых компонентов.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*

Житко В.А.ТехноКПСНиПвСС-2011ст

:= *стандартное библиографическое описание**:

[В. А. Житко, "Технология компонентного проектирования средств навигации и поиска в семантических сетях," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12 февраля 2011г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 255—265]

⇒ *аннотация**:

[В данной работе рассматривается семантическая технология компонентного проектирования средства навигации и поиска в семантических сетях.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*

Журавков М.А..ГИСте пДПИСКГИМ-2004кн

:= *стандартное библиографическое описание**:

[М. Журавков, В. Видякин и О. К. и др, *ГИС-технологии при добыче полезных ископаемых. Специализированная корпоративная геоинформационная система "МарManager"*. Минск: Изд. центр БГУ, 2004, 208 л. — Под общ. ред. М.А. Журавкова.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Загорулько Ю.А..оФормаСОЗвИиИСнОО-2014ст

:= *стандартное библиографическое описание**:

[Ю. А. Загорулько и Г. Б. Загорулько, "О формализации семантики областей знаний в информационных и интеллектуальных системах на основе онтологий," в *Открытые семантические технологии проектирования интеллектуальныхсистем = Open Semantic Technologies for Intelligent Systems (OSTIS-2014) :материалы IV междунар. науч.-техн. конф. (Минск, 20-22 февраля 2014 года)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2014, с. 117—130]

⇐ *библиографическая ссылка**:

- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Задыхайло И.Б.ПроекАППнЦМДОнПРБД-1979кн

:= *стандартное библиографическое описание**:

[И. [д. Задыхайло, *Проект ассоциативного параллельного процессора на ЦМД, ориентированного на поддержку реляционных баз данных*. Акад. наук СССР. Ин-т прикладной математики, 1979, с. 25]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Заливако С.С..СеманТКПИРЗ-2011ст

:= *стандартное библиографическое описание**:

[С. Заливако, О. Савельева и др., "Семантическая технология компонентного проектирования интеллектуальных решателей задач," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12 февраля 2011г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 265—286]

⇒ *аннотация**:
[В работе приводится описание открытой семантической технологии проектирования интеллектуальных решателей задач. Отдельное внимание уделяется методике проектирования решателей и операций, являющихся составными частями таких решателей.]
⇐ *библиографическая ссылка**:
- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*

Заливако С.С..СеманТКПИРЗ-2012ст
:= *стандартное библиографическое описание**:
[С. Заливако и Д. Шункевич, “Семантическая технология компонентного проектирования интеллектуальных решателей задач,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2012) : материалы II Междунар. науч.-техн. конф., Минск, 16–18 февр. 2012 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2012, с. 297—314]
⇒ *аннотация**:
[В работе приводится описание открытой семантической технологии проектирования интеллектуальных решателей задач. Отдельное внимание уделяется методике проектирования решателей и операций, являющихся составными частями таких решателей. Также в работе рассмотрено несколько примеров использования технологии при проектировании конкретных интеллектуальных систем по различным предметным областям.]
⇐ *библиографическая ссылка**:
- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*
- *§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем*

Затуливер Ю.С..ВопроПиМРЯСППсУПД-1981art
:= *стандартное библиографическое описание**:
[Ю. Затуливер и И. Медведев, “Вопросы построения и многопроцессорной реализации языка структурно-параллельного программирования с управлением потоками данных,” *Вопросы кибернетики. Многопроцессорные вычислительные системы с перестраиваемой структурой (Архитектура. Структура. Применения)*, с. 123—166, 1981]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Захаров В.П.ИнфорСДП-2002ст
:= *стандартное библиографическое описание**:
[В. Захаров, “Информационные системы (документальный поиск),” 2002]
⇒ *аннотация**:
[В монографии определяются основные понятия автоматизированного документального поиска. Особое внимание уделяется лингвистическому обеспечению информационно-поисковых систем. Также описываются особенности документального поиска в сети Интернет. Обобщен предшествующий путь развития автоматизированных информационно-поисковых систем и предлагается его периодизация. Дается типология и краткая характеристика современных систем. Делается вывод, что будущее информационного поиска находится прежде всего на путях интеллектуализации поисковых систем. Для специалистов по информационному поиску, студентов и аспирантов, специализирующихся в области прикладной лингвистики, информационных систем и автоматизированных систем обработки текста.]
⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Захарьев В.А..Подхо кУРНнОСАА-2018ст
:= *стандартное библиографическое описание**:
[В. А. Захарьев, И. С. Азаров и К. В. Русецкий, “Подход к устранению речевых неоднозначностей на основе семантико-акустического анализа,” в *Открытые семантические технологии проектирования интеллектуальных систем*, В. В. Голенков, ред., Минск : БГУИР, 2018, с. 211—222]
⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*

Зенг В.А..ОценкКППИНП-2019ст

:= *стандартное библиографическое описание**:
[В. А. Зенг, “Оценка качества проектирования пользовательских интерфейсов нового поколения,” *Известия Тульского государственного университета. Технические науки*, № 12, с. 404—410, 2019]

⇒ *аннотация**:
[В статье приведены различные современные модели для оценки качества интерфейсов]

⇐ *библиографическая ссылка**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Зимин И.В.Реком пПСиСЭОР-2016кн

:= *стандартное библиографическое описание**:
[И. В. Зимин, *Рекомендации по подготовке содержания и структурирования электронного обучающего ресурса*. Национальный Открытый Университет "ИНТУИТ", 2016, с. 102]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Золотов Е.В..РасшиСАД-1982кн

:= *стандартное библиографическое описание**:
[Е. Золотов и И. Кузнецов, *Расширяющиеся системы активного диалога*. Наука, 1982, с. 309]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Ивакин Я.А.МетодИПГСнОО-2009дс

:= *стандартное библиографическое описание**:
[Я. Ивакин, “Методы интеллектуализации промышленных геоинформационных систем на основе онтологий,” 371 л., дис. ... д-ра техн. наук : 05.13.06, Санкт-Петербург, 2009]

⇒ *аннотация**:
[В монографии рассматриваются актуальные вопросы интеллектуализации геоинформационных систем (внедрения методов и средств искусственного интеллекта в их состав), на примере ГИС, ориентированных на диспетчеризацию и обеспечение безопасности пространственных процессов. Основной акцент поставлен на технологиях использования онтологий для инженерии знаний - структурировании и использовании знаний в интересах поддержки принятия решений операторами диспетчерских пунктов управления пространственными процессами. В работе описана технологически замкнутая совокупность методов интеграции экспертных и геоинформационных систем.Так же показан практический эффект от использования результатов этой интеграции. Монография может быть полезна как для исследователей, так и для разработчиков интеллектуальных информационных систем, а так же для преподавателей ВУЗов и специалистов, интересующихся данной проблематикой.]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.2. Систематизация задач, решаемых интеллектуальными геоинформационными системами*

Иванюк Д.С.НейроПОУ-2014ст

:= *стандартное библиографическое описание**:
[Д. С. Иванюк, “Нейро-пид-регулятор ПОУ,” *Вестник Брестского государственного технического университета. Серия: Физика, математика, информатика.*, № 5, с. 35—40, 2014]

⇒ *аннотация**:
[The neuro-PID controller for the pasteurizer was developed. It consists of two parts: the conventional PID (proportional plus integral plus derivative controller) and the neural network, which is based on the multilayer perceptron structure. The outputs of the neural network are proportional (P), integral (I), and derivative (D) gains. The simulation and experimental results show the effectiveness of the proposed approach.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*
- *Подпункт 7.7.1.2.9 Примеры реализации систем адаптации управления*

Ивашенко В.П.ИсслеПРПРСТМО-2020ст

≔ *стандартное библиографическое описание**:

[В. П. Ивашенко, "Исследование производительности реализации параллельной редукционной схемы теоретико-множественных операций," в *Информационные технологии и системы 2020 (ИТС 2020) = Information Technologies and Systems 2020 (ITS 2020) : материалы международной научной конференции, Минск, 18 ноября 2020 г.*, Белорусский государственный университет информатики и радиоэлектроники; редкол. : Л. Ю. Шилин [и др.]., Минск, 2020, с. 76—77]

⇒ *аннотация**:

[В работе приведены результаты исследований для обработки информации на уровне управления знаниями, представленными средствами модели унифицированного семантического представления знаний, особенностью которой является наличие базовой теоретико-множественной семантики представленных знаний.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.Модел иАИЗнООСС-2014дс

≔ *стандартное библиографическое описание**:

[В. П. Ивашенко, "Модели и алгоритмы интеграции знаний на основе однородных семантических сетей (дисс. на соисвание степени канд. техн. наук: 05.13.17)," дис. ... канд. техн. наук : 05.13.17, Белорус. гос. ун-т информатики и радиоэлектроники, Минск, 2014]

⇒ *аннотация**:

[В диссертации исследуются онтологии и фрагменты баз знаний, представленные однородными семантическими сетями специального вида, и отношения их интеграции, возникающие на этапах создания и функционирования соответствующих интеллектуальных систем. Целью работы является разработка моделей, алгоритмов и программных средств, обеспечивающих точность и относительную полноту результатов интеграции такого рода знаний.]

⇐ *библиографическая ссылка**:

- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Ивашенко В.П.МоделРЗвИС-2020кн

≔ *стандартное библиографическое описание**:

[В. П. Ивашенко, *Модели решения задач в интеллектуальных системах. В 2 ч. Ч. 1 : Формальные модели обработки информации и параллельные модели решения задач : учеб.-метод. пособие*. Минск : БГУИР, 2020]

⇒ *аннотация**:

[Излагается учебный материал, к которому прилагается список вопросов и задания для лабораторных работ.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.ОнтолМПВОСиЯвПОЗ-2017ст

≔ *стандартное библиографическое описание**:

[В. П. Ивашенко, "Онтологическая модель пространственно-временных отношений событий и явлений в процессах обработки знаний," *Вестник Брестского государственного технического университета*, т. 107, № 5, с. 13—17, 2017]

⇐ *библиографическая ссылка**:

- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Ивашенко В.П.ОпераУМДвЛАП-2016ст

≔ *стандартное библиографическое описание**:

[С. С. В.П. Ивашенко, "Операции управления массивами данных в линейно адресуемой памяти," *Доклады БГУИР*, т. 6, с. 86—93, окт. 2016]

⇒ *аннотация**:

[Рассматривается задача управления массивами данных в конечной линейно адресуемой памяти. Как часть решения этой задачи приводится способ реализации операций управления массивами данных, основанный на использовании стратегий распределения участков памяти на уровне управления устройствами исполнения систем, основанных на знаниях. Приводятся результаты тестирования различных стратегий.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.ПредсССиАиОиСОнВСсМП-2015ст

:= *стандартное библиографическое описание**:
[В. и. д. Ивашенко, “Представление семантических сетей и алгоритмы их организации и семантической обработки на вычислительных системах с массовым параллелизмом,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 133—140]

⇒ *аннотация**:
[В работе приводится описание способов и некоторых алгоритмов организации хранения и представления семантических сетей в архитектуре проблемно-ориентированного семантического процессора. Рассмотрена возможность применения разработанных алгоритмов для решения прикладных задач.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.РаспрПвНЛАП-2019ст

:= *стандартное библиографическое описание**:
[В. П. Ивашенко, “Распределение памяти в неограниченном линейном адресном пространстве,” в *Информационные технологии и системы 2019 (ИТС 2019) = Information Technologies and Systems 2019 (ITS 2019) : материалы международной научной конференции, Минск, 30 октября 2019 г.*, Белорусский государственный университет информатики и радиоэлектроники; редкол. : Л. Ю. Шилин [и др.]., Минск, 2019, с. 110—111]

⇒ *аннотация**:
[В статье рассматривается подход к построению и реализация алгоритмов распределения неограниченно линейно адресуемой памяти для универсальных моделей решения задач.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.Модел иАИЗнООСС-2014дс

:= *стандартное библиографическое описание**:
[В. П. Ивашенко, “Модели и алгоритмы интеграции знаний на основе однородных семантических сетей,” автореф. дис. ... канд. техн. наук : 05.13.17, Белорус. гос. ун-т информатики и радиоэлектроники, Минск, 2014]

⇒ *аннотация**:
[В диссертации исследуются онтологии и фрагменты баз знаний, представленные однородными семантическими сетями специального вида, и отношения их интеграции, возникающие на этапах создания и функционирования соответствующих интеллектуальных систем. Целью работы является разработка моделей, алгоритмов и программных средств, обеспечивающих точность и относительную полноту результатов интеграции такого рода знаний.]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Ивашенко В.П.Модел иАИЗнООСС-2015ст

:= *стандартное библиографическое описание**:
[В. П. Ивашенко, “Модели и алгоритмы интеграции знаний на основе однородных семантических сетей,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 111—132]

⇒ *аннотация**:
[В работе рассматриваются модели и алгоритмы интеграции знаний, представленных в унифицированном виде однородными семантическими сетями, имеющими теоретико-множественную интерпретацию, учитывающие неполноту, нечеткость, неопределенность представленных знаний и их изменение со временем.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.2. Принципы, лежащие в основе, и структура предлагаемой Программной платформы ostis-систем*

Ивашенко В.П.ПринцПНиПРО-2016ст

:= *стандартное библиографическое описание**:

[В. П. Ивашенко и М. М. Татур, "Принципы платформенной независимости и платформенной реализации OSTIS," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 145—150]

⇒ *аннотация**:

[В работе рассматривается подход к спецификации платформ, их сравнению и принципы их реализации, рассматриваются виды платформенной независимости и даются схемы построения платформенно независимых компонентов интеллектуальных систем, использующих в качестве языка представления знаний однородные семантические сети с теоретико-множественной интерпретацией.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ивашенко В.П.СеманТКПБЗ-2011ст

:= *стандартное библиографическое описание**:

[В. П. Ивашенко, "Семантическая технология компонентного проектирования баз знаний," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. научн.-техн. конф., Минск, 10-12 февраля 2011г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 129—146]

⇒ *аннотация**:

[В работе рассматриваются составляющие технологии компонентного проектирования баз знаний, использующей для представления знаний однородные сеемантические сети с теоретико-множественной семантикой и ориентацией на расширение контингента разработчиков и сокращение сроков проектирования.]

⇐ *библиографическая ссылка**:

- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.6. Многократно используемые компоненты баз знаний ostis-систем*

Ивашенко В.П.УнифиПиИЗ-2013ст

:= *стандартное библиографическое описание**:

[В. П. Ивашенко, "Унифицированное представление и интеграция знаний," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 171—184]

⇒ *аннотация**:

[В работе рассматриваются составляющие и применение средств технологии компонентного проектирования баз знаний в виде однородных семантических сети с теоретико-множественной семантикой для решения задач отладки и интеграции баз знаний.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Игнатущенко В.В.кПостаЗПЭВСнОАМОИ-1981ст

:= *стандартное библиографическое описание**:

[В. Игнатущенко, "К постановке задачи повышения эффективности вычислительных систем на основе ассоциативных методов обработки информации," *Вопросы кибернетики. Многопроцессорные вычислительные системы с перестраиваемой структурой (Архитектура. Структура. Применения)*, с. 14—21, 1981]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

ИскусИММ-1990кн

:= *стандартное библиографическое описание**:

[*Искусственный интеллект: Модели и методы*. М.: Радио и связь, 1990]

⇐ *библиографическая ссылка**:

- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

Исобоев Ш.И..ИнтелСМБСБСнОМО-2022ст

:= *стандартное библиографическое описание**:

[Ш. И. Исобоев, Д. А. Везарко и А. С. Чечельницкий, “Интеллектуальная система мониторинга безопасности сети беспроводной связи на основе машинного обучения,” *Экономика и качество систем связи*, № 1(23), с. 44—48, 2022]

⇒ *аннотация**:

[Для решения проблем традиционных систем мониторинга уязвимостей безопасности сетей беспроводной связи, таких как низкая точность мониторинга и трудоемкость, в статье предлагается интеллектуальная система мониторинга уязвимостей на основе машинного обучения. Представлен алгоритм машинного обучения, который сочетается с программным обеспечением интеллектуальной системы мониторинга для реализации интеллектуального мониторинга уязвимостей безопасности сети беспроводной связи. Результаты эксперимента показывают, что интеллектуальная система мониторинга уязвимостей сети беспроводной связи, основанная на машинном обучении, может эффективно повысить точность мониторинга системы и эффективность мониторинга уязвимостей сети беспроводной связи.]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*

Казарновский В.А..РазвиПкВСУД-2019ст

:= *стандартное библиографическое описание**:

[А. М. Казарновский В.А., “Развитие подходов к внедрению системы «умный дом» в рамках инвестиционно-строительных проектов малоэтажного строительства,” *Московский экономический журнал*, № 6, с. 277—286, 2019]

⇒ *аннотация**:

[Инвестор является главным участником строительного процесса, особенно это касается инновационных проектов – таких, как, например, внедрение систем «умный дом», так как они зачастую требуют дорогостоящих капиталовложений. Цель данной статьи — рассмотреть подходы к внедрению системы «Умный дом» в рамках инвестиционно-строительных проектов малоэтажного строительства. Кратко рассмотреть историю развития такой системы Задачи исследования: рассмотреть инновационную для 21 века систему «Умный дом». Описать ее основные свойства, особенности и преимущества, способах монтажа, и управлении. Гипотеза исследования сводится к то, что в настоящее время без инвестирования невозможно осуществление любого строительного проекта. Методы исследования. Решение поставленных в работе задач осуществлялось на основе применения общенаучных методов исследования в рамках сравнительного, логического и статистического анализа. Результаты данного исследования. Составление технико-экономического обоснования внедрения системы «умный дом» и использование сравнительных методов инвестиционных проектов позволит стабилизировать инвестиции в строительной сфере, придать необходимый стимул к росту инвестиционной активности и достижению лучших конечных результатов.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Калиниченко Л.А..МашинБДиЗ-1990кн

:= *стандартное библиографическое описание**:

[Л. А. Калиниченко и В. М. Рывкин, *Машины баз данных и знаний*. М.: Наука, 1990]

⇒ *аннотация**:

[Рассмотрены вопросы построения машин баз данных и знаний. В книге дан обзор и сравнительный анализ известных проектов и промышленных образцов МБДЗ, приведены методы формирования унифицированных представлений данных и знаний в МБДЗ, определены требования к их интерфейсам. Проанализированы структурные методы повышения производительности МБДЗ, методы организации данных и алгоритмы реализации операций в МБДЗ, а также способы аппаратурной поддержки основных функций управления базами данных и знаний]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Касьянов В.Н..Графы вПОВиП-2003кн

:= *стандартное библиографическое описание**:
[В. Н. Касьянов и В. А. Евстигнеев, “Графы в программировании: обработка, визуализация и применение,” 2003]
⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем*

Кахро М.В..ИнстрСПЕСЭВМ-1988кн
:= *стандартное библиографическое описание**:
[М. В. Кахро, А. П. Калья и Э. Х. Тыугу, *Инструментальная система программирования ЕС ЭВМ (ПРИЗ)*. М.: Финансы и статистика, 1988]
⇒ *аннотация**:
[В книге описывается оригинальная отечественная разработка – система программирования ПРИЗ. Ориентированное на создание всевозможных прикладных программ, снабжённых проблемно-ориентированными языками высокого уровня, система ПРИЗ обеспечивает существенное повышение производительности труда программистов.]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Кикоть А.В.СравнМиМИО-2020ст
:= *стандартное библиографическое описание**:
[А. В. Кикоть, “Сравнение моделей и методов интеллектуальной обработки геопространственных данных на основе семантической технологии,” *Интеллектуальные технологии на транспорте*, № 2, с. 28—34, 2020]
⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Климанская Е.В.СовреПИАО-2014ст
:= *стандартное библиографическое описание**:
[Е. В. Климанская, “Современные платформы интеллектуальной аналитической обработки информации: графовые базы данных,” *Наука вчера, сегодня, завтра*, с. 9—16, 2014]
⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Клини С.К.МатемЛ-1973кн
:= *стандартное библиографическое описание**:
[С. К. Клини, *Математическая логика*. Москва : Мир, 1973, перевод с англ 1967]
⇒ *аннотация**:
[Имя одного из крупнейших современных специалистов в области математической логики С.К. Клини знакомо советскому читателю по русскому переводу его фундаментального труда «Введение в метаматематику» (ИЛ, 1957), ставшего настольной книгой для всех, кто занимается математической логикой, рекурсивными функциями и основаниями математики. Новая его книга представляет собой существенно усовершенствованный, расширенный и приближенный к нуждам университетского преподавания вариант «чисто логической» части этой всемирно известной монографии. Тщательно продуманные иллюстративные упражнения помогают читателю усвоить излагаемый материал. Книга может быть использована как учбеное пособие по курсу математической логики в университетах и пединститутах; таким образом, она адресована прежде всео предполавателям, аспирантам и студентам. Она привлечёт также внимание всех занимающихся или интересующихся математической логикой.]
⇐ *библиографическая ссылка**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*

Колб Д.Г.WebOPCM-2011ст
:= *стандартное библиографическое описание**:
[Д. Г. Колб, “Web-ориентированная реализация семантических моделей интеллектуальных систем для систем дистанционного обучения,” в *Дистанционное обучение — образовательная среда XXI века : материалы VII Международной научно-методической конференции*, БГУИР, Минск, 2011, с. 258—260]
⇒ *аннотация**:

[Подход к построению семантически структурированных веб-сайтов разного уровня интеллекта для систем дистанционного обучения. Предлагаемый подход заключается в использовании однородных семантических сетей с базовой теоретико-множественной интерпретацией.]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Колб Д.Г.ПроблОМПнО-2013ст

:= *стандартное библиографическое описание**:

[Д. Г. Колб, “Проблемы обеспечения мультимедийного поиска на основе семантических сетей в системах дистанционного обучения,” в *Дистанционное обучение — образовательная среда XXI века: материалы VIII международной научно-методической конференции. (Минск, 5–6 декабря 2013 года)*, БГУИР, Минск, 2013, с. 188—189]

⇒ *аннотация**:

[Приводятся подходы к организации поиска мультимедийных данных с использованием баз знаний и методов искусственного интеллекта, которые можно использовать в системах дистанционного обучения. Перечислены основные проблемы существующих методов поиска мультимедийных данных, в основе которых лежат онтологии. Сформулированы требования к модели поиска, которая, по мнению автора, должна обеспечить решение ряда существующих проблем мультимедийного поиска.]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Колесников А.В.ГибриИСТиТ-2001кн

:= *стандартное библиографическое описание**:

[А. В. Колесников, *Гибридные интеллектуальные системы: Теория и технология разработки*, А. М. Яшин, ред. СПб.: Изд-во СПбГТУ, 2001, ISBN: 5-7422-0187-7]

⇒ *аннотация**:

[Монография содержит материалы исследований по гибридным интеллектуальным системам за период 1986-2000 гг. Предназначена для широкого круга читателей.]

⇐ *библиографическая ссылка**:

- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

Комарцова Л.Г..Нейро-2004кн

:= *стандартное библиографическое описание**:

[Л. Г. Комарцова и А. В. Максимов, *Нейрокомпьютеры. - 2-е изд.* (Информатика в техническом университете). Москва: МГТУ им. Баумана, 2004]

⇒ *аннотация**:

[В учебнике изложены вопросы современной теории нейрокомпьютеров. Приведен анализ различных архитектур вычислительных устройств с параллельной организацией работы. Рассмотрен биологический аналог параллельной организации обработки информации. Большое внимание уделено разновидностям построения формальных нейронов, технологии сетей и классическим методам их обучения, методам подготовки задач для решения на нейрокомпьютерах. Приведены оригинальные результаты применения нейронных сетей для решения систем дифференциальных уравнений и степенных рядов в конструировании нейросетевых алгоритмов; обучения нейронных сетей на базе генетического алгоритма и теории адаптивного резонанса. Представлены программные системы эмуляции нейронных сетей, разработанных в Калужском филиале МГТУ им. Н.Э.Баумана. Уделено внимание аппаратной реализации нейрокомпьютеров, в том числе и на отечественной элементной базе.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению*

Корончик Д.Н.РеалиХУСС-2013ст

:= *стандартное библиографическое описание**:

[Д. Н. Корончик, “Реализация хранилища унифицированных семантических сетей,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы Междунар.*

науч.-техн. конф., Минск, 21–23 февр. 2013 г., Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 125—129]

⇒ *аннотация**:

[В работе описана одна из возможных реализаций хранилища унифицированных семантических сетей. Приводится описание принципов хранения элементов семантической сети и связей между этими элементами.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.2. Принципы, лежащие в основе, и структура предлагаемой Программной платформы ostis-систем*

Корончик Д.Н.СеманММПИ

:= *стандартное библиографическое описание**:

[Д. Н. Корончик, “Семантические модели мультимодальных пользовательских интерфейсов и семантическая технология их проектирования,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2012) : материалы Междунар. науч.-техн. конф., Минск, 16–18 февр. 2012 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 339—346]

⇒ *аннотация**:

[В работе описана технология проектирования пользовательских интерфейсов интеллектуальных систем. Приводится описание семантической модели пользовательских интерфейсов, которая является основной описываемой технологии. Предложенная в данной работе технология позволяет проектировать мультимодальные пользовательские интерфейсы на основе готовых совместимых компонентов]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Корончик Д.Н.СеманТКПП-2011ст

:= *стандартное библиографическое описание**:

[Д. Н. Корончик, “Семантическая технология компонентного проектирования пользовательских интерфейсов интеллектуальных систем,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы Междунар. науч.-техн. конф., Минск, 10–12 февр. 2011 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 293—304]

⇒ *аннотация**:

[В работе приводится описание технологии проектирования пользовательских интерфейсов для интеллектуальных систем, основанных на семантических сетях]

⇐ *библиографическая ссылка**:

- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем*

Корончик Д.Н.УнифиСМПИ-2013ст

:= *стандартное библиографическое описание**:

[Д. Н. Корончик, “Унифицированные семантические модели пользовательских интерфейсов интеллектуальных систем и технология их компонентного проектирования,” в *Открытые семантические технологии проектирования интеллектуальных систем*, В. Голенков, ред., БГУИР, Минск, 2013, с. 403—406]

⇒ *аннотация**:

[В работе описывается пример использования и реализации модели пользовательского интерфейса интеллектуальной системы. Поэтапно описывается процесс получения запросов от пользователя, и вывод ответов на них.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Корончик Д.Н.ПользИИМП-2014ст

:= *стандартное библиографическое описание**:
[Д. Н. Корончик, “Пользовательский интерфейс интеллектуальной метасистемы поддержки проектирования интеллектуальных систем,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2014) : материалы Междунар. науч.-техн. конф., Минск, 20–22 февр. 2014 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2014, с. 79—82]

⇒ *аннотация**:
[В работе описывается реализованный пользовательский интерфейс для интеллектуальной метасистемы поддержки проектирования интеллектуальных систем, которая располагается по адресу http://ims.ostis.net. В рамках статьи рассмотрены команды доступные пользователю в рамках ПИ, компоненты, которые реализованы для его функционирования и основные идеи, которые лежат в его основе]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.2. Многократно используемые компоненты интерфейсов ostis-систем*

Корончик Д.Н.РеалиПWСУ-2015ст

:= *стандартное библиографическое описание**:
[Д. Н. Корончик, “Реализация платформы для web-ориентированных систем, управляемых знаниями,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 89—92]

⇒ *аннотация**:
[В работе приводится описание технической реализации платформы для web-ориентированных систем, управляемых знаниями. Описаны основные принципы, которые лежат в основе платформы: ориентация на коллективную разработку базы знаний, возможность отображения (редактирования) базы знаний на различных внешних языках и их отображение в режиме реального времени.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем*

Котов В.Е..АсинхВПнОП-1966ст

:= *стандартное библиографическое описание**:
[В. Е. Котов и А. С. Нариньяни, “Асинхронные вычислительные процессы над общей памятью,” *Кибернетика*, № 3, с. 64—71, 1966]

⇒ *аннотация**:
[В статье предлагается асинхронная модель вычислений, отличительной особенностью которой является управление на основе строгого частичного порядка.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*

Кохонен Т.АссоцЗУ-1982кн

:= *стандартное библиографическое описание**:
[Т. Кохонен, *Ассоциативные запоминающие устройства: Пер. с англ.* Мир, 1982, с. 384]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Кохонен Т.АссоцП-1980кн

:= *стандартное библиографическое описание**:

[Т. Кохонен, *Ассоциативная память: Пер. с англ.* Мир, 1980, с. 240]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Крючков А.Н..ИнтелТвГС-2006кн

:= *стандартное библиографическое описание**:

[А. Крючков и др., "Интеллектуальные технологии в геоинформационных системах," 2006]

⇒ *аннотация**:

[В учебном пособии рассматриваются информационные технологии, применяемые при разработке интеллектуальных компонент геоинформационных систем (ГИС). Дается понятие ГИС, рассматриваются способы организации данных в ГИС, методы статистического анализа геоданных. Особое место уделено применению средств искусственного и интеллекта в ГИС, приведена типология задач интеллектуализации ГИС.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.1. Требования, предъявляемые к интеллектуальным геоинформационным системам нового поколения*

Кузнецов К.А..ПубликДоООП-2012ст

:= *стандартное библиографическое описание**:

[К. А. Кузнецов, В. А. Серебряков и К. Б. Теймуразов, "Публикация данных об особо охраняемых природных территориях в пространстве," в *Интернет и современное общество» (IMS-2012) : Труды XV Всероссийской объединенной конференции, Санкт-Петербург, 10 – 12 октября 2012г.*, Санкт-Петербург, 2012, с. 62—72]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Кузьмицкий В.М.ПринцПГПКОнРЗИИ-2000дс

:= *стандартное библиографическое описание**:

[В. М. Кузьмицкий, "Принципы построения графодинамического параллельного компьютера, ориентированного на решение задач искусственного интеллекта," 236 л., дис. ... канд. техн. наук : 05.13.11, Минск, 2000]

⇒ *аннотация**:

[Работа содержит описание принципов построения графодинамического параллельного компьютера, ориентированного на решение задач искуственного интеллекта]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*
- *§ 6.2.3. Общие принципы, лежащие в основе ассоциативных семантических компьютеров для ostis-систем*
- *§ 6.2.4. Архитектура ассоциативных семантических компьютеров для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*
- *Пункт 6.2.4.2. Вариант крупнозернистой архитектуры ассоциативных семантических компьютеров*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*

Кулик Б.А.ЛогикЕР-2001кн

:= *стандартное библиографическое описание**:

[Б. А. Кулик, *Логика естественных рассуждений*. СПб.: Нев. диалект, 2001]

⇒ *аннотация**:

[Книга по индуктивному выводу]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Курбатов С.С..ПрогрОдАРЗ-2016ст

:= *стандартное библиографическое описание**:

[С. С. Курбатов, А. П. Лобзин и Г. К. Хахалин, “Программное обеспечение для автоматического решения задач по планиметрии,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 159—164]

⇒ *аннотация**:

[В статье рассмотрена система автоматического решения задач по планиметрии с использованием онтологии и естественно-языкового интерфейса. В рамках системы разработано программное обеспечение, использующее онтологию при поиске решения задачи, сформулированной на предметно-ориентированном естественном языке. Для найденного решения разработана визуализация на предметном и онтологическом уровнях. Программное обеспечение включает макросы Word для работы с графикой и решатель, реализованный в программной среде онтологии]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Лазуркин Д.А.ТехноКППОнОСС-2011ст

:= *стандартное библиографическое описание**:

[Д. А. Лазуркин, “Технология компонентного проектирования программ, ориентированных на обработку семантических сетей,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2011) : материалы междунар. науч.-техн. конф., Минск, 10–12 февр. 2011 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 167—176]

⇒ *аннотация**:

[В работе приводится описание основных разделов технологии проектирования программ, ориентированных на обработку семантических сетей]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Летичевский А.А..ИнсерМ-2012ст

:= *стандартное библиографическое описание**:

[А. Летичевский, “Инсерционное моделирование,” *Управляющие системы и машины*, № 6, с. 3—14, 2012]

⇒ *аннотация**:

[Работа посвящена инсерционному программированию.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Летичевский А.А..ИнсерП-2003ст

:= *стандартное библиографическое описание**:

[А. Летичевский и др., “Инсерционное программирование,” *Кибернетика и системный анализ*, № 1, с. 19—32, 2003]

⇒ *аннотация**:

[Работа посвящена инсерционному программированию.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.3. Принципы, лежащие в основе многоагентных моделей решателей задач интеллектуальных компьютерных систем нового поколения*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Ли В..ОнтолМГВдИО-2019ст

:= *стандартное библиографическое описание**:

[Л. Вэньцзу и Ц. Лунвэй, “Онтологическая модель генерации вопросов для интеллектуальных обучающих систем,” в *Дистанционное обучение – образовательная среда XXI века : материалы XI Международной научно-методической конференции, Минск, 12-13 декабря 2019 г.*, Минск: БГУИР, 2019, с. 184—185]

⇒ *аннотация**:
[В этой статье представлены некоторые методы автоматической генерации вопросов в системе обучения искусственному интеллекту.]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Любарский Ю.Я.ИнтелИС-1990кн

:= *стандартное библиографическое описание**:
[Ю. Я. Любарский, *Интеллектуальные информационные системы*. Москва, 1990]

⇒ *аннотация**:
[Излагается методика построения интеллектуальных информационных систем, позволяющих решать многие практические важные задачи. Такие системы, используя декларативное представление знаний о проблемной области и процедурное представление умений решения задач из этой области, обеспечивают взаимодействие с различными группами пользователей на ограниченном естественном языке. Приводятся примеры их конкретных применений.]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Майерс Г.АрхитСЭВМ-1985кн

:= *стандартное библиографическое описание**:
[М. Г., *Архитектура современных ЭВМ: В 2-х кн. Кн. 2. / Пер. с англ.* Мир, 1985, с. 310]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Мартин Дж.ОрганБДвВС-1980кн

:= *стандартное библиографическое описание**:
[Д. Мартин, *Организация баз данных в вычислительных системах: Пер. с англ.* Мир, 1980, с. 120]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Мартынов В.В.вЦентрСЧ-2009кн

:= *стандартное библиографическое описание**:
[В. В. Мартынов, *В центре сознания человека*. Минск: БГУ, 2009]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Мартынов В.В.СемиоОИ-1974кн

:= *стандартное библиографическое описание**:
[В. Мартынов, *Семиологические основы информатики*. Минск : Наука и техника: АН БССР. Ин-т языкознания им. Якуба Коласа, 1974]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Мартынов В.В.Язык вПиВкПГС-2004кн

:= *стандартное библиографическое описание**:
[В. В. Мартынов, *Язык в пространстве и времени: к проблеме глоттогенеза славян*. Москва: УРСС, 2004]

⇒ *аннотация**:
[В книге представлены методы пространственно-временной стратификации носителей языков доисторических периодов по данным сравнительно-исторического языкознания.]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*

- *§ 2.2.3. Смысловое пространство ostis-систем*

Марчук Г.И.МодулАРС-1978кн

:= *стандартное библиографическое описание**:
[М. Г. [др.], *Модульная асинхронная развивающаяся система: В 2-х ч.* Акад. наук. Сиб. отд-ние. Вычислительный центр, 1978]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Массель Л.В..СеманТнОИОК-2013ст

:= *стандартное библиографическое описание**:
[Л. В. Массель и А. Г. Массель, "Семантические технологии на основе интеграции онтологического, когнитивного и событийного моделирования," в *Открытые семантические технологии проектированияинтеллектуальных систем = Open Semantic Technologies for IntelligentSystems (OSTIS-2013): материалы III Междунар. научн.-техн. конф.(Минск, 21-23 февраля 2013г.)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2013, с. 247—250]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Мельчук И.А.КакНМЛ-1998ст

:= *стандартное библиографическое описание**:
[И. А. Мельчук, "Как начиналась математическая лингвистика," 1998, с. 358—370]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Мельчук И.А.ОпытТЛМСТ-1999кн

:= *стандартное библиографическое описание**:
[И. А. Мельчук, *Опыт теории лингвистических моделей "Смысл – текст"*. М.: Школа "Языки русской культуры", 1999]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

МетасOSTIS-2022эл

:= *стандартное библиографическое описание**:
[*Метасистема OSTIS [Электронный ресурс]*, Режим доступа: `http://ims.ostis.net`. — Дата доступа: 04.12.2022]

⇒ *аннотация**:
[Официальный сайт Метасистемы OSTIS]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Библиография OSTIS*
- *Глава 7.2. Метасистема OSTIS*
- *§ 7.2.1. Структура, назначение, особенности и достоинства Метасистемы OSTIS*
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*
- *Пункт 7.5.3.4. Семантическая модель решателя задач подсистемы контроля знаний*
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.4.2. Денотационная семантика SCn-кода*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*

Москаленко Ф.М..ТехноРРЗИСдОППнОРРОИ-2016ст

:= *стандартное библиографическое описание**:
[В. Ф.М.Москаленко, *Технология разработки решателей задач интеллектуальных сервисов для облачной платформы IACPAAS на основе расширяемого редактора орграфов информации*. 2016, с. 39—44]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Москин Н.Д.оПредсЗсПССвИ-2011ст

:= *стандартное библиографическое описание**:
[Н. Д. Москин, “О представлении знаний с помощью семантических сетей в интеллектуальной системе по исследованию фольклорных текстов,” в *Открытые семантические технологии проектированияинтеллектуальных систем = Open Semantic Technologies for IntelligentSystems (OSTIS-2011): материалы Междунар. научн.-техн. конф. (Минск,10-12 февраля 2011 г.)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2011, с. 115—124]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Муромцев Д.И.Модел иМИЭОвК-2020ст

:= *стандартное библиографическое описание**:
[Д. И. Муромцев, “Модели и методы индивидуализации электронного обучения в контексте онтологического подхода,” *Онтология проектирования*, т. 10, № 1 (35), с. 34—49, 2020. DOI: 10.18287/2223-9537-2020-10-1-34-49]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Налимов В.В..РеальНВМБ-1995кн

:= *стандартное библиографическое описание**:
[В. Налимов и Ж. Дрогалина, *Реальность нереального. Вероятностная модель бессознательного*. Москва: Мир идей, 1995]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Налимов В.В.ВероямЯоСЕиИЯ-1979кн

:= *стандартное библиографическое описание**:
[В. Налимов, *Вероятностная модель языка. О соотношении естественных и искусственных языков*. Москва: Наука, 1979]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Налимов В.В.СпонтСВТСиСАЛ-1989кн

:= *стандартное библиографическое описание**:
[В. Налимов, *Спонтанность сознания. Вероятностная теория смыслов и смысловая архитектоника личности*. Москва: Прометей, 1989]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Нариньяни А.С.нФактоКВ-2004ст

:= *стандартное библиографическое описание**:
[А. Нариньяни, “НЕ-факторы: краткое введение,” *Новости искусственного интеллекта*, № 2, с. 52—63, 2004]

⇒ *аннотация**:
[В статье рассмотрено понятие НЕ-факторов, дается их классификация и описание.]

⇐ *библиографическая ссылка**:
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Нариньяни А.С.нФактоНиНРиВ-2000ст

:= *стандартное библиографическое описание**:
[А. С. Нариньяни, “НЕ-ФАКТОРЫ: Неточность и Недоопределенность — различие и взаимосвязь,” *Изв. РАН, Теор. и сист. упр*, № 5, с. 44—56, 2000]

⇒ *аннотация**:
[В статье делается попытка расширить сферу исследования НЕ-факторов, которое пока не выходило за рамки недоопределенности.]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.1. Базовая денотационная семантика SC-кода*

Наумов А.Н..СистеУБДиЗ-1991кн

:= *стандартное библиографическое описание**:
[А. Н. Наумов, А. М. Вендров и В. К. Иванов, *Системы управления базами данными и знаний*. М.: Финансы и статистика, 1991]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

НейроПА-2023эл

:= *стандартное библиографическое описание**:
[*Нейроморфный процессор “Алтай”*, Russian, Mode of access: `https://motivnt.ru/neurochip-altai/`. — Date of access: 29.03.2023]

⇒ *аннотация**:
[Официальный сайт компании-разработчика нейроморфного процессора “Алтай”]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Голенков В.В.ред.ОткрыСТПИ-2014ст

:= *стандартное библиографическое описание**:
[“О Викторе Владимировиче Мартынове,” в *Открытые семантические технологии проектирования интеллектуальныхсистем = Open Semantic Technologies for Intelligent Systems (OSTIS-2014) :материалы IV междунар. науч.-техн. конф. (Минск, 20-22 февраля 2014 года)*, В. В. Голенков, Л. С. Глоба и др., ред., Минск: БГУИР, февр. 2014, с. 25—28]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

ЧтоТОД-2023эл

:= *стандартное библиографическое описание**:
[*Что такое озеро данных?* 2022. url: `https://www.oracle.com/cis/big-data/what-is-data-lake/`]

⇒ *аннотация**:
[Описание технологии “Озеро данных” на сайте Oracle]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами*

Озкарахан Э.МашинБДиУБД-1989кн

:= *стандартное библиографическое описание**:
[Э. Озкарахан, *Машины баз данных и управление базами данных*. Мир, 1989, с. 696]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Орехова Д.А.Разра иИМДП-2013дс

:= *стандартное библиографическое описание**:

[Д. Орехова, “Разработка и исследование метода динамического представления пространственных данных в геоинформационных системах,” дис. ... к-та техн. наук : 05.25.05, Таганрог, 2013]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Орлов С.А.Теори иПЯПУ-2021кн

:= *стандартное библиографическое описание**:

[С. А. Орлов, *Теория и практика языков программирования. Учебник для вузов. Стандарт 3-го поколения*. "Издательский дом Питер", 2021]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Пункт 3.2.4.2. Денотационная семантика программ в ostis-системах*
- *§ 3.2.6. Комплекс свойств, определяющих эффективность программ в ostis-системах*

Осин А.В.Мульт вОКИ-2004кн

:= *стандартное библиографическое описание**:

[А. В. Осин, *Мультимедиа в образовании: контекст информатизации*. М.: Агентство «Издательский сервис», 2004, с. 420]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Особе иПВМ-эл

:= *стандартное библиографическое описание**:

[*Особенности и преимущества версий MATLAB [Электронный ресурс]*, Режим доступа: `https://ru.education-wiki.com/2303364-MATLAB-version`. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Остроух А.В.ИнтелСМ-2020кн

:= *стандартное библиографическое описание**:

[А. В. Остроух, *Интеллектуальные системы: монография*. Красноярск: Научно-инновационный центр, 2020]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*

Палагин А.В..ПроблТиРИ-2013ст

:= *стандартное библиографическое описание**:

[А. Палагин, “Проблемы трансдисциплинарности и роль информатики,” *Кибернетика и системный анализ*, т. 49, № 5, с. 3—13, 2013]

⇐ *библиографическая ссылка**:

- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*

Петров С.В.ГрафоГиА-1978ст

:= *стандартное библиографическое описание**:

[С. В. Петров, “Графовые грамматики и автоматы (обзор),” *Автоматика и телемеханика*, № 7, с. 116—136, 1978]

⇒ *аннотация**:

[Рассматриваются различные модели графовых грамматик и соответствующие им модели автоматов на графах. Описываются области применения таких грамматик, приводится сводка результатов о реберных грамматиках, порождающих гиперграфы, и сводка результатов, относящихся к вершинным графовым грамматикам, грамматикам деревьев и соответствующим автоматным моделям.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем*
- *Пункт 3.2.4.1. Синтаксис программ в ostis-системах*

Пивоварчик О.В.КомпоАИСК-2015ст

:= *стандартное библиографическое описание**:

[О. В. Пивоварчик, “Компонентная архитектура интеллектуальной системы консультационного обслуживания и обучения разработчиков программ,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 141—148]

⇒ *аннотация**:

[В данной статье обоснована необходимость создания интеллектуальных компьютерных средств обучения в области программирования, имеющих открытую многокомпонентную архитектуру. В статье представлена общая компонентная архитектура системы и семантические модели каждого ее компонента.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.2. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Пивоварчик О.В.СеманМПИС-2016ст

:= *стандартное библиографическое описание**:

[О. В. Пивоварчик, “Семантическая модель пользователя интеллектуальной справочной системы по проектированию программ,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 521—526]

⇒ *аннотация**:

[В работе проведен анализ существующих моделей пользователя и перечислены способы адаптации компьютерных систем к пользователю. Предложена семантическая модель пользователя справочной системы по проектированию программ, ориентированных на обработку баз знаний.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*

Пивоварчик.О.В.СемантЯОТРПООЗ-2013ст

:= *стандартное библиографическое описание**:

[О. В. Пивоварчик, “Семантические языки для описания технологии разработки программ, ориентированных на обработку знаний,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 205—216]

⇒ *аннотация**:

[В работе описаны семантические языки представления знаний, используемые для создания баз знаний интеллектуальных help-систем для разработчиков программ, ориентированных на обработку знаний]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*

Подколзин А.С.КомпьМЛПА-2008кн

:= *стандартное библиографическое описание**:

[А. С. Подколзин, *Компьютерное моделирование логических процессов: архитектура решателя и язык решателя задач*. М.: Физматлит, 2008]

⇒ *аннотация**:

[В книге представлено описание разработанного автором пакета прикладных программ "Логическая система "Искра, обобщающего многолетний опыт компьютерного моделирования логических процессов, в результате которого возникла развитая технология обучения "решателя". В основном моделировались процессы решения математических задач]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Пойа Д.Матем иПР-1975кн

:= *стандартное библиографическое описание**:

[Д. Пойа, *Математика и правдоподобные рассуждения*, пер. с англ. И. А. Ванштейна ; под ред. С. А. Яновской, ред. М.: Наука, 1975]

⇒ *аннотация**:

[В книге описывается какими путями добываются новые факты в математике, с какой степенью доверия следует относиться к той или иной математической гипотезе]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Попов А.Ю.ПримеГВСсНК-2019ст

:= *стандартное библиографическое описание**:

[А. Ю. Попов, “Применение гетерогенной вычислительной системы с набором команд дискретной математики для решения задач на графах,” *Информационные технологии*, т. 25, № 11, с. 682—690, 2019]

⇒ *аннотация**:

[В статье изложены принципы, лежащие в основе суперкомпьютера “Терапраф”, разработанного в МГТУ им. Баумана.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Попов А.Ю.ПринцОГВСсН-2020ст

:= *стандартное библиографическое описание**:

[А. Ю. Попов, “Принципы организации гетерогенной вычислительной системы с набором команд дискретной математики,” *Информационные технологии*, т. 26, № 2, с. 67—79, 2020]

⇒ *аннотация**:

[В статье изложены принципы, лежащие в основе суперкомпьютера “Терапраф”, разработанного в МГТУ им. Баумана.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Попов Э.В..ИскусИССОиЭС-1990кн

:= *стандартное библиографическое описание**:

[Э. [д. Попов, ред., *Искусственный интеллект справочник : в 3 кн. – М. : Радио и связь, 1990. – Кн. 1 : Системы общения и экспертные системы*]

⇒ *аннотация**:

[В данной книге описаны системы общения и экспертные системы.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Поспелов Д.А.МоделРОАМА-1989кн

:= *стандартное библиографическое описание**:

[Д. А. Поспелов, *Моделирование рассуждений: опыт анализа мыслительных актов*. М.: Радио и связь, 1989]

⇒ *аннотация**:

[Нечеткий вывод]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Поспелов Д.А.СитуаУТиП-1986кн

:= *стандартное библиографическое описание**:

[Д. А. Поспелов, *Ситуационное управление: теория и практика*. М. : Наука, 1986]

⇒ *аннотация**:

[Ситуационное управление – метод управления сложными техническими и организационными системами, основанный на идеях теории искусственного интеллекта: представление знаний об объекте управления и способах управления им на уровне логико-лингвистических моделей, использование обучения и обобщения в качестве основных процедур при построении процедур управления по текущим ситуациям, использование дедуктивных систем для построения многошаговых решений.]

⇐ *библиографическая ссылка**:

- *§ 7.9.2. Библиография OSTIS*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *§ 3.1.6. Понятие класса методов и языка представления методов*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Прангишвили И.В..СовреСПСЭВМсНСиАУПД-1981ст

:= *стандартное библиографическое описание**:
[И. Прангишвили и Г. Стецюра, “Современное состояние проблемы создания ЭВМ с нетрадиционной структурой и архитектурой, управляемых потоком данных,” *Измерение, контроль, автоматизация*, № 1, с. 36—48, 1981]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Пратт Т..ЯзыкиПРиР-2002кн

:= *стандартное библиографическое описание**:
[Т. Пратт и М. Зелковиц, *Языки программирования: разработка и реализация : пер. с англ.* 4-е, под общ. ред. А. Матросова, ред. СПб.: Питер : Питер принт, 2002]

⇒ *аннотация**:
[В книге рассматриваются общие концепции разработки и реализации языков программирования, а также основы формальных грамматик и конечных автоматов - математических моделей, используемых для определения и реализации языков программирования]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

ПрогрУОпДДиО-2022эл

:= *стандартное библиографическое описание**:
[“Программируем управление освещением по датчикам движения и освещения на Node-RED.” (окт. 2022), url: `https://habr.com/ru/post/396985/`]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.5. Элементы технической реализации умных домов*

Рабинович З.Л.НекотБПкСМЦМ-1979ст

:= *стандартное библиографическое описание**:
[З. Рабинович, “Некоторый бионический подход к структурному моделированию целенаправленного мышления,” *Кибернетика*, № 2, с. 115—118, 1979]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Рабинович З.Л.оКонцеМИ-1995ст

:= *стандартное библиографическое описание**:
[З. Рабинович, “О концепции машинного интеллекта,” *Кибернетика и системный анализ*, № 2, с. 163—173, 1995]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Рабинович З.Л.РазвиСУЭВМвСыПАНИ-1979ст

:= *стандартное библиографическое описание**:

[З. Рабинович, “Развитие структур универсальных ЭВМ в связи с проблемами автоматизации научных исследований,” *Автоматика*, № 5, с. 63—72, 1979]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Раговский А.П.ИнтелМСДВ-2011ст

:= *стандартное библиографическое описание**:

[А. П. Раговский, “Интеллектуальная многоагентная система дедуктивного вывода на основе сетевой организации,” *Искусств. интеллект и принятие решений*, № 2, с. 73—86, 2011]

⇒ *аннотация**:

[В статье рассматриваются вопросы практической реализации методов теории многоагентных систем в организации интеллектуальной прикладной системы, которая в качестве основного механизма обработки знаний использует процедуру дедуктивного вывода. Исследуются задачи создания модели агентов, а также организации многоагентной системы. Разрабатывается механизм опосредованной централизованной коммуникации интеллектуальных агентов. Обсуждаются архитектура и принципы функционирования интеллектуальной прикладной многоагентной системы]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Русецкий К.В.ЛингвБЗИО-2013ст

:= *стандартное библиографическое описание**:

[К. В. Русецкий, “Лингвистическая база знаний интеллектуальной обучающей системы по естественному языку,” в *Дистанционное обучение — образовательная среда XXI века: материалы VIII международной научно-методической конференции. (Минск, 5-6 декабря 2013 года)*, БГУИР, Минск, 2013, с. 198—199]

⇒ *аннотация**:

[Интенсивное развитие информационных технологий в последние несколько десятилетий делает возможным создание интеллектуальных помощников в изучении языков. Такие помощники помогали бы пользователям осваивать язык в удобном для них темпе, открывая новые возможности для карьеры и саморазвития. В данной работе описывается подход к построению лингвистической базы знаний интеллектуального образовательного ресурса, который является основой для создания интеллектуальных помощников по изучению языков.]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Рыбаков В.В.МультВНЛЛ-2020ст

:= *стандартное библиографическое описание**:

[В. В. Рыбаков, “Мультиагентные Временные Нетранзитивные Линейные Логики, Проблема Допустимости,” *Алгебра и Логика*, т. 59, с. 123—141, март 2020. DOI: 10.1007/s10469-020-09581-0]

⇒ *аннотация**:

[В данной статье изучается расширение временной логики – мультиагентную логику на моделях с нетранзитивным линейным временем (в некотором смысле расширение интервальной логики). Предлагаемые реляционные модели допускают пробелы в отношениях достижимости агентов – и эти отношения в принципе различны – то есть информация достижимая для одного из агентов может быть недостижима для других. Логический язык использует временные операторы UNTIL и Next (для каждого из агентов), через которые могут вводится модальные операции возможно и необходимо. Главная изучаемая проблема для вводимой логики это проблема распознавания допустимости правил вывода. Ранее эта проблема исследовалась для логики с равномерной фиксированной длинной интервалов транзитивности. Данная работа не предполагает равномерной длины и расширяет логику индивидуальными временными операторами для различных агентов. Находится алгоритм решающий проблему допустимости в данной логике – а именно – распознающий допустимые правила вывода.]

⇐ *библиографическая ссылка**:

- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Самодумкин С.А..СеманТКПИ-2011ст

:= *стандартное библиографическое описание**:

[С. А. Самодумкин и др., “Семантическая технология компонентного проектирования интеллектуальных геоинформационных систем,” в *Открытые семантические технологии проектирования интеллектуальных*

систем (OSTIS-2011) : материалы II Междунар. науч.-техн. конф., Минск, 10–12 февр. 2011 г., Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2011, с. 505—514]

⇒ *аннотация**:

[В работе рассмотрена технология проектирования интеллектуальных геоинформационных систем на основе открытой семантической технологии компонентного проектирования интеллектуальных систем.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Самодумкин С.А.ИнтелГС-2012ст

:= *стандартное библиографическое описание**:

[С. А. Самодумкин, "Интеллектуальные геоинформационные системы," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2012) : материалы II Междунар. науч.-техн. конф., Минск, 16–18 февр. 2012 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2012, с. 521—526]

⇒ *аннотация**:

[Рассмотрены принципы и подходы к проектированию интеллектуальных геоинформационных систем. В основу предлагаемого подхода положено понятие семантической модели интеллектуальной геоинформационной системы и кодирование информации с использованием семантического SC-кода.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Самодумкин С.А.СеманТПИВ-2009ст

:= *стандартное библиографическое описание**:

[С. А. Самодумкин, "Семантическая технология проектирования интеллектуальных вопросно-ответных систем," *Доклады Белорусского государственного университета информатики и радиоэлектроники*, № 7 (45), с. 67—72, 2009]

⇒ *аннотация**:

[Объектом рассмотрения являются интеллектуальные вопросно-ответные системы, которые дают ответы пользователям на широкий спектр вопросов по заданной предметной области. Предложена модель данного класса систем, а также семантическая технология их проектирования.]

⇐ *библиографическая ссылка**:

- *Глава 3.4. Язык вопросов для ostis-систем*

Самодумкин С.А.ТехноПИВС-2009ст

:= *стандартное библиографическое описание**:

[С. А. Самодумкин, "Технология проектирования интеллектуальных вопросно-ответных систем," 2009]

⇒ *аннотация**:

[Объектом рассмотрения являются интеллектуальные вопросно-ответные системы, которые дают ответы пользователям на широкий спектр вопросов по заданной предметной области. Предложена модель данного класса систем, а также семантическая технология их проектирования.]

⇐ *библиографическая ссылка**:

- *Глава 3.4. Язык вопросов для ostis-систем*

Сапатый П.С.АктивИПкМСРЗнГиС-1984ст

:= *стандартное библиографическое описание**:

[П. Сапатый, "Активное информационное поле как модель структурного решения задач на графах и сетях," *Изв. АН СССР. Техн. кибернет*, № 5, с. 184—208, 1984]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Сапатый П.С.ЯзыкВкОНС-1986ст

:= *стандартное библиографическое описание**:

[П. Сапатый, "Язык ВОЛНА-0 как основа навигационных структур для баз знаний на основе семантических сетей," *Изв. АН СССР. Техническая кибернетика.*, № 5, с. 198—210, 1986]

⇒ *аннотация**:

[В работе предложен язык волнового программирования ВОЛНА-0 для баз знаний на основе семантических сетей.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Сергиевская И.М.МатемЛиТА-2004ст

:= *стандартное библиографическое описание**:
[И. Сергиевская, *Математическая логика и теория алгоритмов*. М.: ПГАТИ, 2004]

⇒ *аннотация**:
[Книга И.М. Сергиевской является основой курса дискретной математики, что нашло отражение в данном учебном пособии.]

⇐ *библиографическая ссылка**:
- *§ 2.6.1. Смысловое представление логических формул и формальных теорий классической логики*

Сердюков Р.Е.БазовАиИСО-2004дс

:= *стандартное библиографическое описание**:
[Р. Е. Сердюков, "Базовые алгоритмы и инструментальные средства обработки информации в графодинамических ассоциативных машинах," 114 л., дис. ... канд. техн. наук : 05.13.11, Минск, 2004]

⇒ *аннотация**:
[Работа содержит описание базовых алгоритмов и иструментальных средств обработки информации в графодинамических ассоциативынх машинах]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Серенков П.С..КонцеИСкБЗ-2004ст

:= *стандартное библиографическое описание**:
[П. С. Серенков и др., "Концепция инфраструктуры стандартизации как базы знаний на основе онтологий," 2004, с. 25—29]

⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 7.2. Метасистема OSTIS*

СистеКАОС-эл

:= *стандартное библиографическое описание**:
[*Системы компьютерной алгебры. Общие сведения [Электронный ресурс]*, Режим доступа: `https://math-it.petrsu.ru/users/semenova/CAS/Prezentation/CAS_1.pdf`. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*

СистеКАМ-эл

:= *стандартное библиографическое описание**:
[*Система компьютерной алгебры Maxima [Электронный ресурс*, Режим доступа: `http://maxima.sourceforge.net/ru/`. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

СистеООАД-2017кн

:= *стандартное библиографическое описание**:
["Система обозначений объектов административно-территориального деления и населенных пунктов : ОКРБ 003-2017," Минск: Госстандарт, 2017]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Скрыпников А.В..РешенЗИБсИ-2021ст

:= *стандартное библиографическое описание**:

[А. В. Скрыпников и др., “Решение задач информационной безопасности с использованием искусственного интеллекта,” *Современные наукоемкие технологии*, № 6, с. 277—281, июль 2021]

⇒ *аннотация**:

[Кибербезопасность – быстро развивающаяся область, требующая постоянного усовершенствования благодаря заметному продвижению в облачных сетях и веб-технологиях, онлайн-банкинге, социальных сетях, мобильной среды окружающей, смарт-сетки и прочих. На данный момент в большинстве случаев информационная безопасность является реактивной. Методы машинного обучения могут быть применены во многих сферах науки. Их отличительные свойства – масштабируемость, адаптивность, потенциал. Именно поэтому есть возможность моментально адаптироваться под новые и ранее неизвестные вызовы. Машинным обучением является класс методов ИИ (искусственного интеллекта), характерной чертой данных методов считается не обычное прямой решение поставленной задачи, а обучение в процессе поиска и применения решений сходных задач во множественном количестве. Создание таких машинных методов требует использования средств статистики математической, численных методов, теории вероятности и графов, прочих техник работы с цифровыми данными. Проанализированы типы программных решений, платформ и устройств информационной безопасности с использованием искусственного интеллекта для автоматизированного реагирования на сетевые и локальные угрозы, также на поведение пользователей и различных информационных сущностей. Показана необходимость применения машинного обучения в сфере кибербезопасности.]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*

Смородин В.С..Метод иСИМТ-2007кн

:= *стандартное библиографическое описание**:

[В. Смородин и И. Максимей, *Методы и средства имитационного моделирования технологических процессов производства*. Гомель: ГГУ им. Ф. Скорины, 2007]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *§ 7.7.1. Адаптивное управление технологическим циклом производства на основе Технологии OSTIS*
- *Подпункт 7.7.1.1.1 Качественные характеристики технологических процессов*

Созинова Е.Н.ПримеЭСдАиОИБ-2011ст

:= *стандартное библиографическое описание**:

[Е. Н. Созинова, “Применение экспертных систем для анализа и оценки информационной безопасности,” *Молодой ученый*, № 10(33), с. 64—66, окт. 2011. url: `https://moluch.ru/archive/33/3766/` (дата обр. 29.03.2023)]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

Соколов А.П..СистеАПКМ-2021ст

:= *стандартное библиографическое описание**:

[А. П. Соколов и А. О. Голубев, “Система автоматизированного проектирования композиционных материалов. Часть 3. Графоориентированная методология разработки средств взаимодействия пользователь-система,” *Известия СПБГЭТУ ЛЭТИ*, с. 43—57, 2021]

⇒ *аннотация**:

[Несмотря на активное развитие средств автоматизации, прикладное наукоемкое ПО, включая системы инженерного анализа и автоматизации проектирования, полноценные облачные платформы продолжают оставаться доступными лишь ограниченному кругу специалистов, что обусловлено их дороговизной и, как правило, сложностью применения. Вместе с тем, потребность в таких системах для ученых, инженеров, обучающихся только возрастает. В статье представлен оригинальный графоориентированный метод организации вычислительных процессов в САПР, а также основанная на его применении методология разработки специализированных модулей расширения, реализованная для добавления новых функциональных возможностей в представляемую САПР КМ и позволяющая описывать логику работы пользователя в системе для решения конкретной прикладной задачи (вычислительной, задачи проектирования и пр.). Проведен анализ существующих программных решений в области организации сложных вычислительных процессов в распределенной вычислительной среде и представлен перечень программных средств, которые необходимы для организации

соответствующей программной инфраструктуры. Обоснована актуальность работы в выбранном направлении. Представленные метод и методология призваны упростить процессы как разработки новых возможностей, так и работы в сложном наукоемком программном обеспечении. Представлены некоторые детали программной реализации отдельных элементов разрабатываемой методологии, а также результаты тестирования разработанных программных средств, обеспечивающих взаимодействие пользователя с системой в процессе обхода тестовой графовой модели. Созданные программные средства позволяют визуализировать текущий статус обхода графовой модели в реальном масштабе времени. Приведен пример графовой модели верхнего уровня абстракции, реализующей методологию автоматизированного проектирования КМ в рамках разрабатываемой САПР КМ, построенной с использованием представленной методологии]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

Соколов М.С.Модел иАИОиА-2012дс

:= *стандартное библиографическое описание**:

[М. Соколов, “Модель и алгоритмы интегрированной обработки и анализа пространственной и атрибутивной информации в муниципальных ГИС для поддержки принятия управленческих решений,” дис. ... к-та техн. наук : 05.13.01, Муром, 2012]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Соловов А.В..ДискрММвИПА..-2021ст

:= *стандартное библиографическое описание**:

[А. В. Соловов и А. А. Меньшикова, “Дискретные математические модели в исследовании процессов автоматизированного обучения,” *Информационные технологии*, № 12, с. 43—48, 2021]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Соловов А.В.МоделСЭОР-2007ст

:= *стандартное библиографическое описание**:

[А. В. Соловов, “Моделирование структуры электронных образовательных ресурсов,” *Информационные технологии*, № 3, с. 43—48, 2007]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Соловов А.В.ПроекКСУН-1995кн

:= *стандартное библиографическое описание**:

[А. В. Соловов, *Проектирование компьютерных систем учебного назначения: учебное пособие*. Самара:СГАУ, 2006, с. 140]

⇐ *библиографическая ссылка**:

- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Соловов А.В.ЭлектОПДТ-2006кн

:= *стандартное библиографическое описание**:

[А. В. Соловов, *Электронное обучение: проблематика, дидактика, технология*. Самара: Новая техника, 2006, с. 464]

⇐ *библиографическая ссылка**:

- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Стахин Н.А.ОсновРсСАС-2008кн

:= *стандартное библиографическое описание**:

[Н. А. Стахин, *Основы работы с системой аналитических (символьных) вычислений Maxima*. 2008, с. 86]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Стефанюк.В.Л.ЛокалОИСМ-2004кн

:= *стандартное библиографическое описание**:

[В. Стефанюк, *Локальная организация интеллектуальных систем. Модели и приложения*. М.: Физмалит, 2004]

⇐ *библиографическая ссылка**:

- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.1. Типология кибернетических систем*

Сулейманов Дж.Ш.ИсслеБППС-2001ст

:= *стандартное библиографическое описание**:
[Д. Ш. Сулейманов, “Исследование базовых принципов построения семантического интерпретатора вопросно-ответных текстов на естественном языке в АОС,” *Образовательные технологии и общество*, т. 4, № 3, с. 178—192, 2001]

⇒ *аннотация**:
[Описывается прагматически-ориентированный подход к созданию модели семантической интерпретации вопросно-ответных текстов в АОС в режиме, когда активной стороной является система. Раскрываются базовые принципы построения процессора вопросно-ответных текстов в условиях «ожидаемости» значений вопроса и «детерминированости» контекста.]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*
- *§ 3.4.2. Денотационная семантика Языка вопросов для ostis-систем*

Сулейманов Дж.Ш.СистеСАОТ-2014ст

:= *стандартное библиографическое описание**:
[Д. Ш. Сулейманов, “Система семантического анализа ответных текстов обучаемого на естественном языке,” *Онтология проектирования*, т. 1, № 1, с. 65—77, 2014]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Таберко В.В..ПроекПРПвК-2018ст

:= *стандартное библиографическое описание**:
[В. В. Таберко и др., “Проектирование предприятия рецептурного производства в контексте направления Industry 4.0,” в *Открытые семантические технологии проектирования интеллектуальных систем*, сер. Вып. 2, Минск : БГУИР, 2018, с. 307—320]

⇒ *аннотация**:
[В работе рассматривается проектирование предприятия рецептурного производства в контексте направления Industry.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0*

Таранчук В.Б.ИнтелВАВБ-2019ст

:= *стандартное библиографическое описание**:
[В. Б. Таранчук, “Интеллектуальные вычисления, анализ, визуализация больших данных,” 2019]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Таранчук В.Б.Метод иТРПС-2019ст

:= *стандартное библиографическое описание**:
[В. Б. Таранчук, “Методические и технические решения, примеры создания интеллектуальных образовательных ресурсов,” в *Дистанционное обучение – образовательная среда XXI века : материалы XI Международной научно-методической конференции, Минск, 12-13 декабря 2019 г.*, БГУИР, Минск, 2019, с. 312—313]

⇒ *аннотация**:
[Рассмотрены новые обоснованные и апробированные методические и технические решения, примеры формирования умной информационно-образовательной среды, основанные на модификации и расширении основных инструментов концепции электронного обучения. Приведены примеры реализации в системе управления обучением Moodle путем включения интерактивных ресурсов Computable Document Format.]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Тарасов В.Б.оМногоСкИОФ-2002кн

:= *стандартное библиографическое описание**:

[В. Тарасов, *От многоагентных систем к интеллектуальным организациям: философия, психология, информатика*. М.: Эдиториал УРСС, 2002]

⇒ *аннотация**:

[Книга содержит описание теории агентов, многоагентных систем и интеллектуальных организаций. Исследованы пути создания искусственных сообществ и интеллектуальных организаций]

⇐ *библиографическая ссылка**:

- *§ 7.9.2. Библиография OSTIS*
- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*
- *§ 7.9.2. Методологические проблемы текущего этапа работ в области Искусственного интеллекта*
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Тейз А..ЛогичПкИИоК-1990кн

:= *стандартное библиографическое описание**:

[А. Тейз, П. Грибомон и Ж. и. д. Луи, *Логический подход к искусственному интеллекту: от классической логики к логическому программированию*, Г. Гаврилов, ред. М.: Мир, 1990, ISBN: 5-03-001636-8]

⇒ *аннотация**:

[Монография специалистов из Бельгии и Швейцарии, излагающая проблемы и методы искусственного интеллекта с точки зрения математической логики. Она состоит из шести глав: логика, аксиоматические системы, представление знаний и рассуждений, логика и модифицируемые рассуждения, формальные грамматики и логическое программирование, Пролог и логическое программирование. Книга построена так, что для понимания материала от читателя требуется только знание основ информатики. Для всех изучающих и использующих методы искусственного интеллекта и логического программирования.]

⇐ *библиографическая ссылка**:

- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Тейз А..ЛогичПкИИоМ-1998кн

:= *стандартное библиографическое описание**:

[А. Тейз, П. Грибомон, Г. Юлен и др., *Логический подход к искусственному интеллекту: от модальной логики к логике баз данных*, Г. Гаврилов, ред. М.: Мир, 1998, ISBN: 5-03-002519-7]

⇒ *аннотация**:

[Монография французских математиков, представляющая собой продолжение книги с тем же названием, но другим подзаголовком (М.: Мир, 1990). Она включает основы модальных и временных логик, анализ естественных языков, семантики Монтегю, немонотонных логик и логической теории баз данных. Изложение живое и наглядное, сопровождаемое графическими иллюстрациями, практическими рекомендациями. Для всех изучающих и применяющих методы искусственного интеллекта и логического программирования.]

⇐ *библиографическая ссылка**:

- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.2. Смысловое представление логических формул и высказываний в прикладных логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Тимченко В.А.МетодПКСС-2013ст

:= *стандартное библиографическое описание**:

[В. А. Тимченко, “Метод преобразования классов семантических сетей,” в *Открытые семантические технологии проектированияинтеллектуальных систем = Open Semantic Technologies for IntelligentSystems (OSTIS-2013): материалы III Междунар. научн.-техн. конф.(Минск, 21-23 февраля 2013г.)*, В. В. Голенков и др., ред., Минск: БГУИР, 2013, с. 81—86]

⇐ *библиографическая ссылка**:

- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Углев В.А.АктуаССПС-2012ст

:= *стандартное библиографическое описание**:

[В. А. Углев, “Актуализация содержания стандартов проектирования сложных технических объектов: онтологический подход,” 2012, с. 80—86]

⇐ *библиографическая ссылка**:

- *Глава 7.2. Метасистема OSTIS*

Уилкс М..СостаПдЭСМ-1953кн

:= *стандартное библиографическое описание**:
[Г. С. Уилкс М. Уиллер Д., *Составление программ для электронных счетных машин*. Издательство иностранной литературы, 1953]

⇒ *аннотация**:
[В предлагаемой читателям книге Уилкса, Уилера и Гилла дается подробное описание методов составления программ для построенной в Англии электронной счетной машины "ЭДСАК"; приведены также многочисленные примеры программ для решения отдельных задач и, в частности, так называемые "стандартные подпрограммы заготовленные заранее и используемые для составления более сложных программ]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Филипов А.А..ЕдинаОПИАД-2016ст

:= *стандартное библиографическое описание**:
[А. А. Филиппов и др., “Единая онтологическая платформа интеллектуального анализа данных,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 77—82]

⇒ *аннотация**:
[В данной работе описана единая онтологическая платформа интеллектуального агализа данных.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Филиппов.А.А..ЕдинаяОПИАД-2016ст

:= *стандартное библиографическое описание**:
[А. А. Филиппов и др., “Единая онтологическая платформа интеллектуального анализа данных,” в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2016) : материалы VI Междунар. науч.-техн. конф., Минск, 18–20 февр. 2016 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2016, с. 77—82]

⇒ *аннотация**:
[В данной работе предложена методика разработки, а также основные принципы построения и архитектура единой платформы интеллектуального анализа данных на основе онтологии предметной области. Помимо этого рассмотрен процесс построения предметной онтологии с помощью встроенного в платформу универсального редактора на примере онтологии «Справочник боцмана».]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*

Финн В.К.кФормаОПИПС-1981кн

:= *стандартное библиографическое описание**:
[В. К. Финн, “К формальному определению понятия информационно-поисковой системы,” *НТИ, сер*, т. 2, с. 5—15, 1981]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Финн В.К.ЛогичПИП-1976кн

:= *стандартное библиографическое описание**:
[В. К. Финн, *Логические проблемы информационного поиска*. Наука, 1976]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Фомина Т.А..ПроекАИИС-2020ст

:= *стандартное библиографическое описание**:
[Т. А. Фомина и Г. М. Новикова, “Проектирование адаптивного интерфейса ИС для поддержки деятельности образовательного учреждения,” *Вестник Алтайской академии экономики и права*, т. 6, № 1, с. 125—133, 2020. DOI: 10.17513/vaael.1174. url: https://vaael.ru/ru/article/view?id=1174]

⇒ *аннотация**:

[Рассматриваются принципы проектирования адаптивных интерфейсов для информационных систем, сопровождающих образовательный процесс, в условиях динамичного изменения потребностей экономики. Авторы отмечают, что используемые информационные системы должны позволять работать с разными типами информации. Способы и методы представления информационного контента, сопровождающего деятельность образовательного учреждения, должны зависеть от характеристик пользователя, являющегося потребителем этого контента, и ситуационного контекста использования системы.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*

фон Нейман Д.ТеориСА-1971кн

:= *стандартное библиографическое описание**:

[Д. фон Нейман, *Теория самовоспроизводящихся автоматов*. М.: Мир, 1971]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Фостер К.АссоцПП-1981кн

:= *стандартное библиографическое описание**:

[К. Фостер, *Ассоциативные параллельные процессоры*. Энергоиздат, 1981, с. 240]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Хайкин С.НейроСПК-2006кн

:= *стандартное библиографическое описание**:

[С. Хайкин, *Нейронные сети: полный курс, 2-е издание*. Москва: Издательский дом “Вильямс”, 2006, ISBN: 5845908906]

⇒ *аннотация**:

[Рассмотрены основные парадигмы искусственных нейронных сетей. Дается строгое математическое обоснование всех нейросетевых парадигм, примеры множества практических задач]

⇐ *библиографическая ссылка**:

- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

Хант Э.ИскусИ-1978кн

:= *стандартное библиографическое описание**:

[Э. Хант, “Искусственный интеллект,” 1978]

⇐ *библиографическая ссылка**:

- *Глава 3.4. Язык вопросов для ostis-систем*

Харламов А.А..СеманСкФОР-2011ст

:= *стандартное библиографическое описание**:

[А. А. Харламов и Т. В. Ермоленко, “Семантические сети как формальная основа решения проблемы интеграции интеллектуальных систем. Формализм автоматического формирования семантической сети с помощью преобразования в многомерное пространство,” в *Открытые семантические технологии проектированияинтеллектуальных систем = Open Semantic Technologies for IntelligentSystems (OSTIS-2011): материалы Междунар. научн.-техн. конф. (Минск,10-12 февраля 2011 г.)*, В. В. Голенков и др., ред., Минск: БГУИР, февр. 2011, с. 87—96]

⇐ *библиографическая ссылка**:

- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Хехт-Нильсен Р.НейроИСП-1998ст

:= *стандартное библиографическое описание**:

[Р. Хехт-Нильсен, “Нейрокомпьютинг: история, состояние, перспективы,” *Открытые системы*, № 4, с. 23—28, 1998]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Хоанг В.К..РешенОЗвРППБРсСБД-2013ст

:= *стандартное библиографическое описание**:

[В. К. Хоанг и А. Ф. Тузовский, "Решения основных задач в разработке программы поддержки безопасности работы с семантическими базами данных," *Доклады Томского государственного университета систем управления и радиоэлектроники*, № 2(28), с. 121—125, 2013]

⇒ *аннотация**:

[Рассматриваются вопросы обеспечения безопасности информационных систем. Поставлена задача поддержки безопасности работы с семантическими базами данных. Описаны предлагаемые алгоритмы их решения.]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*

ЦифроКМИО-2007ст

:= *стандартное библиографическое описание**:

["Цифровые карты местности информация, отображаемая на топографических картах и планах населенных пунктов : ОКРБ 012-2007," 2007]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *Пункт 7.8.3.1. Базовая классификация объектов местности*

Частикова В.А..МетодПСАИ-2022ст

:= *стандартное библиографическое описание**:

[В. А. Частикова и А. И. Митюгов, "Методика построения системы анализа инцидентов информационной безопасности на основе нейроиммунного подхода," *Электронный Сетевой Политематический Журнал «Научные Труды Кубгту»*, № 1, с. 98—105, 2022]

⇒ *аннотация**:

[Предложена методика построения нейроиммунной системы анализа инцидентов информационной безопасности (ИБ), объединяющей модули сбора и хранения сжатия) данных, модуль анализа и корреляции событий ИБ и подсистемы обнаружения сетевых атак. Особенностью данной системы является применение разработанных методов на основе нейроиммунного подхода, обеспечивающих решение каждой из задач перечисленных модулей. Нейроиммунные методы сжатия данных, обнаружения вторжений, анализа и корреляции инцидентов ИБ объединяют такие интеллектуальные решения, как сверточные нейронные сети, система, основанная на правилах, гибридная искусственная система. Разработан программный комплекс системы анализа инцидентов безопасности. Проведена оценка эффективности предложенного подхода. Показана эффективность работы комплекса в рамках задачи анализа инцидентов ИБ.]

⇐ *библиографическая ссылка**:

- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.1. Специфика обеспечения информационной безопасности интеллектуальных систем нового поколения*

Шпаков М.В.РазраИГнОН-2004дс

:= *стандартное библиографическое описание**:

[М. Шпаков, "Разработка интеллектуальных геоинформационных на основе настраиваемой объектной модели предметной области," 168 л., дис. ... к-та техн. наук : 05.13.11, Санкт-Петербург, 2004]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Шункевич Д.В..МетодКПСУЗ-2013ст

:= *стандартное библиографическое описание**:

[Д. В. Шункевич, И. Т. Давыденко, Д. Н. Корончик, А. В. Губаревич и др., "Методика компонентного проектирования систем, управляемых знаниями," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 93—110]

⇒ *аннотация**:

[В данной работе рассматривается методика проектирования систем, управляемых знаниями, по технологии OSTIS. В работе подробно рассматривается процесс разработки систем на примере справочной системы

по геометрии, описаны процессы расширения базы знаний, машины обработки знаний и пользовательского интерфейса системы.]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

Шункевич Д.В..СредстваПКПС-2015ст

:= *стандартное библиографическое описание**:
[Д. В. Шункевич, И. Т. Давыденко, Д. Н. Корончик, И. И. Жуков и др., "Средства поддержки компонентного проектирования систем, управляемых знаниями," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2015) : материалы V Междунар. науч.-техн. конф., Минск, 19–21 февр. 2015 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2015, с. 79—88]

⇒ *аннотация**:
[В работе рассматривается подход к проектированию систем, управляемых знаниями, ориентированный на использование совместимых многократно используемых компонентов, что существенно сокращает трудоемкость разработки таких систем.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*

Шункевич Д.В.БазовПТКПМ-2013ст

:= *стандартное библиографическое описание**:
[Д. В. Шункевич, "Базовые понятия технологии компонентного проектирования машин обработки знаний систем дистанционного обучения," в *Дистанционное обучение – образовательная среда XXI века: материалы VIII международной научно-методической конференции. (Минск, 5–6 декабря 2013 года)*, БГУИР, Минск, 2013, с. 186—187]

⇒ *аннотация**:
[В данной статье описаны некоторые основные концепции технологии проектирования машин обработки знаний в рамках проекта OSTIS, такие как стратегия решения задач и операции логического вывода. Также рассмотрены координационные агенты и алгоритмы их работы.]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Шункевич Д.В.Модели иСКПМ-2013ст

:= *стандартное библиографическое описание**:
[Д. В. Шункевич, "Модели и средства компонентного проектирования машин обработки знаний на основе семантических сетей," в *Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013) : материалы III Междунар. науч.-техн. конф., Минск, 21–23 февр. 2013 г.*, Белорус. гос. ун-т информатики и радиоэлектроники ; редкол.: В. В. Голенков (отв. ред.) [и др.], Минск, 2013, с. 269—280]

⇒ *аннотация**:
[В данной работе рассматриваются проблемы существующих методов, средств и технологий построения машин обработки знаний и ставится проблема отсутствия средств, позволяющих относительно неподготовленному разработчику в удовлетворительные сроки проектировать машины обработки знаний для прикладных интеллектуальных систем различного назначения. Далее рассматривается технология, призванная решить поставленную проблему путем интеграции различных методов и способов решения задач на общей формальной основе.]

⇐ *библиографическая ссылка**:
- *5. Методы и средства проектирования интеллектуальных компьютерных систем нового поколения*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*
- *§ 5.3.3. Многократно используемые компоненты решателей задач ostis-систем*

Щедровицкий Г.П.СхемаМССС-1995кн

:= *стандартное библиографическое описание**:
[Г. П. Щедровицкий, *Схема мыследеятельности – системно-структурное строение, смысл и содержание*. М.: Шк. культ. пол., 1995, ISBN: 5-88969-001-9]
⇒ *аннотация**:
[В книге рассматриваются идеи СМД-методологии, предложенной школой Г.П. Щедровицкого.]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Янкелевич С.С..КонцеНВКОнЗ-2019ст
:= *стандартное библиографическое описание**:
[С. С. Янкелевич и Е. С. Антонов, "Концепция нового вида карт, основанного на знаниях," *Вестник СГУГиТ (Сибирского государственного университета геосистем и технологий)*, т. 24, № 4, с. 188—196, 2019]
⇒ *аннотация**:
[В статье рассмотрена роль карты как образно-знаковой геоинформационной модели действительности для быстрого и адекватного восприятия информации. Создание карт в электронном виде, с использованием ГИС-технологий, является важнейшей задачей современного общества, так как именно карта становится тем инструментом, при помощи которого человек может принимать решение, от самого простого до сложного, даже в экстренных ситуациях. Общество предъявляет к картам все больше требований, пользователь, обращаясь к карте, хочет получать достоверную информацию и из огромного массива данных выбирать только те сведения, которые в большей степени подходили бы для принятия правильного решения. Раскрыта роль психологии восприятия геоизображения человеком. Пользователи из разных предметных областей имеют разные когнитивные и ментальные стереотипы. Необходимы новые продукты и технологии, которые будут ориентированы на различных пользователей, адаптированы к особенностям восприятия человека и будут способствовать быстрому и верному принятию решений. Для анализа ситуации и принятия решений должны привлекаться специалисты и эксперты из различных предметных областей. Сделан вывод о том, что нужны новые карты, содержание которых дополнено пространственными знаниями, а также способствует формированию новых знаний.]
⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Янковская А.Е..ГибриИСЭДН-2017ст
:= *стандартное библиографическое описание**:
[А. Е. Янковская и др., "Гибридная интеллектуальная система экспресс-диагностики нарушителей информационной безопасности с учетом экспресс-диагностики организационного стресса, депрессии, девиантного поведения и тревоги нарушителей на основе конвергенции нескольких наук и научных направлений," в *Труды конгресса по интеллектуальным системам и информационным технологиям «IS&IT 17». Научное издание в 3-х томах*, Изд-во Ступина С.А., 2017, с. 323—329]
⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*

Янковская А.Е.АналиДиЗнОКННиНН-2010ст
:= *стандартное библиографическое описание**:
[А. Е. Янковская, "Анализ данных и знаний на основе конвергенции нескольких наук и научных направлений," в *Международная конференция «Интеллектуализация обработки информации» (ИОИ-8), Кипр, г. Пафос, 17–24 октября 2010 г.*, 2010, с. 196—199]
⇐ *библиографическая ссылка**:

Abrams M..UIMLaA-1999art
:= *стандартное библиографическое описание**:
[M. Abrams и др., "UIML: an appliance-independent XML user interface language," *Computer Networks*, т. 31, № 11, с. 1695—1708, 1999, ISSN: 1389-1286. DOI: https://doi.org/10.1016/S1389-1286(99)00044-4. url: https://www.sciencedirect.com/science/article/pii/S1389128699000444]
⇒ *аннотация**:
[Today's Internet appliances feature user interface technologies almost unknown a few years ago: touch screens, styli, handwriting and voice recognition, speech synthesis, tiny screens, and more. This richness creates problems. First, different appliances use different languages: WML for cell phones; SpeechML, JSML, and VoxML for voice enabled

devices such as phones; HTML and XUL for desktop computers, and so on. Thus, developers must maintain multiple source code families to deploy interfaces to one information system on multiple appliances. Second, user interfaces differ dramatically in complexity (e.g, PC versus cell phone interfaces). Thus, developers must also manage interface content. Third, developers risk writing appliance-specific interfaces for an appliance that might not be on the market tomorrow. A solution is to build interfaces with a single, universal language free of assumptions about appliances and interface technology. This paper introduces such a language, the User Interface Markup Language (UIML), an XML-compliant language. UIML insulates the interface designer from the peculiarities of different appliances through style sheets. A measure of the power of UIML is that it can replace hand-coding of Java AWT or Swing user interfaces.**]**

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Ackerman W.B.DataFL-1979art

:= *стандартное библиографическое описание**:
[W. Ackerman, “Data flow language,” *Proc. National Computer Conf. AFIPS Press*, с. 1087—1095, 1979**]**

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Adger D.CoreSaMA-2003bk

:= *стандартное библиографическое описание**:
[D. Adger, *Core Syntax: A Minimalist Approach* (Core linguistics), англ. Oxford University Press, 2003, ISBN: 9780199243709. url: `https://books.google.by/books?id=GMJ1QgAACAAJ`**]**

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.1. Формализация синтаксиса естественных языков*

AEST3250-2004art

:= *стандартное библиографическое описание**:
[“AES Tech 3250-2004 Specification of the digital audio interface (AES/EBU),” Audio Engineering Society, Standard, 2004**]**

⇒ *аннотация**:
[This document specifies a recommended interface for the serial digital transmission of two channels of periodically sampled and linearly represented digital audio data in a broadcasting complex, up to a distance of a few hundred metres.**]**

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Afanasyev I..VGLaHPGPFftSXATVA-2021art

:= *стандартное библиографическое описание**:
[I. V. Afanasyev и др., “VGL: a high-performance graph processing framework for the NEC SX-Aurora TSUBASA vector architecture,” *The Journal of Supercomputing*, т. 77, № 8, с. 8694—8715, янв. 2021. DOI: `10.1007/s11227-020-03564-9`. url: `https://doi.org/10.1007/s11227-020-03564-9`**]**

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Awais A.Proac iIPAaSB-2022art

:= *стандартное библиографическое описание**:
[A. Akbar, “Proactivity in Intelligent Personal Assistants: A Simulation-Based Approach,” в *Multi-Agent Systems*, D. Baumeister и J. Rothe, ред., Springer International Publishing, 2022, с. 423—426, ISBN: 978-3-031-20614-6**]**

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

Allen J..ParalMAfPRS-1989el

:= *стандартное библиографическое описание**:

[J. D. Allen, J. Philip и L. Butler, *Parallel machine architecture for production rule systems*, июнь 1989]
⇒ *аннотация**:
[Патент на архитектуру параллельных машин для систем продукционных правил.]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Almusaylim Z..aRevie oSHPSaC-2019art
:= *стандартное библиографическое описание**:
[Z. A. Almusaylim и N. Zaman, "A review on smart home present state and challenges: linked to context-awareness internet of things (IoT)," *Wireless Networks*, т. 25, № 6, с. 3193—3204, авг. 2019, ISSN: 1572-8196. DOI: 10.1007/s11276-018-1712-5. url: https://doi.org/10.1007/s11276-018-1712-5]
⇒ *аннотация**:
[The smart home is considered as an essential domain in Internet of Things (IoT) applications, it is an interconnected home where all types of things interact with each other via the Internet. This helps to automate the home by making it smart and interconnected. However, at the same time, it raises a great concern of the privacy and security for the users due to its capability to be controlled remotely. Hence, the rapid technologically growth of IoT raises abundant challenges such as how to provide the home users with safe and secure services keeping privacy in the account and how to manage the smart home successfully under the controlled condition to avoid any further secrecy or theft of personal data. A number of the research papers are available to address these critical issues, researchers presented different approaches to overcome these stated issues. This research review will analyze smart home approaches, challenges and will suggest possible solutions for them and illustrate open issues that still need to be addressed.]
⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Alt H.Compu tFDbTPC-1995art
:= *стандартное библиографическое описание**:
[H. Alt и M. GODAU, "Computing the Fréchet Distance between Two Polygonal Curves," англ., *Int. J. Comput. Geometry Appl.*, т. 5, с. 75—91, март 1995. DOI: 10.1142/S0218195995000064]
⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

AmazoAOSWiA-2022el
:= *стандартное библиографическое описание**:
["Amazon Alexa Official Site: What Is Alexa?" Англ. Mode of access: https://developer.amazon.com/alexa. — Date of access: 15.09.2022. ()]
⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Ameri F..ProduLMCtKL-2005art
:= *стандартное библиографическое описание**:
[F. Ameri и D. Dutta, "Product lifecycle management: closing the knowledge loops," *Computer-Aided Design and Applications*, т. 2, № 5, с. 577—590, 2005]
⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения*

Anderson P..SPICE-2016art
:= *стандартное библиографическое описание**:
[P. Anderson и др., "SPICE: Semantic Propositional Image Caption Evaluation," англ., в *Computer Vision – ECCV 2016*, Springer International Publishing, 2016, с. 382—398. DOI: 10.1007/978-3-319-46454-1_24. url: https://doi.org/10.1007/978-3-319-46454-1_24]
⇒ *аннотация**:
[Работа содержит описание алгоритма вычисления подобия между метками изображений]
⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*

- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.2 Автоматическая проверка ответов пользователей*

Andrushevich A..TowarSBGDA-2010art

:= *стандартное библиографическое описание**:
[A. Andrushevich, M. Staub и др., “Towards semantic buildings: Goal-driven approach for building automation service allocation and control,” в *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, IEEE, 2010, с. 1—6**]**

⇒ *аннотация**:
[The idea of applying semantic web technologies to the area of smart homes (SH) and building automation has resulted in a number of research activities and initiatives that have been recently developed. This article starts with an overview of ongoing work towards embedding semantics into home automation services. We then highlight the problems not considered by previous solutions. The core value of the work presented here is contained in a novel goal-driven approach for building automation service allocation and control. A new concept of semantic homes (SeH) and its architectural vision is another significant contribution. The comparison between existing and suggested solutions is rounded off by a use case scenario from the area of ambient assisted living.**]**

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*

Andrushevich A..ZigBe814T-2009art

:= *стандартное библиографическое описание**:
[A. Andrushevich, R. Kistler и др., “Zigbee/ieee 802.15.4 technologies in ambient assisted living applications,” в *3rd European ZigBee Developers’ Conference (EuZDC)*, Citeseer, 2009**]**

⇒ *аннотация**:
[Ambient assisted living environments are full of hu- man attention demanding applications. The introduction of wireless solutions in the assisted living area brings con- vincing advantages in terms of staff burden reduction, observation reliability and treating measures response time. The paper identifies some typical application requirements. Applicability analysis of existing solutions is made. Zig- Bee with TCP/IP fusion initiative is considered as enabling point for ZigBee/IEEE 802.15.4 integration into ambient assisted living environments.**]**

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.4. Подсистемы умного дома*
- *Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми*

Arp R..BuildOwBFO-2015bk

:= *стандартное библиографическое описание**:
[R. Arp, B. Smith и A. Spear, *Building Ontologies with Basic Formal Ontology*, англ. The MIT Press, 2015**]**

⇒ *аннотация**:
[An introduction to the field of applied ontology with examples derived particularly from biomedicine, covering theoretical components, design practices, and practical applications.**]**

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *§ 2.5.6. Онтологии верхнего уровня*

Averin A.I..UsingPiDI-2004art

:= *стандартное библиографическое описание**:
[A. Averin и V. Vagin, “Using parallelism in deductive inference,” англ., *Journal of Computer and Systems Sciences International*, т. 43, с. 603—614, июль 2004**]**

⇒ *аннотация**:
[Problems of applying parallelism in deductive inference are considered. A general classification of parallelism types is given. Two subtypes of parallelism are analyzed: parallel unification and parallelism at the clause level. An approach for representing terms of first-order predicate logic and an algorithm for parallel unification that uses this representation are proposed. An algorithm of parallel inference on graph structures is presented.**]**

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

AXIOM.tSCS-эл

:= *стандартное библиографическое описание**:

[*AXIOM. The Scientific Computation System [Электронный ресурс]*, Режим доступа: `http://axiom-developer.org/index.html`. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Azarov E..InstaHRoSUM..-2013art

:= *стандартное библиографическое описание**:

[E. Azarov, M. Vashkevich и A. A. Petrovsky, "Instantaneous harmonic representation of speech using multicomponent sinusoidal excitation.," в *INTERSPEECH*, 2013, с. 1697—1701]

⇒ *аннотация**:

[This paper introduces a framework for parametric speechmodeling that can be used in various speech applications suchas text-to-speech synthesis, voice conversion etc. In order toreduce impact of pitch variations the harmonic analysis is donein the warped time scale that is aligned with instantaneous pitch values. It is assumed that each harmonic has its own periodic excitation source that evolves in time and can be modeled as a sum of several sinusoidal components with close frequencies. The parameters of the excitation components are estimated using a modified instantaneous Prony's method. The proposed analysis/synthesis technique is compared with TANDEM-STRAIGHT.]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Backus J.CanPBLftNS-1978art

:= *стандартное библиографическое описание**:

[J. Backus, "Can programming be liberated from the von Neumann style?" *Communications of the ACM*, т. 21, № 8, с. 613—641, авг. 1978. DOI: `10.1145/359576.359579`. url: `https://doi.org/10.1145/359576.359579`]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*

Bai J..fPlatf tKGEoLA-2022art

:= *стандартное библиографическое описание**:

[J. Bai и др., "From platform to knowledge graph: evolution of laboratory automation," англ., *JACS Au*, т. 2, № 2, с. 292—309, 2022]

⇒ *аннотация**:

[High-fidelity computer-aided experimentation is becoming more accessible with the development of computing power and artificial intelligence tools. The advancement of experimental hardware also empowers researchers to reach a level of accuracy that was not possible in the past. Marching toward the next generation of self-driving laboratories, the orchestration of both resources lies at the focal point of autonomous discovery in chemical science. To achieve such a goal, algorithmically accessible data representations and standardized communication protocols are indispensable. In this perspective, we recategorize the recently introduced approach based on Materials Acceleration Platforms into five functional components and discuss recent case studies that focus on the data representation and exchange scheme between different components. Emerging technologies for interoperable data representation and multi-agent systems are also discussed with their recent applications in chemical automation. We hypothesize that knowledge graph technology, orchestrating semantic web technologies and multi-agent systems, will be the driving force to bring data to knowledge, evolving our way of automating the laboratory.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Bakker B.ReinfLwLSTM-2001art

:= *стандартное библиографическое описание**:

[B. Bakker, "Reinforcement Learning with Long Short-Term Memory," англ., в *NIPS'01: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS, 2001, с. 1475—1482]

⇒ *аннотация**:

[This paper presents reinforcement learning with a Long Short-Term Memory recurrent neural network: RL-LSTM. Model-free RL-LSTM using Advantage learning and directed exploration can solve non-Markovian tasks with long-term dependencies between relevant events. This is demonstrated in a T-maze task, as well as in a difficult variation of the pole balancing task.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*

Balaji P.G..aIntro tMAS-2010art
:= *стандартное библиографическое описание**:
[P. G. Balaji и D. Srinivasan, “An introduction to multi-agent systems,” в *Innovations in multi-agent systems and applications-1*, Springer, 2010, с. 1—27]
⇐ *библиографическая ссылка**:
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Ballinger C.tTeradSS-2009art
:= *стандартное библиографическое описание**:
[C. Ballinger, “The teradata scalability story,” *Technical report, Teradata Corporation*, 2009]
⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Bantsevich K.A.Metas otOSTIS-2022art
:= *стандартное библиографическое описание**:
[K. Bantsevich, “Metasystem of the ostis technology and the standard of the ostis technology,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. Iss. 6, BSUIR, Minsk, 2022, с. 357—368]
⇒ *аннотация**:
[In the article, an approach to automating the processes of creation, development, and application of standards based on the OSTIS Technology is proposed. The general problems related to the development and usage of modern standards in various fields are considered. Standardization of intelligent computer systems is proposed, as well as standardization of methods and means of their design within the proposed approach.]
⇐ *библиографическая ссылка**:
- *Глава 7.2. Метасистема OSTIS*

Bantsevich K.A.Struc oKBoNGI-2022art
:= *стандартное библиографическое описание**:
[K. Bantsevich, “Structure of knowledge bases of next-generation intelligent computer systems: A hierarchical system of subject domains and their corresponding ontologies,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. Iss. 6, BSUIR, Minsk, 2022, с. 87—98]
⇒ *аннотация**:
[The article is dedicated to the ontological approach to the design of knowledge bases of next-generation intelligent computer systems. This approach is based on the representation of the knowledge base as a hierarchical structure of interrelated subject domains and their ontologies built on the basis of top-level ontologies.]
⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

BasicFOB-2021el
:= *стандартное библиографическое описание**:
[“Basic Formal Ontology (BFO).” англ. (апр. 2021), url: `https://github.com/BFO-ontology/BFO`]
⇒ *аннотация**:
[The Basic Formal Ontology (BFO) is a small, upper-level ontology that is designed for use in supporting information retrieval, analysis and integration in scientific and other domains.]
⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *§ 2.5.6. Онтологии верхнего уровня*

Bateman J.A..aLinguOoSfNLP-2010art
:= *стандартное библиографическое описание**:
[J. A. Bateman, J. Hois и др., “A linguistic ontology of space for natural language processing,” англ., *Artificial Intelligence*, т. 174, № 14, с. 1027—1071, 2010, ISSN: 0004-3702. DOI: `https://doi.org/10.1016/j.artint.2010.05.008`. url: `https://www.sciencedirect.com/science/article/pii/S0004370210000858`]
⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Bateman J.A..tGenerUMKBO-2002art

:= *стандартное библиографическое описание**:
[J. A. Bateman и G. Fabris, “The Generalized Upper Model Knowledge Base: Organization and Use,” 2002]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Bateman J.A.tTheorSoOiNLP-1997art

:= *стандартное библиографическое описание**:
[J. A. Bateman, *The Theoretical Status of Ontologies in Natural Language Processing*, Available at: `https://arxiv.org/abs/cmp-lg/9704010`, (accessed 2022, October)]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Bay H..SpeedURF-2008art

:= *стандартное библиографическое описание**:
[H. Bay и др., “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, т. 110, № 3, с. 346—359, 2008. DOI: `10.1016/j.cviu.2007.09.014`]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Bayer R..PrefiBT-1977art

:= *стандартное библиографическое описание**:
[R. Bayer и K. Unterauer, “Prefix B-trees,” *ACM Transactions on Database Systems (TODS)*, т. 2, № 1, с. 11—26, 1977]

⇒ *аннотация**:
[Two modifications of B-trees are described, simple prefix B-trees and prefix B-trees. Both store only parts of keys, namely prefixes, in the index part of a B*-tree. In simple prefix B-trees those prefixes are selected carefully to minimize their length. In prefix B-trees the prefixes need not be fully stored, but are reconstructed as the tree is searched. Prefix B-trees are designed to combine some of the advantages of B-trees, digital search trees, and key compression techniques while reducing the processing overhead of compression techniques.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.5. Реализация файловой памяти в Программной платформе ostis-систем*

Beaudin M..HomeEMSaRoM-2015art

:= *стандартное библиографическое описание**:
[M. Beaudin и H. Zareipour, “Home energy management systems: A review of modelling and complexity,” *Renewable and Sustainable Energy Reviews*, т. 45, № C, с. 318—335, 2015. DOI: `10.1016/j.rser.2015.01.04`. url: `https://ideas.repec.org/a/eee/rensus/v45y2015icp318-335.html`]

⇒ *аннотация**:
[The increasing demand for electricity and the emergence of smart grids have presented new opportunities for home energy management systems (HEMS) in demand response markets. HEMS are demand response tools that shift and curtail demand to improve the energy consumption and production profile of a dwelling on behalf of a consumer. HEMS usually create optimal consumption and production schedules by considering multiple objectives such as energy costs, environmental concerns, load profiles, and consumer comfort.]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью*

Behounek L.Intro tMFL-2011bk

:= *стандартное библиографическое описание**:
[L. Behounek, P. Cintula и P. Hájek, “Introduction to mathematical fuzzy logic,” в янв. 2011, т. 37, с. 1—101]

⇐ *библиографическая ссылка**:
- *Глава 2.6. Смысловое представление логических формул и высказываний в различного вида логиках*
- *§ 2.6.3. Смысловое представление логических формул и высказываний в неклассических логиках*

Belazzougui D..FastPSiLSwA-2010art

:= *стандартное библиографическое описание**:
[D. Belazzougui и др., “Fast prefix search in little space, with applications,” англ., в *European Symposium on Algorithms*, 2010, с. 427—438]

⇒ *аннотация**:
[A prefix search returns the strings out of a given collection S that start with a given prefix. Traditionally, prefix search is solved by data structures that are also dictionaries, that is, they actually contain the strings in S. For very large collections stored in slow-access memory, we propose extremely compact data structures that solve weak prefix searches they return the correct result only if some string in S starts with the given prefix. Our data structures for weak prefix search use O(|S|log L) bits in the worst case, where L is the average string length, as opposed to O(|S| L) bits for a dictionary. We show a lower bound implying that this space usage is optimal.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.5. Реализация файловой памяти в Программной платформе ostis-систем*

Bellegarda J..SpokeLUfNIS-2014art

:= *стандартное библиографическое описание**:
[J. R. Bellegarda, “Spoken language understanding for natural interaction: The siri experience,” *Natural interaction with robots, knowbots and smartphones*, с. 3—14, 2014]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Benjamins V.R..KnowlSTOaPS-1999art

:= *стандартное библиографическое описание**:
[V. R. Benjamins и A. Gomez-Perez, “Knowledge-System Technology: Ontologies and Problem-Solving Methods,” янв. 1999]

⇒ *аннотация**:
[В работе говорится о важности повторного использования онтологий и методов решения задач как компонентов систем, основанных на знаниях.

Приводится обзор сущствующих определений понятия онтология и актуальных подходов к их разработке на момент публикации работы. Рассматриваются методологии построения онтологий, проекты по их разработке, а также известные приложения, использующие онтологии. Делается попытка описать архитектуру метода решения задач (МРЗ), отмечается взаимосвзяь МРЗ с классом задач и предметной областью. Рассматриваются основные существующие на тот момент МРЗ, однако не делается попытки их унификации или хотя бы классификации. Кратко рассматриваются подходы к разработке библиотек МРЗ. Делается вывод о необходимости интеграции МРЗ и онтологий, специфицирующих знания предметной области.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Berners-Lee T..SemanW-2001art

:= *стандартное библиографическое описание**:
[T. Berners-Lee, J. Hendler и O. Lassila, “The semantic web,” *Scientific american*, т. 284, № 5, с. 34—43, 2001]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

Besold T.R..NeuraSLaRaS-2017art

:= *стандартное библиографическое описание**:
[T. R. Besold и др., “Neural-Symbolic Learning and Reasoning: A Survey and Interpretation,” нояб. 2017, (accessed 2020, Jun). url: `https://arxiv.org/pdf/1711.03902.pdf`]

⇒ *аннотация**:
[This joint survey reviews the personal ideas and views of several researchers on neural-symbolic learning and reasoning]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Bhumij G..aOverv oWStFoRT-2018art

:= *стандартное библиографическое описание**:

[Bhumij Gupta1, Dr. M.P. Vani, “An Overview of Web Sockets: The future of Real-Time Communication,” в *International Research Journal of Engineering and Technology (IRJET)*, 2018]

⇒ *аннотация**:

[As the development and implementation of HTML 5 (Hypertext Markup Language 5), it has opened a new range of possibilities for real-time communication between client and server. Currently, vastly used methods for asynchronous realtime communication are HTTP Polling and HTTP Long Polling. But a new protocol is recently introduced called Web sockets(WS) and Web sockets secure(WSS). Web Sockets allow full-duplex communication in HTML5 compliant browser over a single socket. This paper will cover the basic overview of Web sockets, and see if it is a better option than its competitors.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.6. Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*

Biallas M..LivinSaAiaAtHF-2017art

:= *стандартное библиографическое описание**:

[M. Biallas и др., “Living safely and actively in and around the home: Four applied examples from avatars and ambient cubes to active walkers,” в *Safe at Home with Assistive Technology*, Springer, 2017, с. 5—30]

⇒ *аннотация**:

[Assistive technology may support people in staying longer independent and improving the quality of life inside and outside of their homes. However, bringing all necessary aspects together to finally bring a product successfully to the market is challenging, as many researchers in the field have experienced over the last years. It means to tackle the technical issues arising, very closely study the potential customers getting to know their needs, their barriers of acceptance and knowing how to creating value for them and to eventually market, sell and ship the final product to them. We believe that only a joint effort of an interdisciplinary team involving technology, human sciences and business partners will have a chance of success. The following pages document four successful and promising AAL projects and their teams, results and experiences on the way to get there.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.4. Подсистемы умного дома*
- *Пункт 7.6.4.2. Подсистема наблюдения за одинокими пожилыми людьми*

Black A.aAppro tCSS-1993th

:= *стандартное библиографическое описание**:

[A. Black, “An approach to computational situation semantics,” дис. . . . док., PhD thesis, Department of Artificial Intelligence, University of Edinburgh . . ., 1993]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*

Blahser J..ThineAfaFTDP-2021art

:= *стандартное библиографическое описание**:

[J. Blahser, T. Goller и M. Bohmer, “Thine — Approach for a fault tolerant distributed packet manager based on hypercore protocol,” англ., в *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, 2021. DOI: 10.1109/compsac51774.2021.00266. url: https://doi.org/10.1109/compsac51774.2021.00266]

⇒ *аннотация**:

[Существует ряд менеджеров пакетов, которые используются почти в каждом проекте разработки программного обеспечения для интеграции сторонних пакетов в проект. Эти пакеты обычно хранятся в центральных репозиториях в облаке. Этот центральный облачный подход имеет некоторые недостатки, особенно в отношении отказоустойчивости, которую мы хотим решить с помощью нашего подхода, вдохновленного мыслями о росистых вычислениях. В этой статье мы представляем полностью распределенный менеджер пакетов, который можно использовать как в облаке, так и на периферии или в гибридной среде. Подход основан на одноранговой сети, реализованной с помощью протокола Hypercore.]

⇐ *библиографическая ссылка**:

- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Bohm D.tUndivUaOIoQT-1993bk

:= *стандартное библиографическое описание**:

[D. Bohm и B. Hiley, *The Undivided Universe: An Ontological Interpretation of Quantum Theory*, англ. London: Routledge, 1993]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Bohm D.Whole atIO-2002bk

≔ *стандартное библиографическое описание**:
[D. Bohm, *Wholeness and the Implicate Order*, англ. London: Routledge, 2002]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Boissier O..MultiAOPwJC-2013art

≔ *стандартное библиографическое описание**:
[O. Boissier и др., “Multi-agent Oriented Programming with JaCaMo,” англ., *Science of Computer Programming*, т. 78, № 6, с. 747—761, июнь 2013]

⇒ *аннотация**:
[Статья объединяет парадигмы программирования, возникшие в результате исследований в области многоагентных систем.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Boley H..DigitEPaS-2007art

≔ *стандартное библиографическое описание**:
[H. Boley и E. Chang, “Digital Ecosystems: Principles and Semantics,” в *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*, 2007, с. 398—403. DOI: 10.1109/DEST.2007.372005]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

Bordini R.H..ProgrMASiAS-2007bk

≔ *стандартное библиографическое описание**:
[R. H. Bordini, J. F. Hubner и M. J. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, англ. Chichester: Wiley, 2007]

⇒ *аннотация**:
[Книга описывает и объясняет расширение AgentSpeak, интерпретируемое платформой Jason, и показывает, как создавать мультиагентные системы с использованием платформы Jason.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Boriskin A.S..OntolBDoISU-2017art

≔ *стандартное библиографическое описание**:
[A. S. Boriskin и др., “Ontology-Based Design of Intelligent Systems User Interface,” англ., в *Open semantic technologies for intelligent systems*, Minsk : BSUIR, 2017, с. 95—106]

⇒ *аннотация**:
[The work is devoted to the development of intelligent systems user interfaces design technology, which is based on an ontological model of the interfaces itself and the ontological model of the design process.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*

Bork D..aOpenPfMMCtO-2019art

≔ *стандартное библиографическое описание**:
[D. Bork и др., “An open platform for modeling method conceptualization: the OMiLAB digital ecosystem,” 2019]

⇒ *аннотация**:

[This paper motivates, describes, demonstrates in use, and evaluates the Open Models Laboratory (OMiLAB)—an open digital ecosystem designed to help one conceptualize and operationalize conceptual modeling methods.]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*

Bouayad-Agha N..NaturLGitCotSW-2014art

:= *стандартное библиографическое описание**:

[N. Bouayad-Agha, G. Casamayor и L. Wanner, "Natural Language Generation in the context of the Semantic Web," англ., *Semantic Web*, т. 5, с. 493—513, янв. 2014. DOI: 10.3233/SW-130125]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Brdnik S..IntelUIaTEaS-2022art

:= *стандартное библиографическое описание**:

[S. Brdnik, T. Heričko и B. Šumak, "Intelligent User Interfaces and Their Evaluation: A Systematic Mapping Study," *Sensors*, т. 22, 2022. DOI: 10.3390/s22155830]

⇒ *аннотация**:

[Intelligent user interfaces (IUI) are driven by the goal of improvement in human–computer interaction (HCI), mainly improving user interfaces' user experience (UX) or usability with the help of artificial intelligence. The main goal of this study is to find, assess, and synthesize existing state-of-the-art work in the field of IUI with an additional focus on the evaluation of IUI. This study analyzed 211 studies published in the field between 2012 and 2022. Studies are most frequently tied to HCI and SE domains. Definitions of IUI were observed, showing that adaptation, representation, and intelligence are key characteristics associated with IUIs, whereas adaptation, reasoning, and representation are the most commonly used verbs in their description. Evaluation of IUI is mainly conducted with experiments and questionnaires, though usability and UX are not considered together in evaluations. Most evaluations (81% of studies) reported partial or complete improvement in usability or UX. A shortage of evaluation tools, methods, and metrics, tailored for IUI, is noticed. Most often, empirical data collection methods and data sources in IUI evaluation studies are experiment, prototype development, and questionnaire.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Briscoe B..DigitESOoEA-2008art

:= *стандартное библиографическое описание**:

[G. Briscoe и P. D. Wilde, "Digital Ecosystems: Self-Organisation of Evolving Agent Populations," *CoRR*, т. abs/0803.2675, 2008. arXiv: 0803.2675. url: http://arxiv.org/abs/0803.2675]

⇐ *библиографическая ссылка**:

- *Глава 7.3. Структура Экосистемы OSTIS*

Brownston L..ProgrESiOPS-1985bk

:= *стандартное библиографическое описание**:

[L. Brownston, R. Farrell и K. E., *Programming Expert Systems in OPS5*, июль 1985]

⇒ *аннотация**:

[This book presents techniques for rule-based programming of expert systems in OPS5. OPS5 is a widely-used language designed to simplify rule-based programming. This book is aimed at experienced programmers in industry and universities, and contains examples and exercises with answers. The first section of the book is a tutorial. The development of a small, self-contained OPS5 program is followed from problem definition to testing. The second section considers the nature of production-system architectures, and compares OPS5 with other tools for programming expert systems.]

⇐ *библиографическая ссылка**:

- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

Brukle H.J.HighLLOHaPVNE-1978art

:= *стандартное библиографическое описание**:

[B. H.J., "High level language oriented hardware and post - von Neumann era," *Proc. 5-th Symp Computer Architecture*, с. 60—65, 1978]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*

- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Buitelaar P..aMultiMLMfO-2006art

:= *стандартное библиографическое описание**:
[P. Buitelaar, M. Sintek и M. Kiesel, “A Multilingual/Multimedia Lexicon Model for Ontologies,” англ., в *The Semantic Web: Research and Applications*, Y. Sure и J. Domingue, ред., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, с. 502—513, ISBN: 978-3-540-34545-9]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Buitelaar P..LingiDaAoaMftIoL-2006art

:= *стандартное библиографическое описание**:
[P. Buitelaar, T. Declerck и др., “LingInfo: Design and Applications of a Model for the Integration of Linguistic Information in Ontologies,” англ., 2006. url: `http://smartweb.dfki.de/Vortraege/OntoLex2006.pdf`]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Buitelaar P..TowarLGO-2009art

:= *стандартное библиографическое описание**:
[P. Buitelaar, P. Cimiano и др., “Towards Linguistically Grounded Ontologies,” в *The Semantic Web: Research and Applications*, L. Aroyo и др., ред., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, с. 111—125, ISBN: 978-3-642-02121-3]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Бухараев Р.Г..СеманАвВОС-1990кн

:= *стандартное библиографическое описание**:
[Р. Г. Бухараев и Д. Ш. Сулейманов, *Бухараев РГ, Сулейманов ДШ Семантический анализ в вопросно-ответных системах*. Казань: Изд-во Казан. ун-та, 1990]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Burstrom T..SoftwENaitFaDS-2022art

:= *стандартное библиографическое описание**:
[T. Burstrom и др., “Software ecosystems now and in the future: A definition, systematic literature review, and integration into the business and digital ecosystem literature,” *IEEE Transactions on Engineering Management*, 2022]

⇒ *аннотация**:
[В работе приведен литературный обзор касательно понятия цифровой экосистемы.]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

Caldarola E..Appro tOIfORiK-2015art

:= *стандартное библиографическое описание**:
[E. G. Caldarola, A. Picariello и A. M. Rinaldi, “An approach to ontology integration for ontology reuse in knowledge based digital ecosystems,” с. 1—8, 2015]

⇒ *аннотация**:
[In this paper, an approach to ontology reuse based on heterogeneous matching techniques will be presented]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*

Calonder M..BRIEF-2010art

:= *стандартное библиографическое описание**:
[M. Calonder и др., “BRIEF: Binary Robust Independent Elementary Features,” в Springer Berlin Heidelberg, 2010, с. 778—792. DOI: `10.1007/978-3-642-15561-1_56`]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*

- *Пункт 4.4.8.2. Локальные признаки изображений*

Calzolari N.Acqui aRSIiaLK-1991art

:= *стандартное библиографическое описание**:
[N. Calzolari, “Acquiring and Representing Semantic Information in a Lexical Knowledge Base.,” англ., июнь 1991, с. 235—243, ISBN: 978-3-540-55801-9. DOI: 10.1007/3-540-55801-2_38]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Cao O..BehavIaNP-2014art

:= *стандартное библиографическое описание**:
[L. Cao и др., “Behavior Informatics: A New Perspective,” *IEEE Intelligent Systems*, т. 29, № 4, с. 62—80, июль 2014. DOI: 10.1109/mis.2014.60. url: https://doi.org/10.1109/mis.2014.60]

⇒ *аннотация**:
[В работе рассматривается применение идей бихевиоризма в информатике.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Cao O.IndepBUaUtBI-2010art

:= *стандартное библиографическое описание**:
[L. Cao, “In-depth behavior understanding and use: The behavior informatics approach,” *Information Sciences*, т. 180, № 17, с. 3067—3085, сент. 2010. DOI: 10.1016/j.ins.2010.03.025. url: https://doi.org/10.1016/j.ins.2010.03.025]

⇒ *аннотация**:
[В работе рассматривается применение идей бихевиоризма в информатике.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Carnie A.Syntax aGL-2012bk

:= *стандартное библиографическое описание**:
[A. Carnie, *Syntax: A Generative Introduction* (Introducing Linguistics), англ. Wiley, 2012, ISBN: 9781118321874. url: https://books.google.by/books?id=MFZ1UV3YGtgC]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.1. Формализация синтаксиса естественных языков*

Castelvecchi D.CanWOtBB-2016art

:= *стандартное библиографическое описание**:
[D. Castelvecchi, “Can we open the black box of AI?” *Nature News*, т. 538, № 7623, окт. 2016]

⇒ *аннотация**:
[Machine learning is becoming ubiquitous in basic research as well as in industry. But for scientists to trust it, they first need to understand what the machines are doing]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Castillo O..RecenAoHIS-2014bk

:= *стандартное библиографическое описание**:
[O. Castillo, P. Melin и J. Kacprzyk, *Recent advances on hybrid intelligent systems*, англ. Berlin: Springer, 2014]

⇒ *аннотация**:
[В работе рассматривается построение гибридных интеллектуальных систем на основе многоагентных систем.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Chakraborty A..SmartHSaCR-2023art

:= *стандартное библиографическое описание**:

[A. Chakraborty и др., “Smart Home System: A Comprehensive Review,” *Journal of Electrical and Computer Engineering*, т. 2023, I. Skliarova, ред., с. 7 616 683, март 2023, Publisher: Hindawi, ISSN: 2090-0147. DOI: 10.1155/2023/7616683. url: https://doi.org/10.1155/2023/7616683**]**

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Chaparro O..otImpac oROoCQM-2014art

:= *стандартное библиографическое описание**:

[O. Chaparro и др., “On the impact of refactoring operations on code quality metrics,” в *2014 IEEE International Conference on Software Maintenance and Evolution*, IEEE, 2014, с. 456—460**]**

⇒ *аннотация**:

[Refactorings are behavior-preserving source code transformations. While tool support exists for (semi) automatically identifying refactoring solutions, applying or not a recommended refactoring is usually up to the software developers, who have to assess the impact that the transformation will have on their system. Evaluating the pros (e.g., the bad smell removal) and cons (e.g., side effects of the change) of a refactoring is far from trivial. We present RIPE (Refactoring Impact Prediction), a technique that estimates the impact of refactoring operations on source code quality metrics. RIPE supports 12 refactoring operations and 11 metrics and it can be used together with any refactoring recommendation tool. RIPE was used to estimate the impact on 8,103 metric values, for 504 refactorings from 15 open source systems. 38% of the estimates are correct, whereas the median deviation of the estimates from the actual values is 5% (with a 31% average).**]**

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Chatterjee B..nBlockDUGwWCAB-2022art

:= *стандартное библиографическое описание**:

[B. Chatterjee и др., “Non-Blocking Dynamic Unbounded Graphs with Worst-Case Amortized Bounds,” en, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. DOI: 10.4230/LIPICS.OPODIS.2021.20. url: https://drops.dagstuhl.de/opus/volltexte/2022/15795/**]**

⇒ *аннотация**:

[В работе рассмотрены основные концепции lock-free алгоритмов.**]**

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

Chen C..MPEoRaGD-2022art

:= *стандартное библиографическое описание**:

[C. Chen и др., “Multi-Perspective Evaluation of Relational and Graph Databases,” англ., 2022**]**

⇒ *аннотация**:

[How to store data is an enduring topic in the computer science field, and traditional relational databases have done this well and are still widely used today. However, with the growth of non-relational data and the challenges in the big data era, a series of NoSQL databases have come into view. Thus, comparing, evaluating, and choosing a better database has become a worthy topic of research. In this thesis, an experiment that can store the same data set and execute the same tasks or workload on the relational, graph and multi-model databases is designed. The investigation proposes how to adapt relational data, tables on a graph database and, conversely, store graph data on a relational database. Similarly, the tasks performed are unified across query languages. We conducted exhaustive experiments to compare and report the performance of the three databases. In addition, we propose a workload classification method to analyze the performance of the databases and compare multiple aspects of the database from an end-user perspective. We have selected PostgreSQL, ArangoDB, Neo4j as representatives. The comparison in terms of task execution time does not have any database that completely wins. The results show that relational databases have performance advantages for tasks such as data import, but the execution of multi-table join tasks is slow and graph algorithm support is lacking. The multi-model databases have impressive support for simultaneous storage of multiple data formats and unified language queries, but the performance is not outstanding. The graph database has strong graph algorithm support and intuitive support for graph query language, but it is also important to consider whether the format and interrelationships of the original data, etc. can be well converted into graph format.**]**

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Chen G..TempoLIfFDoS-2021art

:= *стандартное библиографическое описание**:

[G. Chen, P. Wei и M. Liu, “Temporal Logic Inference for Fault Detection of Switched Systems With Gaussian Process Dynamics,” англ., *IEEE Transactions on Automation Science and Engineering*, с. 1—16, май 2021. DOI: 10.1109/TASE.2021.3074548]

⇒ *аннотация**:

[In this article, we present a method for constructing the fault detector in the form of signal temporal logic (STL) formulas, which can be understood by human users and formally proven to detect faults with probabilistic satisfaction guarantees, for a class of switched nonlinear systems with partially unknown dynamics. First, the partially unknown internal dynamics are approximated by the Gaussian process with stability guarantees. Second, a novel temporal logic inference algorithm is proposed to find the fault detector, which takes advantage of the internal properties of temporal logic and searches for the optimal formula along a partially ordered direction. Moreover, the algorithm is not allowed for missing faults but allowed for false alarms during the temporal logic inference process. In addition, we simulate finitely many trajectories with Chua’s circuit and infer the temporal logic formulas with the Gaussian optimization. The results show that the proposed method can find a temporal logic formula to detect the faulty trajectory with a probability guarantee. Note to Practitioners —The method proposed in this article can be used to detect faults for switched systems with partially unknown dynamics. STL is used to describe the behaviors of the system, which acts as a classifier and detector, such that all normal behaviors of the system will satisfy the description, while the faulty behaviors will violate the description. Moreover, STL formulas can be understood by human operators, which is important for the timely response to faulty events. For example, the normal behavior of a smart grid can be described as follows: “if the smart grid is safe, it should reach 9 kV within 15 min when the voltage to region A is above 12 kV,” which can be expressed with STL. Due to the unknown dynamics, the Gaussian process regression is applied to estimate the model and the region that is robust to noises.]

⇐ *библиографическая ссылка**:

- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Chen N..SelfSDLfSC-2021art

:= *стандартное библиографическое описание**:

[N. Chen, C. You и Y. Zou, “Self-supervised dialogue learning for spoken conversational question answering,” в *Proc. Interspeech 2021*, 2021, с. 231—235]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Chiarcos C..Inter oCaA-2012art

:= *стандартное библиографическое описание**:

[C. Chiarcos, “Interoperability of Corpora and Annotations,” в *Linked Data in Linguistics: Representing and Connecting Language Data and Language Metadata*, C. Chiarcos, S. Nordhoff и S. Hellmann, ред. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, с. 161—179, ISBN: 978-3-642-28249-2. DOI: 10.1007/978-3-642-28249-2_16. url: https://doi.org/10.1007/978-3-642-28249-2%5C_16]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Chiarcos C..Ontol oLASaP-2012art

:= *стандартное библиографическое описание**:

[C. Chiarcos, “Ontologies of Linguistic Annotation: Survey and perspectives,” в *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, Istanbul, Turkey: European Language Resources Association (ELRA), май 2012, с. 303—310. url: http://www.lrec-conf.org/proceedings/lrec2012/pdf/911%5C_Paper.pdf]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Cho J..StramMtToCBS-2019art

:= *стандартное библиографическое описание**:

[J.-H. Cho и др., “STRAM: Measuring the Trustworthiness of Computer-Based Systems,” *ACM Comput. Surv.*, т. 51, № 6, февр. 2019]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

Chu Y.Archi oaHDI-1977art
:= *стандартное библиографическое описание**:
[Y. Chu, “Architecture of a Hardware Data Interpreter,” *Proc. 4-th IEEE Symp. on Computer Architecture*, с. 1—9, 1977]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Chu Y.Evolu oCMS-1976art
:= *стандартное библиографическое описание**:
[Y. Chu, “Evolution of Computer Memory Structure,” *Proc. National Computer Conf. AFIPS Press*, с. 733—748, 1976]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Cimiano P..ExploOLfGNL-2013art
:= *стандартное библиографическое описание**:
[P. Cimiano, J. Lüker и др., “Exploiting Ontology Lexica for Generating Natural Language Texts from RDF Data,” в *Proceedings of the 14th European Workshop on Natural Language Generation*, Sofia, Bulgaria: Association for Computational Linguistics, авг. 2013, с. 10—19. url: `https://aclanthology.org/W13-2102`]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Cimiano P..Lexon aMfOLfON-2007art
:= *стандартное библиографическое описание**:
[P. Cimiano, P. Haase и др., “LexOnto: A Model for Ontology Lexicons for Ontology-based NLP,” англ., 2007]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Clips-2015эл
:= *стандартное библиографическое описание**:
[“Clips - a tool for building expert systems [Electronic resource],” англ., Mode of access: `http://clipsrules.sourceforge.net`. — Date of access: 18.02.2018]
⇒ *аннотация**:
[Официальный сайт Clips]
⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

Collatz L.FunctAaNM-1966bk
:= *стандартное библиографическое описание**:
[*Functional Analysis and Numerical Mathematics*, англ. New York, San Francisco, London: Academic Press, 1966]
⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Cordella L.P..aSubGIAfLG-2004art
:= *стандартное библиографическое описание**:
[L. P. Cordella и др., “A (sub) graph isomorphism algorithm for matching large graphs,” *IEEE transactions on pattern analysis and machine intelligence*, т. 26, № 10, с. 1367—1372, 2004]
⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*

- *Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу*

Cortana-2022el

:= *стандартное библиографическое описание**:
[“Cortana helps you achieve more with less effort. Your personal productivity assistant helps you stay on top of what matters, follow through, and do your best work.” англ. Mode of access: `https://www.microsoft.com/en-us/cortana`. — Date of access: 15.09.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

CUDAT-2023el

:= *стандартное библиографическое описание**:
[*CUDA Toolkit*, English, Mode of access: `https://developer.nvidia.com/cuda-toolkit`. — Date of access: 29.03.2023]

⇒ *аннотация**:
[Официальный сайт проекта CUDA Tookit]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Cypher-2022эл

:= *стандартное библиографическое описание**:
[*Chapter 3. Cypher [Electronic resource] // Neo4j*, English, Mode of access: `http://neo4j.com/docs/stable/cypher-query-lang.html`. — Date of access: 01.12.2022]

⇒ *аннотация**:
[Страница официального сайта Графовой СУБД Neo4j, посвященная языку запросов Cypher]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Библиография OSTIS*

Davies E.R..AdvanMaDLiCV-2021bk

:= *стандартное библиографическое описание**:
[E. Davies и M. Turk, ред., *Advanced Methods and Deep Learning in Computer Vision*, English. Academic Press, 2022]

⇒ *аннотация**:
[Книга посвящена современным подходам к использованию методов обработки информации, машинного обучения и глубокого обучения в задачах компьютерного зрения.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.8. Базовые понятия компьютерного зрения в задаче трехмерной реконструкции*

Davydenko I.T.OntolBDoIS-2017art

:= *стандартное библиографическое описание**:
[I. T. Davydenko, “Ontology-based design of intelligent systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 1, Minsk : BSUIR, 2017, с. 57—72]

⇒ *аннотация**:
[Работа содержит описание онтологического проектирования интеллектуальных систем]

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

Davydenko I.T.SemanMMaToKB-2018art

:= *стандартное библиографическое описание**:
[I. Davydenko, “Semantic models, method and tools of knowledge bases coordinated development based on reusable components,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 2, BSUIR, Minsk, 2018, с. 99—118]

⇒ *аннотация**:
[Работа содержит описание семантических моделей, методов и средств согласованной разработки баз знаний на основе многократно используемых компонентов]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *§ 4.2.3. Методика и средства разработки естественно-языковых интерфейсов*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.1. Действия и методики проектирования баз знаний ostis-систем*

Deepa P..aRepor oVRSTM-2021art

:= *стандартное библиографическое описание**:
[P. Deepa и R. Khilar, “A Report on Voice Recognition System: Techniques, Methodologies and Challenges using Deep Neural Network,” в *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2021, с. 1—5. DOI: 10.1109/i-PACT52855.2021.9697005]

⇒ *аннотация**:
[The main contribution of this work is to provide a brief overview of the field of deep neural networks and voice recognition, describing its system, underlying approaches, and challenges.]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Degottex G..MixedSMaiAVT-2013art

:= *стандартное библиографическое описание**:
[G. Degottex и др., “Mixed source model and its adapted vocal tract filter estimate for voice transformation and synthesis,” *Speech Communication*, т. 55, № 2, с. 278—294, 2013]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Deli V..SpeecTPBoNM-2019art

:= *стандартное библиографическое описание**:
[V. Delić и др., “Speech technology progress based on new machine learning paradigm.,” *Computational intelligence and neuroscience*, 2019]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

DOLCE-2009el

:= *стандартное библиографическое описание**:
[“DOLCE : a Descriptive Ontology for Linguistic and Cognitive Engineering [Electronic resource],” англ., *Laboratory for Applied Ontology*, Mode of access: http://www.loa.istc.cnr.it/old/DOLCE.html. — Date of access: 26.12.2016]

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *§ 2.5.6. Онтологии верхнего уровня*

Dijkstra E.ProgrD-1978bk

:= *стандартное библиографическое описание**:
[D. E., *Programming Discipline*. M.: Mir, 1978, с. 277]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.3. Предлагаемый подход к разработке технологий программирования для ostis-систем*

Dijkstra E.W.CoopeSP-2002bk

:= *стандартное библиографическое описание**:
[E. W. Dijkstra, “Cooperating Sequential Processes,” в *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls*. Berlin, Heidelberg: Springer-Verlag, 2002, с. 65—138, ISBN: 0387954015]

⇒ *аннотация**:
[В работе рассмотрены базовые алгоритмы синхронизации параллельных процессов, положившие начало критическим секциям и более сложным механизмам синхронизации.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

Dillon T..GridSGSotWE-2007art

:= *стандартное библиографическое описание**:

[T. Dillon, C. wu и E. Chang, “GRIDSpace: Semantic Grid Services on the Web-Evolution towards a SoftGrid,” англ., в *3rd International Conference on Semantics, Knowledge, and Grid, SKG 2007*, нояб. 2007, с. 7—13]

⇒ *аннотация**:

[This paper advances the notion of software in existing grid services as discussed in OGSA, which provides software middleware or wrappers for accessing hardware resources towards the notion that the resources provided can be hardware, software, or hybrid hardware/software. It also proposes an approach using the integration of grid service, semantic grid, and Web2.0 to overcome some of the limitations of the existing Web services architecture which relies on having the Web version of the RPC mechanism and thus has difficulty in dealing with massive scale of user participation and communication across the Internet. The new approach produces a novel grid architecture - GRIDSpace.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

Dillon T..OntolBSESE-2008art

:= *стандартное библиографическое описание**:

[T. Dillon, E. Chang и P. Wongthongtham, “Ontology-Based Software Engineering-Software Engineering 2.0,” апр. 2008, с. 13—23, ISBN: 978-0-7695-3100-7. DOI: 10.1109/ASWEC.2008.4483185]

⇒ *аннотация**:

[This paper describes the use of ontologies in different aspects of software engineering. This use of ontologies varies from support for software developers at multiple sites to the use of an ontology to provide semantics in different categories of software, particularly on the Web. The world’s first and only software engineering ontology and a project management ontology in conjunction with a domain ontology are used to provide support for software development that is taking place at multiple sites. Ontologies are used to provide semantics to deal with heterogeneity in the representation of multiple information sources, enable the selection and composition of web services and grid resources, provide the shared knowledge base for multiagent systems, provide semantics and structure for trust and reputation systems and privacy based systems and codification of shared knowledge within different domains in business, science, manufacturing, engineering and utilities. They, therefore, bring a new paradigm to software engineering through the use of semantics as a central mechanism which will revolutionize the way software is developed and consumed in the future leading to the development of software as a service bringing about the dawn of software engineering 2.0.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

Dobrescu R..ConteACaMSwI-2019art

:= *стандартное библиографическое описание**:

[R. Dobrescu, D. Merezeanu и S. Mocanu, “Context-aware control and monitoring system with IoT and cloud support,” *Computers and Electronics in Agriculture*, т. 160, с. 91—99, 2019, Publisher: Elsevier]

⇒ *аннотация**:

[The main goal of the paper is to integrate three emergent technologies (Internet of Things, Cloud Computing and Context awareness) in a multi-layered architecture for developing real-time process control agriculture application. For this, the paper presents original solutions for a Control and Monitoring unit (CMU) which performs real-time control as an entity running on an IoT platform, a Context-aware Control Platform (CaCP) with a three-tier architecture, serving as middleware mechanism for interfacing environmental sensors with IoT and Cloud and a four-level architecture to perform agriculture process control, that includes the CMU and CaCP modules. These solutions are validated by a case study application implemented on an IBM Bluemix IoT platform which performs automatic control of an irrigation system using context aware adaptation of controller parameters in response to environmental changes.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Dobrov A..CompuOoTfMD-2018art

:= *стандартное библиографическое описание**:

[A. Dobrov и др., “Computer Ontology of Tibetan for Morphosyntactic Disambiguation,” англ., в *Digital Transformation and Global Society*, D. A. Alexandrov и др., ред., Cham: Springer International Publishing, 2018, с. 336—349, ISBN: 978-3-030-02846-6]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Dorri A..MultiASaS-2018art

:= *стандартное библиографическое описание**:
[A. Dorri, S. S. Kanhere и R. Jurdak, “Multi-agent systems: A survey,” *Ieee Access*, т. 6, с. 28 573—28 593, 2018]

⇒ *аннотация**:
[Proposed a high-level comprehensive discussion regarding diverse aspects of MAS which helps newcomers to grasp basic concepts of MAS, study existing applications in multiple disciplines, the challenges in developing MAS, and the methods to study MAS performance]

⇐ *библиографическая ссылка**:
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Dubrovin E..GraphRMftDMI-2022art

:= *стандартное библиографическое описание**:
[E. N. Dubrovin и A. Y. Popov, “Graph Representation Methods for the Discrete Mathematics Instructions Set Computer,” в *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, янв. 2020, с. 1925—1930. DOI: 10.1109/EIConRus49466.2020.9039222]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Duchi J..AdaptSMfOL-2011art

:= *стандартное библиографическое описание**:
[J. Duchi, E. Hazan и Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, 2011]

⇒ *аннотация**:
[The paper presents a new family of subgradient methods that incorporate knowledge of the data’s geometry observed in earlier iterations for more informative gradient-based learning. The authors describe an apparatus for adaptively modifying the proximal function and give several efficient algorithms for empirical risk minimization problems with common regularization functions and domain constraints. They show through experimental studies that the adaptive subgradient methods outperform state-of-the-art, non-adaptive subgradient algorithms. The text concludes that their paradigm allows for finding predictive but rarely seen features and simplifies setting a learning rate, resulting in regret guarantees as good as the best proximal function chosen in hindsight]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

Dutta S.KnowlPaAAI-1993bk

:= *стандартное библиографическое описание**:
[S. Dutta, *Knowledge processing and applied artificial intelligence*, англ. Oxford : Butterworth-Heinemann, 1993]

⇒ *аннотация**:
[В данной книге представлены обсуждения бизнес потенциала обработки знаний и рассматриваются аспекты прикладной технологии искусственого интеллекта.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

EAGLERftMAoC-2022el

:= *стандартное библиографическое описание**:
[*EAGLES Recommendations for the Morphosyntactic Annotation of Corpora*, Available at: https://home.uni-leipzig.de/burr/Verb/htm/LinkedDocuments/annotate.pdf, (accessed 2022, October)]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Eden A..Probl itOoCP-2007art

:= *стандартное библиографическое описание**:

[A. H. Eden и R. Turner, “Problems in the ontology of computer programs,” *Applied Ontology*, т. 2, № 1, с. 13—36, 2007]

⇒ *аннотация**:

[As a first step in the larger project of charting the ontology of computer programs, we pose three central questions: (1) Can programs, hardware, and metaprograms be organized into a meaningful taxonomy? (2) To what ontology are computer programs committed? (3) What explains the proliferation of programming languages and how do they come about? Taking the complementary perspectives software engineering and mathematical logic, we take inventory of programs and related objects and conclude that the notions of abstraction and concretization take a central role in this investigation.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.2. Существующие онтологии языков программирования*

Ehlert P.IntelUIIaS-2003bk

:= *стандартное библиографическое описание**:

[P. Ehlert, *Intelligent User Interfaces: Introduction and Survey*. февр. 2003, с. 39]

⇒ *аннотация**:

[Research on new communication methods focuses on natural language systems, gesture recognition, image recognition, and multi-modal interfaces. Adapting to the user is done by using techniques from artificial intelligence to perform reasoning and learning, for example user modeling and plan recognition. Intelligent interface agents, which are anthropomorphic computerized beings, combine several of these techniques..]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Emel’yanov G.M..Analy oSRiCoSIioS-2007art

:= *стандартное библиографическое описание**:

[G. Emel’yanov, D. Mikhailov и N. Stepanova, “Analysis of semantic relations in classification of sense images of statements,” *Pattern Recognition and Image Analysis*, т. 17, с. 274—278, 2007]

⇒ *аннотация**:

[Semantic relations in classification of images of natural-language statements are analyzed. The classification of the sense images of statements is considered on an example of construction of semantic relations in the RussNet thesaurus. The semantic statements are formalized with mathematical methods of the lattice theory.]

⇐ *библиографическая ссылка**:

- *Глава 3.4. Язык вопросов для ostis-систем*

Erekhinskaya T.N..TenWoLOfR-2020art

:= *стандартное библиографическое описание**:

[T. N. Erekhinskaya и др., “Ten Ways of Leveraging Ontologies for Rapid Natural Language Processing Customization for Multiple Use Cases in Disjoint Domains,” англ., *Open J. Semantic Web*, т. 7, с. 33—51, 2020]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Eve aWBAP-2023el

:= *стандартное библиографическое описание**:

[*Eve – a web-based agent platform [Electronic resource]*, English, Mode of access: `http://eve.almende.com/index.html`. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Официальный сайт проекта Eve.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Evertsz R..ImpleIMAS-2004art

:= *стандартное библиографическое описание**:

[R. Evertsz и др., “Implementing Industrial Multi-agent Systems using JACK,” англ., в *Programming multi-agent systems : first Intern. workshop, PROMAS 2003, Melbourne, 15 July, 2003 : sel. rev. a. invited papers*, M. M. Dastani, J. Dix и A. E. Fallah-Seghrouchni, ред., Berlin, 2004, с. 18—48]

⇒ *аннотация**:
[Статья содержит обсуждение концепций агентного программирования, связанных с платформой JACK.]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Excelente-Toledo C.B..tDynamSoCM-2004art
:= *стандартное библиографическое описание**:
[C. B. Excelente-Toledo и N. R. Jennings, “The Dynamic Selection of Coordination Mechanisms,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 9, № 1/2, с. 55—85, 2004]
⇒ *аннотация**:
[В статье рассматривается структура принятия решений, которая позволяет автономным агентам динамически выбирать механизм, который они используют для координации их взаимосвязанных действий.]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Farrar S..aCommoOfLC-2002art
:= *стандартное библиографическое описание**:
[S. Farrar, W. D. Lewis и T. Langendoen, “A Common Ontology for Linguistic Concepts,” 2002]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Farrar S..aLinguOftSW-2003art
:= *стандартное библиографическое описание**:
[S. Farrar и D. Langendoen, “A Linguistic Ontology for the Semantic Web,” *Glot International*, т. 7, с. 97—100, март 2003]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Fayans A.M..atOntoloTTaMotS-2020art
:= *стандартное библиографическое описание**:
[A. Fayans и V. Kneller, “About the ontology of task types and methods of their solution,” *Ontology of Designing*, т. 10, № 3, с. 273—295, окт. 2020. DOI: 10.18287/2223-9537-2020-10-3-273-295. url: https://doi.org/10.18287/2223-9537-2020-10-3-273-295]
⇐ *библиографическая ссылка**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *§ 3.1.2. Понятие задачи*

Ferber J..fAgent tOaOVoM-2003art
:= *стандартное библиографическое описание**:
[J. Ferber, O. Gutknecht и F. Michel, “From agents to organizations: an organizational view of multi-agent systems,” в *International workshop on agent-oriented software engineering*, Springer, 2003, с. 214—230]
⇐ *библиографическая ссылка**:
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Fernandes E.Speec aVRM-2022el
:= *стандартное библиографическое описание**:
[E. Fernandes. “Speech and Voice Recognition Market / Verified Market Research.” (2022), url: https://www.globenewswire.com/en/news-release/2022/09/06/2510621/0/en/Speech-and-Voice-Recognition-Market-size-worth-59-6-Billion-Globally-by-2030-at-22-57-CAGR-Verified-Market-Research.html#]
⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Finin T..KQML aaACL-1994art
:= *стандартное библиографическое описание**:

[T. Finin и др., “KQML as an agent communication language,” англ., в *CIKM’94 : proc. of the third Intern. Conf. on Inform. a. Knowledge Management, Gaithersburg, 29 Nov. – 2 Dec. 1994*, Assoc. for Computing, New York, 1994, с. 456—463]

⇒ *аннотация**:

[В статье описывается разработка и эксперименты с языком запросов и манипулирования знаниями (KQML), языком и протоколом для обмена информацией и знаниями]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Finn V.K.IntelIOGZiO-2021bk

:= *стандартное библиографическое описание**:

[V. Finn, “Intellekt, informatsionnoe obshchestvo, gumanitarnoe znanie i obrazovanie,” *Intelligence, information society, humanitarian knowledge and education, LENAND, Moscow, Russia*, 2021]

⇐ *библиографическая ссылка**:

- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

FIPAACL-2023el

:= *стандартное библиографическое описание**:

[“FIPA ACL message structure specification [Electronic resource],” English, *The Foundation for Intelligent Physical Agent*, Mode of access: `http://www.fipa.org/specs/fipa00061/SC00061G.html`. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Стандарт языка взаимодействия агентов.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Flotyski J..OntolBRaMoS-2017art

:= *стандартное библиографическое описание**:

[J. Flotyński и K. Walczak, “Ontology-Based Representation and Modelling of Synthetic 3D Content: A State-of-the-Art Review,” англ., *Computer Graphics Forum*, т. 36, № 8, с. 329—353, 2017. DOI: `10.1111/cgf.13083`. url: `https://doi.org/10.4324/9781315210551-7`]

⇐ *библиографическая ссылка**:

- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции*

Foggia P..aPerfoCoFAfG-2001art

:= *стандартное библиографическое описание**:

[P. Foggia, C. Sansone и M. Vento, “A performance comparison of five algorithms for graph isomorphism,” в *Proceedings of the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, Citeseer, 2001, с. 188—199]

⇒ *аннотация**:

[Despite the significant number of isomorphism algorithms presented in the literature, till now no efforts have been done for characterizing their performance. Consequently, it is not clear how the behavior of those algorithms varies as the type and the size of the graphs to be matched varies in case of real applications. In this paper we present a benchmarking activity for characterizing the performance of a bunch of algorithms for exact graph isomorphism. To this purpose we use a large database containing 10,000 couples of isomorphic graphs with different topologies (regular graphs, randomly connected graphs, bounded valence graph), enriched with suitably modified versions of them for simulating distortions occurring in real cases. The size of the considered graphs ranges from a few nodes to about 1000 nodes.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу*

Ford B..CompoD-2019art

:= *стандартное библиографическое описание**:

[B. Ford, R. Schiano-Phan и J. A. Vallejo, “Component Design,” англ., в *The Architecture of Natural Cooling*, Routledge, 2019, с. 160—174. DOI: 10.4324/9781315210551-7. url: https://doi.org/10.4324/9781315210551-7]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*

Forgy C..ReteaFAftMPMOPMP-2019art

:= *стандартное библиографическое описание**:
[C. L. Forgy, “Rete: a fast algorithm for the many pattern/many object pattern match problem,” Elsevier Science Publishers Ltd., 1982, с. 17—37. DOI: 10.1016/0004-3702(82)90020-0. url: https://doi.org/10.1016/0004-3702(82)90020-0]

⇒ *аннотация**:
[The Rete Match Algorithm is an efficient method for comparing a large collection of patterns to a large collection of objects. It finds all the objects that match each pattern. The algorithm was developed for use in production system interpreters, and it has been used for systems containing from a few hundred to more than a thousand patterns and objects. This article presents the algorithm in detail. It explains the basic concepts of the algorithm, it describes pattern and object representations that are appropriate for the algorithm, and it describes the operations performed by the pattern matcher.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.2. Языки продукционного программирования, используемые ostis-системами*

Fortin S.tGraphIP-1996art

:= *стандартное библиографическое описание**:
[S. Fortin, “The graph isomorphism problem,” 1996]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу*

Frame-2022el

:= *стандартное библиографическое описание**:
[“FrameNet.” англ. Mode of access: http://framenet.icsi.berkeley.edu/. — Date of access: 18.10.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Fritzson P.ModelLO-2014art

:= *стандартное библиографическое описание**:
[P. Fritzson, “Modelica Library Overview,” англ., в *Principles of Object Oriented Modeling and Simulation with Modelica 3.3*, John Wiley & Sons, Inc., дек. 2014, с. 909—975. DOI: 10.1002/9781118989166.ch16. url: https://doi.org/10.1002/9781118989166.ch16]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Fry R.L.tEngin oCS-2002art

:= *стандартное библиографическое описание**:
[R. L. Fry, “The engineering of cybernetic systems,” в *AIP Conference Proceedings*, American Institute of Physics, т. 617, 2002, с. 497—528]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*

Fujiwara M..PreneNFTiS-2021art

:= *стандартное библиографическое описание**:

[M. Fujiwara и T. Kurahashi, “Prenex Normal Form Theorems in Semi — Classical Arithmetic,” *The Journal of Symbolic Logic*, т. 86, № 3, с. 1124—1153, июнь 2021. DOI: 10.1017/jsl.2021.47. url: https://doi.org/10.1017/jsl.2021.47]

⇒ *аннотация**:

[The work contains a description of the Prenex paradigm theorem in semi-classical arithmetic]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*

GAMAP-2023el

:= *стандартное библиографическое описание**:

[*GAMA Platform [Electronic resource]*, English, Mode of access: http://gama-platform.org/. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Официальный сайт проекта GAMA.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Gao R..PerfoMfISaE-2002art

:= *стандартное библиографическое описание**:

[R. Gao и L. Tsoukalas, “Performance metrics for intelligent systems: An engineering perspective,” *NIST SPECIAL PUBLICATION SP*, с. 5—10, 2002]

⇐ *библиографическая ссылка**:

- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы*

Garcez A.A..NeuraLaRCaC-2015art

:= *стандартное библиографическое описание**:

[A. d. Garcez и др., “Neuralsymbolic learning and reasoning: Contributions and challenges,” *In: McCallum, A., Gabrilovich, E., Guha, R., Murphy, K. (eds.) Proceedings of the AAAI 2015 Propositional Rule Extraction under Background Knowledge 11 Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches. AAAI Press Technical Report*, т. SS-15-03, 2015]

⇒ *аннотация**:

[Neural-symbolic computation aims at integrating robust connectionist learning algorithms with sound symbolic reasoning. The recent impact of neural learning, in particular of deep networks, has led to the creation of new representations that have, so far, not really been used for reasoning. Results on neural-symbolic computation have shown to offer powerful alternatives for knowledge representation, learning and inference in neural computation. This paper presents key challenges and contributions of neural-symbolic computation to this area]

⇐ *библиографическая ссылка**:

- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Gaulke W..UsingPOtLM-2015art

:= *стандартное библиографическое описание**:

[W. Gaulke и J. Ziegler, “Using profiled ontologies to leverage model driven user interface generation,” июнь 2015, с. 254—259. DOI: 10.1145/2774225.2775070]

⇒ *аннотация**:

[Based on an overview of existing approaches this paper sets out a conceptual framework which combines both task model and ontology based concepts. It is shown that the proposed combination leads to more abstract and reusable task models]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Gavrilova T..VisuaAaaRF-2009art

:= *стандартное библиографическое описание**:

[T. Gavrilova и N. Goulyakina, “Visual Atlases as a Research Framework,” англ., в *Proceeding of rhe 11th International Workhop on Computer Science and Information Technologies SCIT-2009*, т. 1, Crete, Greece, 2009, с. 70—75]

⇐ *библиографическая ссылка**:
- *§ 7.5.4. Представление дидактической информации в базах знаний ostis-систем*

Geramian A..FuzzyISAfF-2017art

:= *стандартное библиографическое описание**:
[A. Geramian и др., “Fuzzy inference system application for failure analyzing in automobile industry,” англ., *International Journal of Quality and Reliability Management*, с. 1493—1507, 2017. url: https://doi.org/10.1108/IJQRM-03-2016-0026]

⇒ *аннотация**:
[Nowadays, quality is one of the most important key success factors in the automobile industry. Improving the quality is based on optimizing the most important quality characteristics and usually launched by highly applied techniques such as failure mode and effect analysis (FMEA). According to the literature, however, traditional FMEA suffers from some limitations. Reviewing the literature, on one hand, shows that the fuzzy rule-base system, under the artificial intelligence category, is the most frequently applied method for solving the FMEA problems. On the other hand, the automobile industry, which highly takes advantages of traditional FMEA, has been deprived of benefits of fuzzy rule-based FMEA (fuzzy FMEA). Thus, the purpose of this paper is to apply fuzzy FMEA for quality improvement in the automobile industry.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Gerhard D.ProduLMCoC-2017art

:= *стандартное библиографическое описание**:
[D. Gerhard, “Product lifecycle management challenges of CPPS,” *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects*, с. 89—110, 2017]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.3. Семантически совместимые интеллектуальные корпоративные ostis-системы различного назначения*

GlobaVAMbT-2019el

:= *стандартное библиографическое описание**:
[“Global Voice Assistant Market By Technology, By Application, By End User, By Region, Competition, Forecast & Opportunities, 2024.” англ. Mode of access: https://www.businesswire.com/news/home/20190916005535/en/Global-Voice-Assistant-Market-Projected-Grow-1.2. — Date of access: 15.12.2019. ()]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Glorot X..Under tDoTDFN-2010art

:= *стандартное библиографическое описание**:
[X. Glorot и Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Journal of Machine Learning Research - Proceedings Track*, т. 9, с. 249—256, янв. 2010]

⇒ *аннотация**:
[The text discusses the successful training of deep neural networks with new initialization or training mechanisms since 2006, showing the superiority of deeper architectures. The author’s objective is to understand why standard gradient descent from random initialization performs poorly with deep neural networks and to design better algorithms in the future. The text examines the influence of non-linear activation functions, finding that the logistic sigmoid activation is unsuitable for deep networks with random initialization due to saturation. The text proposes a new non-linearity that saturates less and a new initialization scheme that brings faster convergence. The author also studies how activations and gradients vary across layers and during training, suggesting that training may be more difficult when the singular values of the Jacobian associated with each layer are far from 1]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

SIL:GoLT-2022el

:= *стандартное библиографическое описание**:
[“SIL: Glossary of Linguistic Terms.” англ. Mode of access: https://glossary.sil.org. — Date of access: 21.10.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.1. Формализация синтаксиса естественных языков*

GOAL-2023el

:= *стандартное библиографическое описание**:
[“GOAL [Electronic resource],” English, *Atlassian*, Mode of access: `https://goalapl.atlassian.net/wiki/spaces/GOAL/overview`. — Date of access: 18.01.2023]

⇒ *аннотация**:
[Язык для программирования агентов.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Godfrey M.D..tCompu avNPI-1993art

:= *стандартное библиографическое описание**:
[M. D. Godfrey и D. F. Hendry, “The computer as von Neumann planned it,” *IEEE Annals of the History of Computing*, т. 15, № 1, с. 11—21, 1993. DOI: `10.1109/85.194088`. url: `https://doi.org/10.1109/85.194088`]

⇒ *аннотация**:
[Статья, в которой рассматривается архитектура фон Неймана, ее особенности и основные недостатки.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.2. Методы и средства реализации ostis-систем*

Golenkov V..otCurreSaCoAI-2022art

:= *стандартное библиографическое описание**:
[V. Golenkov, N. Guliakina, V. Golovko и др., “On the Current State and Challenges of Artificial Intelligence,” англ., в *Open semantic technologies for intelligent systems*, Springer, 2022, с. 1—18]

⇒ *аннотация**:
[In the article, the strategic goals of Artificial intelligence and the main problems of scientific and technological activities in this field are described. The problems relevant for the development of the main directions and forms of its activity are defined. Approaches to their solution, based on a new technological wave, are suggested. Issues important for the development of this scientific and technological discipline are discussed.]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Golenkov V.V..ArtifISIaK-2020art

:= *стандартное библиографическое описание**:
[V. Golenkov, N. Guliakina, V. Golovko и др., “Artificial Intelligence Standardization Is a Key Challenge for the Technologies of the Future,” англ., в *Open Semantic Technologies for Intelligent System*, Cham: Springer International Publishing, 2020, с. 1—21, ISBN: 978-3-030-60446-2. DOI: `10.1007/978-3-030-60447-9_1`]

⇒ *аннотация**:
[Artificial intelligence (AI) is the future of computer technologies and at present it has achieved great progress in different areas. In this paper, we study the general AI problem from a standardization point of view, and introduce a semantic interoperability for intelligent computer systems. We propose a standard for the interior semantic representation of knowledge in the memory of an intelligent computer system, which is called the SC-code (Semantic Code). Integration of various types of knowledge is performed due to hybrid knowledge base models. This model includes a hierarchical set of top-level ontologies that provide semantic compatibility of various types of knowledge and permits to integrate facts, specifications of various objects, logical statements, events, situations, programs and algorithms, processes, problem formulations, domain models, ontologies, and so on. A variant of the presentation of the AI standard based on the semantic representation of knowledge, in the form of a part of the knowledge base of the Intelligent Computer Metasystem (IMS.ostis) is proposed.]

⇐ *библиографическая ссылка**:

- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*

Golenkov V.V..Metho aTfECoC-2019art

:= *стандартное библиографическое описание**:
[V. Golenkov, N. Guliakina, I. Davydenko и др., “Methods and tools for ensuring compatibility of computer systems,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 4, BSUIR, Minsk, 2019, с. 25—52]

⇒ *аннотация**:
[Работа содержит описание методов и средств обеспечения совместимости компьютерных систем]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*
- *Глава 7.9. Обеспечение информационной безопасности в рамках Экосистемы OSTIS*
- *§ 7.9.2. Принципы, лежащие в основе обеспечения информационной безопасности ostis-систем*
- *Глава 7.2. Метасистема OSTIS*
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Глава 2.1. Информационные конструкции и языки*
- *§ 2.1.2. Внешние информационные конструкции и внешние языки ostis-систем*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Golenkov V.V..MethoPotCSoW-2021art

:= *стандартное библиографическое описание**:
[V. Golenkov, N. Guliakina, V. Golovko и др., “Methodological problems of the current state of works in the field of artificial intelligence,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 5, BSUIR, Minsk, 2021, с. 17—24]

⇒ *аннотация**:
[Работа содержит описание методов и средств обеспечения совместимости компьютерных систем]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Golenkov V.V..NextGICS-2022art

:= *стандартное библиографическое описание**:
[V. V. Golenkov и N. A. Gulyakina, “Next-Generation Intelligent Computer Systems and Technology of Complex Support of their Life Cycle,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 6, Minsk : BSUIR, 2022, с. 27—40]

⇒ *аннотация**:
[The paper considers the principles of building next-generation intelligent computer systems, as well as the principles of building a comprehensive technology for their development and life cycle support — OSTIS Technology. Semantic compatibility and interoperability are highlighted as the key properties of the next-generation intelligent systems. The paper considers an approach to providing these properties, realized within the framework of OSTIS Technology.]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*

Golenkov V.V..OntolDoBME-2017art

:= *стандартное библиографическое описание**:
[V. V. Golenkov и other, “Ontology-based Design of Batch Manufacturing Enterprises,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2017, с. 265—280. url: `https://libeldoc.bsuir.by/handle/123456789/12256`]

⇒ *аннотация**:
[This paper presents an ontology-based approach to design batch manufacturing enterprises]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.1 Проблемы разработки систем комплексной автоматизации*

- *Подпункт 7.7.2.2.1 Подходы к автоматизации предприятий*

Golenkov V.V..Princ oOaAotSC-2019art

:= *стандартное библиографическое описание**:

[V. V. Golenkov, D. V. Shunkevich и др., “Principles of organization and automation of the semantic computer systems development,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 3, Minsk : BSUIR, 2019, с. 53—90]

⇒ *аннотация**:

[Работа посвящена принципам разработки семантических компьютерных систем нового поколения на основе Открытой семантической технологии проектирования интеллектуальных систем (Технологии OSTIS). Обоснованы преимущества перехода от традиционных компьютерных систем к семантическим компьютерным системам с точки зрения процесса их проектирования, а также рассмотрены преимущества реализации средств автоматизации проектной деятельности как семантических компьютерных систем.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Golenkov V.V..tStand oICSaaK-2020art

:= *стандартное библиографическое описание**:

[V. Golenkov, V. Golovko и др., “The standardization of intelligent computer systems as a key challenge of the current stage of development of artificial intelligence technologies,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 4, BSUIR, Minsk, 2020, с. 73—88]

⇒ *аннотация**:

[Работа рассматривает вопросы стандартизации интеллектуальных компьютерных систем нового поколения]

⇐ *библиографическая ссылка**:

- *§ 7.9.2. Заключение. Основные направления, проблемы и перспективы развития интеллектуальных компьютерных систем нового поколения и соответствующей им технологии*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.2. Принципы, лежащие в основе смыслового представления информации*

Golenkov V.V..tStand oICSaaK-2018art

:= *стандартное библиографическое описание**:

[V. Golenkov, N. Guliakina, G. V. и др., “The standardization of intelligent computer systems as a key challenge of the current stage of development of artificial intelligence technologies,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 4, BSUIR, Minsk, 2018, с. 73—88]

⇒ *аннотация**:

[This work is devoted to the consideration of the most important factor providing semantic compatibility of intelligent computer systems and their components — standardization of intelligent computer systems, as well as standardization of methods and tools of their design.]

⇐ *библиографическая ссылка**:

- *Глава 7.2. Метасистема OSTIS*

Golenkov V.V.OntolBDoIS-2017art

:= *стандартное библиографическое описание**:

[V. V. Golenkov, “Ontology-based design of intelligent systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 1, Minsk : BSUIR, 2017, с. 37—56]

⇒ *аннотация**:

[Работа содержит описание онтологического проектирования интеллектуальных систем]

⇐ *библиографическая ссылка**:

- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

GooglAYOP-2022el

:= *стандартное библиографическое описание**:

[“Google Assistant, your own personal Google.” англ. Mode of access: `https://assistant.google.com/`. — Date of access: 15.09.2022. ()]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Goylo A..Means oFDoSaD-2022art

:= *стандартное библиографическое описание**:

[A. Goylo и S. Nikiforov, *Means of formal description of syntax and denotational semantics of various languages in next-generation intelligent computer systems*, англ. Минск, 2022, с. 99—118]

⇐ *библиографическая ссылка**:
- *Глава 2.1. Информационные конструкции и языки*
- *§ 2.1.1. Формализация понятия информационной конструкции*

Griffin D.W..MultiEV-1988art

:= *стандартное библиографическое описание**:
[D. W. Griffin и J. S. Lim, "Multiband excitation vocoder," *IEEE Transactions on acoustics, speech, and signal processing*, т. 36, № 8, с. 1223—1235, 1988]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Gungov A.tAmpliLiDtAoA-2018art

:= *стандартное библиографическое описание**:
[A. Gungov, "The ampliative leap in diagnostics: The advantages of abductive inference in clinical reasoning," англ., *History of Medicine*, т. 5, с. 233—242, янв. 2018. DOI: 10.3897/hmj.5.4.35635]

⇒ *аннотация**:
[Examining diagnostics in logical terms, attention is usually paid to the interaction between deductive and inductive reasoning. This article discusses Ch.S. Peirce's theory of abductive inference in the clinical diagnosis. The process of diagnostics is seen as a logical transition from the effect (patient's symptoms and signs) to the cause (the current health disorder), which is the direction common to abductive reasoning. For Peirce, abduction is performed through the transposition of the conclusion and the major premise in the categorical syllogism or, in his later writings, of the result and the rule. An emphasis is put on the ampliative leap from the premise (individual clinical signs and symptoms) to the conclusion (particular diagnosis) abduction features; the universal rule (the nosological unit) mediates between the individual clinical picture and the particular patient's diagnosis. The abductive inference draws on Kantian view on reflective judgment and G.B. Vico's ideas about imaginative universals. Reflective judgment aims at identifying a concept for some sensible data, whereas imaginative universals are not rational concepts but contain general characteristics like the regular concepts; formation of an imaginative universal resembles giving a diagnosis where an imagination drive inference is performed based on the combination of general elements of the relevant nosological unit and individual clinical signs and symptoms. Attention is also paid to the principles of coherence and teleology in performing an abductive inference in diagnostics as well as to the dual criterion of its truthfulness based both on coherence and correspondence. Examples from various medical fields are offered to illustrate the validity of the above logical claims in clinical practice.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Hadzic M..OntolBMAS-2009bk

:= *стандартное библиографическое описание**:
[M. Hadzic и др., *Ontology-based multi-agent systems*. Springer, 2009, с. 283]

⇐ *библиографическая ссылка**:
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Haegeman L.Intro tGaBT-1994bk

:= *стандартное библиографическое описание**:
[L. Haegeman, *Introduction to Government and Binding Theory* (Blackwell Textbooks in Linguistics), англ. Wiley, 1994, ISBN: 9780631190677. url: https://books.google.by/books?id=%5C_fvUInHLUTwC]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.1. Формализация синтаксиса естественных языков*

Hagan M..NeuraNfC-1999art

:= *стандартное библиографическое описание**:
[D. H. Hagan M.T., "Neural networks for control," в *Proceedings of the American Control Conference*, IEEE, 1999, с. 1642—1656]

⇒ *аннотация**:
[Provides a quick overview of neural networks and explains how they can be used in control systems. We introduce the multilayer perceptron neural network and describe how it can be used for function approximation. The backpropagation algorithm (including its variations) is the principal procedure for training multilayer perceptrons; it is briefly described here. Care must be taken, when training perceptron networks, to ensure that they do not overfit the training data and then fail to generalize well in new situations. Several techniques for improving generalization

are discussed. The article also presents several control architectures, such as model reference adaptive control, model predictive control, and internal model control, in which multilayer perceptron neural networks can be used as basic building blocks.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Hagoort P..SemanU-2009art

:= *стандартное библиографическое описание**:
[P. Hagoort, G. Baggio и R. M. Willems, “Semantic unification,” англ., в *The cognitive neurosciences, 4th ed.* MIT press, 2009, с. 819—836]

⇒ *аннотация**:
[Language and communication are about the exchange of meaning. A key feature of understanding and producing language is the construction of complex meaning from more elementary semantic building blocks. The functional characteristics of this semantic unification process are revealed by studies using event related brain potentials. These studies have found that word meaning is assembled into compound meaning in not more than 500 ms. World knowledge, information about the speaker, co-occurring visual input and discourse all have an immediate impact on semantic unification, and trigger similar electrophysiological responses as sentence-internal semantic information. Neuroimaging studies show that a network of brain areas, including the left inferior frontal gyrus, the left superior/middle temporal cortex, the left inferior parietal cortex and, to a lesser extent their right hemisphere homologues are recruited to perform semantic unification.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*

Halavataya K..3DRep oOiNGICS-2022art

:= *стандартное библиографическое описание**:
[K. Halavataya и A. Halavaty, “3D representation of objects in new generation intelligent computer systems,” англ., в *Open semantic technologies for intelligent systems*, Minsk : BSUIR, нояб. 2022, с. 251—260]

⇒ *аннотация**:
[Данная статья посвящена рассмотрению вопросов построения и использования трехмерного представления в различных задачах прикладных интеллектуальных систем, а также соответствующих систем позиционирования и ориентации в пространстве. Описание самого представления, а также принципов его построения осуществляется на основе базы знаний OSTIS-системы, что позволяет проводить глубокую интеграцию различных задач и методов, а также в последствии приводит к повышению степени конвергенции различных направлений.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции*

Halavataya K..AdjusV3DRRUTD-2022art

:= *стандартное библиографическое описание**:
[K. Halavataya, K. Kozadaev и V. Sadau, “Adjusting videoendoscopic 3D reconstruction results using tomographic data,” англ., *Computer Optics*, т. 46, № 2, с. 246—251, март 2022. DOI: 10.18287/2412-6179-co-910]

⇒ *аннотация**:
[В статье рассматривается подход к формированию трехмерного представления путем комбинирования первичной информации, полученной с помощью различных методов. Предлагается метод подстройки трехмерной модели объекта, полученной по видеоэндоскопическим данным, на основе данных компьютерной томографии.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.9.3. Действия по подбору и генерации алгоритма*

Halavataya K.LocalFDIfIMaODiRTA-2020art

:= *стандартное библиографическое описание**:
[K. Halavataya, “Local feature descriptor indexing for image matching and object detection in real-time applications,” англ., *Pattern Recognition and Information Processing*, т. 30, № 1, с. 16—21, 2020. DOI: 10.1134/S105466182001006X]

⇒ *аннотация**:
[Local feature extraction and description algorithms can be used to address a large variety of computer vision problems, including pattern recognition, frame stitching and 3D reconstruction. One of the most computationally expensive and time-consuming stages of any method based on local features is keypoint matching. This paper discusses possible

ways to optimize matching for frame alignment and object detection applications using Vantage-Point tree indexing to optimize pairwise keypoint matching, and image keypoint graph indexing to optimize pose estimation.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Hamilton Inter..Intero aKEftG-2006art

:= *стандартное библиографическое описание**:
[Hamilton и др., “Interoperability - A Key Element for the Grid and DER of the Future,” в *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, с. 927—931]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Harris C..aCombiCaED-1988art

:= *стандартное библиографическое описание**:
[C. Harris и M. Stephens, “A Combined Corner and Edge Detector,” в *Procedings of the Alvey Vision Conference 1988*, Alvey Vision Club, 1988. DOI: 10.5244/c.2.23]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Harris J.AlgebGaFC-1992bk

:= *стандартное библиографическое описание**:
[J. Harris, *Algebraic Geometry: A First Course*, англ. New York: Springer, 1992]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Harting R.L..UsingMAtRwMO-2008art

:= *стандартное библиографическое описание**:
[R. L. Hartung и A. Hakansson, “Using Meta-agents to Reason with Multiple Ontologies,” англ., в *Lecture Notes in Computer Science*, 2008, с. 261—270]

⇒ *аннотация**:
[В статье рассмотрен механизм метаагентов для управления агентами более низкого уровня.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

He K..DelviDiRSHP-2015art

:= *стандартное библиографическое описание**:
[K. He и др., *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*, 2015. arXiv: 1502.01852 [cs.CV]]

⇒ *аннотация**:
[This text discusses the use of rectified activation units, or rectifiers, in neural networks for image classification. The authors propose a Parametric Rectified Linear Unit (PReLU) that generalizes the traditional rectified unit and improves model fitting without added computational cost or overfitting risk. They also derive a robust initialization method that considers rectifier nonlinearities, enabling the training of extremely deep rectified models directly from scratch. The authors achieve a 4.94% top-5 test error on the ImageNet 2012 classification dataset with their PReLU networks, a 26% relative improvement over the previous ILSVRC 2014 winner, and the first to surpass human-level performance on this visual recognition challenge.]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

Heckmann D..tUserMaCOGUMO-2001art

≔ *стандартное библиографическое описание**:
[D. Heckmann и др., “The User Model and Context Ontology GUMO revisited for future Web 2.0 Extensions,” т. 298, янв. 2007]
⇒ *аннотация**:
[Revisit of the top-level ontology Gumo for the uniform management of user and context models in a semantic web environment. Paper discusses design decisions, while putting the focus on ontological issues]
⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Heim I..Seman iGG-1998bk
≔ *стандартное библиографическое описание**:
[I. Heim и A. Kratzer, *Semantics in Generative Grammar* (Blackwell Textbooks in Linguistics), англ. Wiley, 1998, ISBN: 9780631197133. url: `https://books.google.by/books?id=jAvR2DB3pPIC`]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.2. Формализация денотационной семантики естественных языков*

Hepp M.OntolSotABPaGC-2008art
≔ *стандартное библиографическое описание**:
[M. Hepp, “Ontologies: State of the Art, Business Potential, and Grand Challenges,” англ., в 2008, с. 3—22, ISBN: 978-0-387-69900-4. DOI: `10.1007/978-0-387-69900-4_1`. url: `https://doi.org/10.1007/978-0-387-69900-4_1`]
⇒ *аннотация**:
[В данной работе рассматривается понятие онтологии и как ее можно использовать. Автор обсуждает выразительность онтологии, количество элементов предметной области. Проблемы управления онтологиями в реальных приложениях и онтологиечкого проектирования.]
⇐ *библиографическая ссылка**:
- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Hervas R..aConteMBoOLaPfIV-2010art
≔ *стандартное библиографическое описание**:
[R. Hervas, J. Bravo и J. Fontecha, “A Context Model based on Ontological Languages: a Proposal for Information Visualization,” англ., *Journal of Universal Computer Science*, т. 16, № 12, с. 1539—1555, 2010]
⇒ *аннотация**:
[В работе рассматривается задача использования контекстно-семантических моделей для формирования трехмерных визуализааций данных из разнородных источников.]
⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.4. Семантическое представление объектов и сцены*

Hewitt C.MiddHoLPRPPatJFGP-2009el
≔ *стандартное библиографическое описание**:
[C. Hewitt. “Middle History of Logic Programming: Resolution, Planner, Prolog and the Japanese Fifth Generation Project.” англ. Mode of access: `http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=07A4074D9BBEC56C92085D21A10A6B4D?doi=10.1.1.363.8517&rep=rep1&type=pdf`. — Date of access: accessed 20.03.2020. ()]
⇒ *аннотация**:
[Middle History of Logic Programming: Resolution, Planner, Prolog and the Japanese Fifth Generation Project. ArXiv 2009.]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Hitz M..UsingAOftAGoUIfDBA-2016art
≔ *стандартное библиографическое описание**:
[M. Hitz и T. Kessel, “Using Application Ontologies for the Automatic Generation of User Interfaces for Dialog-Based Applications,” т. 268, дек. 2016, ISBN: 978-3-319-49943-7. DOI: `10.1007/978-3-319-49944-4_2`]

⇒ *аннотация**:
[The paper presents a data-centric, model driven approach for the automatic generation of user interfaces (UIs) for dialog-based applications using ontological descriptions. It focuses on Interview Applications, a common pattern e.g. for self-service applications in EIS. Existing approaches for the automatic UI generation usually rely on proprietary, UI-specific description models, designed and developed manually for the application in focus. The manual creation of the artefacts leads to a gap in the automated development, i.e. for dialog-based application UIs, where the structure and behavior is driven by the processed data. Furthermore, the UI specific nature of the artefacts impedes their (re-)use in different contexts. The presented approach is a shift away from a UI-specific towards a data-centric method of modelling dialog-based applications, bridging this gap. Application-Ontologies are used as description means, which leads to reusable, sharable model artifacts, applicable to different contexts of use]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Hoare C.A.CommuSP-1983art

:= *стандартное библиографическое описание**:
[C. A. R. Hoare, “Communicating Sequential Processes,” *Commun. ACM*, т. 26, № 1, с. 100—106, янв. 1983, ISSN: 0001-0782. DOI: 10.1145/357980.358021. url: https://doi.org/10.1145/357980.358021]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.4. Принципы синхронизации деятельности sc-агентов*

Hochreiter D.LongSTM-1997art

:= *стандартное библиографическое описание**:
[S. Hochreiter и J. Schmidhuber, “Long Short-term Memory,” *Neural computation*, т. 9, с. 1735—80, дек. 1997. DOI: 10.1162/neco.1997.9.8.1735]

⇒ *аннотация**:
[Learning to store information over extended time intervals by recurrent backpropagation takes a very long time, mostly because of insufficient, decaying error backflow. We briefly review Hochreiter’s (1991) analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called long short-term memory (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is O(1). Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*

Hopcroft ..Intro tATLaC-2000art

:= *стандартное библиографическое описание**:
[J. E. Hopcroft, R. Motwani и J. D. Ullman, *Introduction to automata theory, languages, and computation*, en, 2-е изд. Upper Saddle River, NJ: Pearson, нояб. 2000]

⇒ *аннотация**:
[Влиятельный учебник Джона Хопкрофта и Джеффри Ульмана по формальным языкам и теории вычислений, первое издание которого вышло в 1968 году. Раджив Мотвани участвовал в более поздних выпусках, начиная с 2000 г. В книге, в частности, описывается абстрактная машина Тьюринга.]

⇐ *библиографическая ссылка**:
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.2. Методы и средства реализации ostis-систем*

Horn B.DeterOF-1981art

:= *стандартное библиографическое описание**:
[B. K. Horn и B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, т. 17, с. 185—203, 1981. DOI: 10.1016/0004-3702(81)90024-2]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Hoy M.B.AlexaSCamItVA-2018art

:= *стандартное библиографическое описание**:

[M. B. Hoy, “Alexa, Siri, Cortana, and more: an introduction to voice assistants,” *Medical reference services quarterly*, т. 37, № 1, с. 81—88, 2018]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Hu Y..GraphAGLAoHBMEFPGAs-2021art

:= *стандартное библиографическое описание**:

[Y. Hu и др., “GraphLily: Accelerating Graph Linear Algebra on HBM-Equipped FPGAs,” *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, с. 1—9, 2021]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Hu Y.GeospS-2019art

:= *стандартное библиографическое описание**:

[Y. Hu, “Geospatial Semantics,” *Comprehensive Geographic Information Systems*, с. 80—94, 2018]

⇒ *аннотация**:

[Геопространственная семантика - это обширная область, охватывающая множество областей исследований. Термин «семантика» относится к значению вещей и отличается от термина «синтаксика». Соответственно, исследования геопространственной семантики обычно сосредоточены на понимании значения географических объектов, а также их аналогов в когнитивном и цифровом мире, таких как когнитивные географические концепции и цифровые справочники. Геопространственная семантика может облегчить проектирование географических информационных систем за счет повышения функциональной совместимости распределенных систем и разработки более интеллектуальных интерфейсов для взаимодействия с пользователем. В последние годы было проведено много исследований, подходящих к геопространственной семантике с разных точек зрения, с использованием различных методов и направленных на решение различных проблем. Между тем, появление больших геоданных, особенно большого количества неструктурированных текстовых данных в Интернете, и быстрая разработка методов обработки естественного языка открывают новые направления исследований в геопространственной семантике. Таким образом, эта статья предоставляет систематический обзор существующих геопространственных семантических исследований. Выявлены и обсуждены шесть основных областей исследований, включая семантическую совместимость, цифровые справочники, поиск географической информации, геопространственную семантическую сеть, семантику мест и когнитивно-географические концепции.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах*

Huang C.ed.FormalOaItSaW-2010bk

:= *стандартное библиографическое описание**:

[A. Pease и C. Fellbaum, “Formal ontology as interlingua: the SUMO and WordNet linking project and global WordNet,” англ., в *Ontology and the Lexicon: A Natural Language Processing Perspective* (Studies in Natural Language Processing), C.-r. Huang и др., ред., Studies in Natural Language Processing. Cambridge University Press, 2010, с. 25—35. DOI: 10.1017/CBO9780511676536.003]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Huang R..JointRLfTa3DPC-2023art

:= *стандартное библиографическое описание**:

[R. Huang и др., “Joint Representation Learning for Text and 3D Point Cloud,” англ., 2023. url: https://arxiv.org/pdf/2301.07584.pdf]

⇒ *аннотация**:

[В статье рассматривается подход к использованию обучения по представлениям для задачи семантической разметки трехмерного облака точек. Предложенная авторами модель пригодна для задачи семантической сегментации облака точек, отделения и детектирования объектов.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.4. Семантическое представление объектов и сцены*

Hussain J..ModelBAUIBoCaUEE-2018art

:= *стандартное библиографическое описание**:
[J. Hussain и др., “Model-based adaptive user interface based on context and user experience evaluation,” англ., *Journal on Multimodal User Interfaces*, т. 12, с. 17, февр. 2018. DOI: 10.1007/s12193-018-0258-2]

⇒ *аннотация**:
[Domain and device-independent model-based adaptive user interfacing methodology proposed. Methodology is dependent on the evaluation of user context and user experience (UX). The proposed methodology is implemented as an adaptive UI/UX authoring (A-UI/UX-A) tool; a system capable of adapting user interface based on the utilization of contextual factors, such as user disabilities, environmental factors (e.g. light level, noise level, and location) and device use, at runtime using the adaptation rules devised for rendering the adapted interface.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Ide N..WhatDIMA-2010art

:= *стандартное библиографическое описание**:
[N. Ide и J. Pustejovsky, “What Does Interoperability Mean , Anyway ? Toward an Operational Definition of Interoperability for Language Technology,” 2010]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

IECAVaRE-2015el

:= *стандартное библиографическое описание**:
[“IEC 62087-2:2015. Audio, video, and related equipment — Determination of power consumption — Part 2: Signals and media,” International Electrotechnical Commission, Standard, 2015]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

IEEESfSGAC-2020el

:= *стандартное библиографическое описание**:
[“IEEE 1857.8-2020 — IEEE Standard for Second Generation Audio Coding,” Institute of Electrical и Electronics Engineers, Standard, 2020]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Iliadis A.T.Tower oBPMDM-2019art

:= *стандартное библиографическое описание**:
[A. Iliadis, “The Tower of Babel Problem: Making Data Make Sense with Basic Formal Ontology,” *Online Information Review*, т. 43, № 6, с. 1021—1045, 2019]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Пункт 7.1.1.1. Общая оценка современного состояния человеческой деятельности в области Искусственного интеллекта*

IMS-IMS-2023el

:= *стандартное библиографическое описание**:
[*IMS — Instructional Management System (Системы организации обучения)*, https://www.1edtech.org/, accessed 19-Mar-2023, 2023]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Iqbal R..Analy oEM-2013art

:= *стандартное библиографическое описание**:

[R. Iqbal, A. Murad и A. Mustapha, “An analysis of ontology engineering methodologies : a lit. rev.,” англ., *Research J. of Appl. Sciences, Engineering a. Technology*, т. 6, № 16, с. 48—62, 2013]

⇒ *аннотация**:

[В настоящее время широко распространено мнение, что онтология играет решающую роль в достижении цели для машины сети, также известной как семантическая сеть.]

⇐ *библиографическая ссылка**:

- *Глава 2.4. Представление формальных онтологий базовых классов сущностей в ostis-системах*

ISA51-2022el

:= *стандартное библиографическое описание**:

[*ISA5.1 Standard*, `https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa5-1/`, (accessed 2022, October)]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

ISA88BC-2022el

:= *стандартное библиографическое описание**:

[*ISA88: Batch Control [Electronic resource]*, англ., International Society of Automation, Mode of access: `https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa88`. — Date of access: 08.10.2022]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*
- *Подпункт 7.7.2.2.4 Формализация стандарта ISA-88*

ISA95-2022el

:= *стандартное библиографическое описание**:

[*ISA-95 standard*, `https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95/`, (accessed 2022, October)]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

ISOEoHSIPGoVUIE-2016el

:= *стандартное библиографическое описание**:

[“ISO 9241-161:2016 Ergonomics of human-system interaction — Part 161: Guidance on visual user-interface elements,” International Organization for Standardization, Geneva, CH, Standard, 2016, с. 69]

⇐ *библиографическая ссылка**:

- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

ISOIECITCoAVO-2005art

:= *стандартное библиографическое описание**:

[“ISO/IEC 14496-3:2005. Information technology — Coding of audio-visual objects — Part 3: Audio,” International Organization for Standardization, Geneva, CH, Standard, 2005]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

ISOIECITMAT-2020art

:= *стандартное библиографическое описание**:

[“ISO/IEC 23003-3:2020 Information technology — MPEG audio technologies — Part 3: Unified speech and audio coding,” International Organization for Standardization, Geneva, CH, Standard, 2020]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

ISOIECITTLOPR-2021el

:= *стандартное библиографическое описание**:

[“ISO/IEC 21838-1:2021 Information technology — Top-level ontologies (TLO) — Part 1: Requirements,” International Organization for Standardization, Geneva, CH, Standard, 2021]

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *§ 2.5.6. Онтологии верхнего уровня*

ISOTREoHSI-2002el

:= *стандартное библиографическое описание**:
[“ISO/TR 16982:2002 Ergonomics of human-system interaction — Usability methods supporting human-centred design,” International Organization for Standardization, Geneva, CH, Standard, 2002, с. 44]

⇐ *библиографическая ссылка**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

ITAPKIT-2012el

:= *стандартное библиографическое описание**:
[*IT/APKIT Professional Standards [Electronic Resource]*, Available at: `http://www.apkit.webtm.ru/committees/education/meetings/standarts.php`, (accessed 7.05.2012)]

⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*

Ivaniuk D.NeuroPIDCfaP-2013art

:= *стандартное библиографическое описание**:
[D. Ivaniuk, “Neuro-PID Controller for a Pasteurizer,” в *XV International PhD Workshop OWD*, 2013]

⇒ *аннотация**:
[The neuro-PID controller for the pasteurizer was developed. It consists of two parts: the conventional PID (proportional plus integral plus derivative controller) and the neural network, which is based on the multilayer perceptron structure. The outputs of the neural network are proportional (P), integral (I), and derivative (D) gains. The simulation and experimental results show the effectiveness of the proposed approach.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Ivashenko V.P.Appli oaIPfOMBPSUaUSKR-2021art

:= *стандартное библиографическое описание**:
[V. Ivashenko, “Application of an integration platform for ontological model-based problem solving using an unified semantic knowledge representation,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 5, Minsk : BSUIR, 2022, с. 179—186]

⇒ *аннотация**:
[В статье рассматривается решение в виде интеллектуальной интеграционной платформы, основанной на модели унифицированного семантического представления знаний, для разработки многоагентных систем, управляемых знаниями. В работе применяются: модель унифицированного семантического представления знаний на основе семантических сетей, модели и методы теории меры и теории вероятностей, методы дискретной оптимизации и прикладной математики, компьютерное моделирование и многоагентный подход. Работа направлена на разработку компьютерных средств с когнитивной архитектурой, использующих элементы искусственного сознания, способствующие гибкому взаимодействию и адаптации этих средств в сложных образовательных приложениях. Были разработаны и реализованы виртуальные машины, другие подсистемы интеграционной платформы, а также - справочно-проверяющая прикладная многоагентная система, функционирующие в рамках системы человеко-машинного взаимодействия.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ivashenko V.P.GenerPSRLaSS-2022art

:= *стандартное библиографическое описание**:
[V. Ivashenko, “General-purpose semantic representation language and semantic space,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 6, Minsk : BSUIR, 2022, с. 41—64]

⇒ *аннотация**:
[Статья рассматривает модели и средства, обеспечивающие унифицированное представление знаний и их интеграцию в рамках «смыслового пространства». Для этого вводится понятие «обобщенного формального языка», позволяющего выявить с целью анализа взаимоотношение формальных языков и известных язы-

ков представления знаний, включая семантические сети. На основе этого анализа уточняется семантика языков модели унифицированного представления знаний, вводится язык, являющийся основой стандарта для технологии разработки интеллектуальных систем, и дается концепция «смыслового пространства», ориентированного использование в целях оценки качества интеллектуальных компьютерных систем в рамках технологии OSTIS. Рассматриваются прикладные задачи на основе предложенных моделей и дальнейшие перспективы развития технологии и ее компонентов.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Ivashenko V.P..SemanLoREiaFBTM-2021art

:= *стандартное библиографическое описание**:
[O. M. Ivashenko V. Zotov N., “Semantic Logging of Repeating Events in a Forward Branching Time Model,” англ., в *Pattern Recognition and Information Processing (PRIP’2021) = Распознавание образов и обработка информации (2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus*, United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Минск, 2021, с. 149—152]

⇒ *аннотация**:
[The tasks of knowledge logging in the form of semantic networks of the model of the unified semantic knowledge representation are considered. The formal model of a semantic log of repeating events in knowledge processing and algorithms for adding and retrieving logged events from the log are presented. The spatial-time structure of logged processes should satisfy a forward branching time model.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Ivashenko V.P.StrinPMfKDS-2020art

:= *стандартное библиографическое описание**:
[V. Ivashenko, “String processing model for knowledge-driven systems,” *Doklady BGUIR*, т. 18, с. 33—40, окт. 2020. DOI: 10.35596/1729-7648-2020-18-6-33-40]

⇒ *аннотация**:
[The purpose of the work is to confirm experimentally theoretical estimates for time complexity of operations of the string processing model linked with the metric space for solving data processing problems in knowledge-driven systems including the research and comparison of the operation characteristics of these operations with the characteristics of similar operations for the most relevant data structures. Integral and unit testing were used to obtain the results of the performed computational experiments and verify their correctness. The C C++ implementation of operations of the string processing model was tested. The paper gives definitions of concepts necessary for the calculation of metric features calculated over strings. As a result of the experiments, theoretical estimates of the computational complexity of the implemented operations and the validity of the choice of parameters of the used data structures were confirmed, which ensures near-optimal throughput and operation time indicators of operations. According to the obtained results, the advantage is the ability to guarantee the time complexity of the string processing operations no higher than O at all stages of a life cycle of data structures used to represent strings, from their creation to destruction, which allows for high throughput in data processing and responsiveness of systems built on the basis of the implemented operations. In case of solving particular string processing problems and using more suitable for these cases data structures such as vector or map the implemented operations have disadvantages meaning they are inferior in terms of the amount of data processed per time unit. The string processing model is focused on the application in knowledge-driven systems at the data management level.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Ivory M..tState otAiAUEoUI-2001art

:= *стандартное библиографическое описание**:
[M. Ivory и M. Hearst, “The State of the Art in Automating Usability Evaluation of User Interfaces,” англ., *ACM Comput. Surv.*, т. 33, с. 470—, янв. 2001]

⇒ *аннотация**:

[The article discusses the state of the art in usability evaluation automation, and highlight the approaches that merit further investigation]

⇐ *библиографическая ссылка**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Iyengar A.CompoDfRD-2021art

:= *стандартное библиографическое описание**:
[A. Iyengar, “Component Design for Relational Databases,” англ., в *Data Management*, Auerbach Publications, 2021, с. 143—156. DOI: 10.1201/9780429114878-15. url: https://doi.org/10.1201/9780429114878-15]

⇒ *аннотация**:
[В этой работе исследуются преимущества компонентного проектирования для управления данными и предлагаются стратегии, которые могут быть использованы при использовании реляционных централизованных баз данных с объектно-ориентированными компонентными архитектурами приложений. Данная работа отвечает на такие вопросы как "Как объектные технологии и бизнес-компоненты используются при управлении данных? "Как модели данных связаны с объектными или компонентными моделями?"и другие.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *5.1. Введение в Главу 5.1.*

Jackendoff R..XS aSoPS-1977bk

:= *стандартное библиографическое описание**:
[R. Jackendoff и R. Jackendoff, *X Syntax: A Study of Phrase Structure* (Linguistic inquiry monographs), англ. MIT Press, 1977, ISBN: 9780262100182. url: https://books.google.by/books?id=ALf6PgAACAAJ]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *§ 4.2.2. Понимание естественно-языковых сообщений, входящих в ostis-систему*
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.1. Формализация синтаксиса естественных языков*

Jackson P.Intro tES-1998bk

:= *стандартное библиографическое описание**:
[P. Jackson, *Introduction to expert systems*, англ., 3rd. Boston: Addison-Wesley, 1998]

⇒ *аннотация**:
[Третье издание книги Питера Джексона «Введение в экспертные системы» обновляет технологическую базу исследования экспертных систем и включает эти результаты в контекст самых разных областей применения. В более ранних главах используется более практичный подход к основным темам, чем в предыдущих изданиях, а в более поздних главах представлены новые тематические области, такие как рассуждения на основе прецедентов, коннекционистские системы и гибридные системы. Результаты в смежных областях, таких как машинное обучение и рассуждения с неопределенностью, также подвергаются тщательной обработке.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Jagannathan V..BlackAaA-1989bk

:= *стандартное библиографическое описание**:
[V. Jagannathan, K. Dodhiawala и L. Baum, *Blackboard architectures and applications*, англ. New York: Academic Press, 1989]

⇒ *аннотация**:
[В книге подробно рассматривается принцип "доски объявлений" для взаимодействия агентов.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Janowicz K..GeospSaLSD-2012art

:= *стандартное библиографическое описание**:
[K. Janowicz и др., “Geospatial semantics and linked spatiotemporal data –Past, present, and future,” *Semantic Web*, т. 3, с. 321—332, окт. 2012. DOI: 10.3233/SW-2012-0077]

⇒ *аннотация**:
[Геонауки и география - это не просто еще одна область применения семантических технологий. Огромная неоднородность участвующих дисциплин, начиная от естественных наук и социальных наук, ставит новые задачи с точки зрения взаимодействия. Кроме того, внутренние пространственные и временные информационные компоненты также требуют определенных семантических подходов. По этим причинам геопространственная семантика, гео-онтологии и семантическая совместимость были активными областями исследований в течение последних 20 лет. Сообщество геопространственной семантики было одним из первых последователей Семантической сети, предлагая методы, онтологии, варианты использования и наборы данных. Сегодня географическая информация является важной частью многих центральных узлов в Сети связанных данных. В этой редакционной статье мы описываем область исследования геопространственной семантики, выделяем основные направления и тенденции исследований и рассматриваем будущие задачи. Мы надеемся, что этот текст будет полезен для геофизиков, заинтересованных в семантических исследованиях, а также для инженеров, заинтересованных в пространственно-временных данных.]
⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*
- *§ 7.8.3. Основные формальные онтологии баз знаний в интеллектуальных геоинформационных ostis-системах*

JAVAADFE-el
:= *стандартное библиографическое описание**:
[*JAVA Agent DEvelopment Framework [Electronic resource]*, англ., Режим доступа: `http://jade.tilab.com/`. — Дата доступа: 18.01.2023]
⇒ *аннотация**:
[Официальный сайт проекта JADE.]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Jeffries R..UserIEitRW-1991art
:= *стандартное библиографическое описание**:
[R. Jeffries и др., “User Interface Evaluation in the Real World: A Comparison of Four Techniques,” в *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, сер. CHI ’91, New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, с. 119—124, ISBN: 0897913833. DOI: `10.1145/108844.108862`. url: `https://doi.org/10.1145/108844.108862`]
⇒ *аннотация**:
[The relative advantages of all the techniques of user interface evaluation are discussed, and suggestions for improvements in the techniques are offered.]
⇐ *библиографическая ссылка**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Jeni P..CanDPAP-2022art
:= *стандартное библиографическое описание**:
[J. Paay и др., “Can digital personal assistants persuade people to exercise?” *Behaviour & Information Technology*, т. 41, № 2, с. 416—432, 2022. DOI: `10.1080/0144929X.2020.1814412`]
⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

Ji M..aShortTSCM-2022art
:= *стандартное библиографическое описание**:
[M. Ji и X. Zhang, “A Short Text Similarity Calculation Method Combining Semantic and Headword Attention Mechanism,” англ., *Scientific Programming*, т. 2022, 2022]
⇒ *аннотация**:
[В работе рассмотрен подход к вычислению семантического подобия коротких текстов на основе семантики]
⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.2 Автоматическая проверка ответов пользователей*

Kamilaris A..GeospAatIoT-2018art

:= *стандартное библиографическое описание**:

[A. Kamilaris и F. O. Ostermann, “Geospatial analysis and the internet of things,” *ISPRS international journal of geo-information*, т. 7, № 7, с. 269, 2018, Publisher: Multidisciplinary Digital Publishing Institute]

⇒ *аннотация**:

[As the Internet of Things (IoT) penetrates our everyday lives, being used to address a wide variety of real-life challenges and problems, the location of things becomes an important parameter. The exact location of measuring the physical world through IoT is highly relevant to understand local environmental conditions, or to develop powerful, personalized and context-aware location-based services and applications. This survey paper maps and analyzes the IoT based on its location dimension, categorizing IoT applications and projects according to the geospatial analytical methods performed. The survey investigates the opportunities of location-aware IoT, and examines the potential of geospatial analysis in this research area.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Kawahara H..Devel oERTB-2009art

:= *стандартное библиографическое описание**:

[H. Kawahara и др., “Development of exploratory research tools based on TANDEM-STRAIGHT,” в *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, Asia-Pacific Signal и Information Processing Association, 2009, 2009, с. 111—120]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Kawahara H.Explo otOAoVR-2010art

:= *стандартное библиографическое описание**:

[H. Kawahara, “Exploration of the other aspect of vocoder revisited: AZ STRAIGHT, TANDEM-STRAIGHT and morphing,” в *Seventh ISCA Workshop on Speech Synthesis*, 2010]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Kerr C..aConceMfTI-2006art

:= *стандартное библиографическое описание**:

[C. I. Kerr и др., “A conceptual model for technology intelligence,” *International Journal of Technology Intelligence and Planning*, т. 2, № 1, с. 73—93, 2006]

⇐ *библиографическая ссылка**:

- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

Khurana D..NaturLPSotA-2022art

:= *стандартное библиографическое описание**:

[D. Khurana, A. Koli и K. Khatter, “Natural language processing: state of the art, current trends and challenges,” англ., в *Multimed Tools Appl*, 2022. DOI: `https://doi.org/10.1007/s11042-022-13428-4`]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Kingma D..Adam aMfSO-2014art

:= *стандартное библиографическое описание**:

[D. Kingma и J. Ba, “Adam: A Method for Stochastic Optimization,” англ., 2014. url: `https://arxiv.org/pdf/1412.6980.pdf`]

⇒ *аннотация**:

[The text introduces Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions based on adaptive estimates of lower-order moments. Adam is computationally efficient, has low memory requirements, is invariant to diagonal rescaling of the gradients, and is suitable for problems that are large in terms of data and/or parameters. The algorithm is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyperparameters require little tuning and have intuitive interpretations. The authors discuss connections to related algorithms, analyze the theoretical convergence properties of the algorithm, and provide a regret bound on the convergence rate that is comparable to the best-known results. Empirical results

demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. The text also discusses AdaMax, a variant of Adam based on the infinity norm]

⇐ *библиографическая ссылка**:

- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *§ 3.6.2. Логико-семантическая модель ostis-системы автоматизации проектирования искусственных нейронных сетей, семантически совместимых с базами знаний ostis-систем*

Kirrane S..PrivaSaPaRoP-2018art

:= *стандартное библиографическое описание**:

[S. Kirrane, S. Villata и M. dAquin, “Privacy, security and policies: A review of problems and solutions with semantic web technologies,” *Semantic Web*, т. 9, № 2, с. 153—161, 2018]

⇐ *библиографическая ссылка**:

- *Глава 7.3. Структура Экосистемы OSTIS*

KnowlBEST-el

:= *стандартное библиографическое описание**:

[*Knowledge Base Editor - specialized tool for knowledge base creation. [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/kbe`. — Date of access: 20.03.2023]

⇐ *библиографическая ссылка**:

- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.2. Индивидуальный аспект проектирования и разработки баз знаний ostis-систем*

KnowlIFS-2023el

:= *стандартное библиографическое описание**:

[“Knowledge Interchange Format Specification [Electronic resource],” англ., *Stanford Logic Group*, Mode of access: `http://logic.stanford.edu/kif/specification.html`. — Date of access: 18.01.2023]

⇒ *аннотация**:

[Официальный сайт, описывающий спецификацию языка KIF.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Kong J..Desig oHCAM-2011art

:= *стандартное библиографическое описание**:

[J. Kong и др., “Design of human-centric adaptive multimodal interfaces,” англ., *International Journal of Human-Computer Studies*, т. 69, с. 854—869, дек. 2011. DOI: `10.1016/j.ijhcs.2011.07.006`]

⇒ *аннотация**:

[Multimodal interfaces have attracted more and more attention. Most researches focus on each interaction mode independently and then fuse information at the application level. Recently, several frameworks and models have been proposed to support the design and development of multimodal interfaces. However, it is challenging to provide automatic modality adaptation in multimodal interfaces. Existing approaches are using rule-based specifications to define the adaptation of input/output modalities. Rule-based specifications have the problems of completeness and coherence. Distinct from previous work, this paper presents a novel approach that quantifies the user preference of each modality and considers the adaptation as an optimization issue that searches for a set of input/output modalities matching user’s preference. Our approach applies a cross-layer design, which considers the adaptation from the perspectives of the interaction context, available system resources, and QoS requirements. Furthermore, our approach supports human-centric adaptation. A user can report the preference of a modality so that selected modalities fit user’s personal needs. An optimal solution and a heuristic algorithm have been developed to automatically select an appropriate set of modality combinations under a specific situation. We have designed a framework based on the heuristic algorithm and existing ontology, and applied the framework to conduct a utility evaluation, in which we have employed a within-subject experiment. Fifty participants were invited to go through three scenarios and compare automatically selected modalities with randomly selected modalities. The results from the experiment show that users perceived the automatically selected modalities as appropriate and satisfactory.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Kontakis K..DECOaOFaI3DI-2014art

:= *стандартное библиографическое описание**:
[K. Kontakis и др., “DEC-O: an ontology framework and interactive 3D interface for interior decoration applications in the web,” в *Proceedings of the 19th International ACM Conference on 3D Web Technologies*, ACM, 2014. DOI: 10.1145/2628588.2628596]
⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.2. Анализ существующих подходов к трехмерной реконструкции*

Kostareva T..UsingODMtD-2016art
:= *стандартное библиографическое описание**:
[T. Kostareva, S. Chuprina и A. Nam, “Using Ontology-Driven Methods to Develop Frameworks for Tackling NLP Problems,” в *AIST*, 2016]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Kostrikin A..LineaAaG-1997bk
:= *стандартное библиографическое описание**:
[Y. M. A. Kostrikin, *Linear Algebra and Geometry*, англ. Gordon и Breach Science Publishers, 1997]
⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Kovalev M.V.Conve aIoANN-2022art
:= *стандартное библиографическое описание**:
[M. V. Kovalev, “Convergence and integration of artificial neural networks with knowledge bases in next-generation intelligent computer systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 6, Minsk : BSUIR, 2022, с. 173—186]
⇒ *аннотация**:
[In the article, an approach to the integration and convergence of artificial neural networks with knowledge bases in next-generation intelligent computer systems through the representation and interpretation of artificial neural networks in a knowledge base is considered. The syntax, denotational, and operational semantics of the language for representing neural network methods in knowledge bases are described. The stages of building of neural network problem-solving methods with the help of intelligent framework for designing artificial neural networks are described.]
⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*
- *Пункт 3.6.1.2. Операционная семантика моделей искусственных нейронных сетей, используемых в ostis-системах*

Kowalski R.PrediLaPL-1974art
:= *стандартное библиографическое описание**:
[R. Kowalski, “Predicate logic as programming language,” англ., в *IFIP congress*, т. 74, 1974, с. 569—544]
⇒ *аннотация**:
[Работа содержит описание логики предикатов как языка программирования]
⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Krig S.CompuVM-2016bk
:= *стандартное библиографическое описание**:
[S. Krig, *Computer Vision Metrics*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-33762-3]
⇒ *аннотация**:
[Книга посвящена методам квантификации и признакового описания информации, представленной в виде изображений. Рассматриваемые численные и признаковые описания находят широкое применение в задаче трехмерной реконструкции и компьютерного зрения в целом.]
⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Krom M.tDecisPfFiPC-1970art

:= *стандартное библиографическое описание**:
[M. R. Krom, “The decision problem for formulas in prenex conjunctive normal form with binary disjunctions,” англ., *The Journal of Symbolic Logic*, т. 35, № 2, с. 210—216, 1970]

⇒ *аннотация**:
[Работа содержит описание логических операций над формулами логики предикатов]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Kroshchanka A..aNeuraSAtCV-2022art

:= *стандартное библиографическое описание**:
[A. Kroshchanka и др., “A Neural-Symbolic Approach to Computer Vision,” в *Open Semantic Technologies for Intelligent Systems*, V. Golenkov и др., ред., Cham: Springer International Publishing, 2022, с. 282—309]

⇒ *аннотация**:
[The paper presents a general computer vision model based on neural-symbolic artificial intelligence capable of performing semantic analysis of a video stream. The proposed approach integrates artificial neural networks (ANN) with the knowledge base on the basis of an ontological approach. In this case, the knowledge base interacts with the neural networks as with agents and the results of their functioning are used for further semantic analysis. The problems of object detection and recognition of images, as well as emotions, are considered as tasks of computer vision. The features, advantages and prospects of using this model are described. Its implementation is considered on the example of an intelligent module that includes the FaceNet and eXnet neural network models for face identification and emotion recognition in the conversational modeling system]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Kwok C.CT..ScaliQAttW-2001art

:= *стандартное библиографическое описание**:
[C. C. Kwok, O. Etzioni и D. S. Weld, “Scaling question answering to the web,” в *Proceedings of the 10th international conference on World Wide Web*, 2001, с. 150—161]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Laird J.E..Claim aCiEHLIS-2009art

:= *стандартное библиографическое описание**:
[J. E. Laird и др., “Claims and challenges in evaluating human-level intelligent systems,” в *2nd Conference on Artificiel General Intelligence (2009)*, Atlantis Press, 2009, с. 80—85]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы*

Lando P..aOntollitFoCP-2007art

:= *стандартное библиографическое описание**:
[P. Lando и др., “An ontological investigation in the field of computer programs,” в *Software and Data Technologies*, Springer, 2007, с. 371—383]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.2. Существующие онтологии языков программирования*

Lando P..Towar aGOoCP-2007art

:= *стандартное библиографическое описание**:
[P. Lando и др., “TOWARDS A GENERAL ONTOLOGY OF COMPUTER PROGRAMS,” в *International Conference on Software and Data Technologies*, SCITEPRESS, т. 2, 2007, с. 163—170]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.2. Существующие онтологии языков программирования*

Lanzenberger M..MakinOTKIitSW-2008art

:= *стандартное библиографическое описание**:

[Lanzenberger, Monika and Sampson, Jennifer and Kargl, Horst and Wimmer, Manuel and Conroy, Colm and O'Sullivan, Declan and Lewis, David and Brennan, Rob and Ramos-Gargantilla, José Ángel and Gómez-Pérez, Asunción and Fürst, Frédéric and Trichet, Francky and Euzenat, Jérôme and Polleres, Axel and Scharffe, François and Kotis, Konstantinos, "Making Ontologies Talk: Knowledge Interoperability in the Semantic Web," *IEEE Intelligent Systems*, т. 23, № 6, с. 72—85, 2008]

⇐ *библиографическая ссылка**:

- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*

LargeLMaNML-2021el

:= *стандартное библиографическое описание**:

[“Large Language Models: A New Moore’s Law?” Англ. Mode of access: `https://huggingface.co/blog/large-language-models`. — Date of access: 15.10.2022. (2021)]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Laroche J..HNSSMBoaHNM-1993art

:= *стандартное библиографическое описание**:

[J. Laroche, Y. Stylianou и E. Moulines, “HNS: Speech modification based on a harmonic+ noise model,” в *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, т. 2, 1993, с. 550—553]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Lawan A.. tSemanWRLEES-2019art

:= *стандартное библиографическое описание**:

[L. Abba и R. Abdur, “The Semantic Web Rule Language Expressiveness Extensions-A Survey,” англ., 2019]

⇒ *аннотация**:

[Semantic Web Rule Language (SWRL) является прямым расширением OWL 2 DL с подмножеством RuleML и предназначен для использования в качестве языка правил Semantic Web. В данной статье исследуется современное состояние расширений выразительности SWRL, предложенных с течением времени. В качестве мотивации обсуждается эффективность комбинации SWRL/OWL в моделировании фактов предметной области, а также выделяются некоторые общие выразительные ограничения комбинации. Затем в документе классифицируются и представляются соответствующие языковые расширения SWRL и их дополнительные выразительные возможности для исходного определения SWRL. Кроме того, он обеспечивает сравнительный анализ синтаксиса и семантики предлагаемых расширений. В заключение оцениваются требования разрешимости и удобство использования каждого расширения выразительности для эффективного включения в онтологии OWL.]

⇐ *библиографическая ссылка**:

- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

LegUpHLS-2023el

:= *стандартное библиографическое описание**:

[*LegUp High-Level Synthesis*, English, Mode of access: `http://legup.eecg.utoronto.ca/`. — Date of access: 29.03.2023]

⇒ *аннотация**:

[Официальный сайт проекта LegUp – инструмента с открытым исходным кодом высокоуровневого синтеза. Платформа LegUp позволяет исследователям улучшить синтез C в Verilog, не создавая инфраструктуру с нуля.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*

Lehmann J..DBpedaLSMKBEfW-2015art

:= *стандартное библиографическое описание**:

[J. Lehmann и др., “DBpedia – A large-scale, Multilingual Knowledge Base Extracted from Wikipedia,” *Semantic Web*, т. 6, № 2, с. 167—195, 2015. DOI: 10.3233/sw-140134. url: https://doi.org/10.3233/sw-140134]

⇒ *аннотация**:

[The DBpedia community project extracts structured, multilingual knowledge from Wikipedia and makes it freely available on the Web using Semantic Web and Linked Data technologies. The project extracts knowledge from 111 different language editions of Wikipedia. The largest DBpedia knowledge base which is extracted from the English edition of Wikipedia consists of over 400 million facts that describe 3.7 million things. The DBpedia knowledge bases that are extracted from the other 110 Wikipedia editions together consist of 1.46 billion facts and describe 10 million additional things. The DBpedia project maps Wikipedia infoboxes from 27 different language editions to a single shared ontology consisting of 320 classes and 1,650 properties. The mappings are created via a world-wide crowd-sourcing effort and enable knowledge from the different Wikipedia editions to be combined. The project publishes releases of all DBpedia knowledge bases for download and provides SPARQL query access to 14 out of the 111 language editions via a global network of local DBpedia chapters. In addition to the regular releases, the project maintains a live knowledge base which is updated whenever a page in Wikipedia changes. DBpedia sets 27 million RDF links pointing into over 30 external data sources and thus enables data from these sources to be used together with DBpedia data. Several hundred data sets on the Web publish RDF links pointing to DBpedia themselves and make DBpedia one of the central interlinking hubs in the Linked Open Data (LOD) cloud. In this system report, we give an overview of the DBpedia community project, including its architecture, technical implementation, maintenance, internationalisation, usage statistics and applications.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Lemley J..DeepLfCDaSP-2017art

:= *стандартное библиографическое описание**:

[J. Lemley, S. Bazrafkan и P. Corcoran, “Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision.,” *IEEE Consumer Electronics Magazine*, т. 6, № 2, с. 48—56, 2017]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Lenat D.B..CycTPwCS-1990art

:= *стандартное библиографическое описание**:

[D. B. Lenat и др., “Cyc: toward programs with common sense,” *Communications of the ACM*, т. 33, № 8, с. 30—49, 1990]

⇒ *аннотация**:

[Cyc is a bold attempt to assemble a massive knowledge base (on the order of 108 axioms) spanning human consensus knowledge. This article examines the need for such an undertaking and reviews the authos’ efforts over the past five years to begin its construction. The methodology and history of the project are briefly discussed, followed by a more developed treatment of the current state of the representation language used (epistemological level), techniques for efficient inferencing and default reasoning (heuristic level), and the content and organization of the knowledge base.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

Lenat D.B.Cyc aLSIiKI-1995art

:= *стандартное библиографическое описание**:

[D. B. Lenat, “Cyc: A large-scale investment in knowledge infrastructure,” *Communications of the ACM*, т. 38, № 11, с. 33—38, 1995]

⇒ *аннотация**:

[Since 1984, a person-century of effort has gone into building CYC, a universal schema of roughly 105 general concepts spanning human reality. Most of the time has been spent codifying knowledge about these concepts; approximately 106 commonsense axioms have been handcrafted for and entered into CYC’s knowledge base, and millions more have been inferred and cached by CYC. This article examines the fundamental assumptions of doing such a large-scale project, reviews the technical lessons learned by the developers, and surveys the range of applications that are or soon will be enabled by the technology.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Leutenegger S..BRISKBRISK-2011art

:= *стандартное библиографическое описание**:
[S. Leutenegger, M. Chli и R. Siegwart, “BRISK: Binary Robust invariant scalable keypoints,” в *2011 International Conference on Computer Vision*, IEEE, 2011. DOI: 10.1109/iccv.2011.6126542]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Li H.Resea oIAGBoD-2012art

:= *стандартное библиографическое описание**:
[H. Li, “Research on item automatic generation based on DL and domain ontology,” *Journal of Changchun University of Technology (Natural Science Edition)*, т. 33, № 4, с. 461—464, авг. 2012]

⇒ *аннотация**:
[Работа содержит описание вопросов генерации с использованием DL]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.1 Автоматическая генерация тестовых вопросов*

Li W..Devel oaPSfAAV-2021art

:= *стандартное библиографическое описание**:
[W. Li и L. Qian, “Development of a problem solver for automatic answer verification in the intelligent tutoring systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 5, Minsk : BSUIR, 2021, с. 169—178]

⇒ *аннотация**:
[Работа содержит описание разработки решателя задач для проверки ответов пользователей]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Пункт 7.5.3.2. Предлагаемый подход к автоматизации контроля знаний*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*

Li W..DigitECaP-2012art

:= *стандартное библиографическое описание**:
[W. Li, Y. Badr и F. Biennier, “Digital ecosystems: challenges and prospects,” с. 117—122, 2012]

⇒ *аннотация**:
[Обзор концепции цифровой экосистемы и перечень проблем в предметной области цифровых экосистем]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*

Li W..OntolAfQGaKC-2020art

:= *стандартное библиографическое описание**:
[W. Li, N. Grakova и L. Qian, “Ontological Approach for Question Generation and Knowledge Control,” в *Communications in Computer and Information Science*, Springer International Publishing, 2020, с. 161—175. DOI: 10.1007/978-3-030-60447-9_10. url: https://doi.org/10.1007/978-3-030-60447-9_10]

⇒ *аннотация**:
[Подход к генерации тестовых вопросов на основе онтологии представлен в работе]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.2.1 Предлагаемый подход к автоматической генерации тестовых вопросов*

Li X.Reali oASAfSQ-2009art

:= *стандартное библиографическое описание**:
[X. Li, “Realization of automatic scoring algorithm for subjective questions based on artificial intelligence,” *Journal of Jiangnan University (Natural Science Edition)*, т. 8, № 3, с. 292—295, 2009]

⇒ *аннотация**:
[Работа содержит описание алгоритма автоматической оценки субъективных вопросов]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Limbourg Q..UsiXM aUIDLSMLoI-2004art

:= *стандартное библиографическое описание**:
[Q. Limbourg и J. Vanderdonckt, “UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence,” янв. 2004, с. 325—338]

⇒ *аннотация**:
[USer Interface eXtensible Markup Language (UsiXML) consists of a User Interface Description Language (UIDL) allowing designers to specify a user interface at mul- tiple levels of abstraction depending on the development path they are following: task and concepts, abstract user interface, concrete user interface, and final user in- terface. These levels support to some extent independence with respect to device, computing platform, modality of interaction, channel of information, and context of use. A single user interface can be specified and produced at and from different, possibly multiple, levels of abstraction while maintaining the mappings between these levels if required. Thus, the development process can be initiated from any level of abstraction and proceed towards obtaining one or many final user interfaces for various contexts of use (forward engineering), by recovering the final user inter- face into any upper level (reverse engineering), or by adapting at any level of ab- straction (reengineering).]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

List oCAS-2023el

:= *стандартное библиографическое описание**:
[*List of computer algebra systems [Electronic resource]*, англ., Mode of access: `http://en.wikipedia.org/wiki/List_of_computer_algebra_systems`. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.1. Назначение, принципы работы, классификация, структура и основные функциональные возможности систем компьютерной алгебры*

Liu B..DerivUIfOaMBA-2005art

:= *стандартное библиографическое описание**:
[B. Liu, H. Chen и W. He, “Deriving user interface from ontologies: A model-based approach,” т. 2005, дек. 2005, с. 6, ISBN: 0-7695-2488-5. DOI: `10.1109/ICTAI.2005.55`]

⇒ *аннотация**:
[Paper presents an approach to derive user interface (UI) from ontologies. It automatically generates UI according to declarative model specifications, and the UI helps a novice user to construct valid queries against a knowledge base (KB)]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Lopes L.S..SemanIitIoT-2022art

:= *стандартное библиографическое описание**:
[P. Lopes de Lopes de Souza, W. Lopes de Lopes de Souza и R. R. Ciferri, “Semantic Interoperability in the Internet of Things: A Systematic Literature Review,” в *ITNG 2022 19th International Conference on Information Technology-New Generations*, S. Latifi, ред., Cham: Springer International Publishing, 2022, с. 333—340]

⇐ *библиографическая ссылка**:
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Lowe D.G.DistiIFfSIK-2004art

:= *стандартное библиографическое описание**:
[D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, т. 60, № 2, с. 91—110, 2004. DOI: `10.1023/b:visi.0000029664.99615.94`]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Lowe W.Towar aToSS-2001art

:= *стандартное библиографическое описание**:
[W. Lowe, “Towards a Theory of Semantic Space,” англ., янв. 2001, с. 576—581]

⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Lu K..RethiMCfSCtSC-2022art

:= *стандартное библиографическое описание**:
[K. Lu и др., “Rethinking modern communication from semantic coding to semantic communication,” англ., *IEEE Wireless Communications*, 2022]

⇒ *аннотация**:
[Modern communications are usually designed to pursue a higher bit-level precision and fewer bits while transmitting a message. This article rethinks these two major features and introduces the concept and advantage of semantics that characterizes a new kind of semantics-aware communication framework, incorporating both the semantic encoding and the semantic communication problem. After analyzing the underlying defects of existing semantics-aware techniques, we establish a confidence-based distillation mechanism for the joint semanticsnoise coding (JSNC) problem and a reinforcement learning (RL)-powered semantic communication paradigm that endows a system the ability to convey the semantics instead of pursuing the bit level accuracy. On top of these technical contributions, this work provides a new insight to understand how the semantics are processed and represented in a semantics-aware coding and communication system, and verifies the significant benefits of doing so. Targeted on the next generation’s semantics-aware communication, some critical concerns and open challenges such as the information overhead, semantic security and implementation cost are also discussed and envisioned.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.5. Методы и средства поддержки проектирования и разработки программ в ostis-системах*

Lu L..ConteAfACaS-2002art

:= *стандартное библиографическое описание**:
[L. Lu, H.-J. Zhang и H. Jiang, “Content analysis for audio classification and segmentation,” *IEEE Transactions on speech and audio processing*, т. 10, № 7, с. 504—516, 2002]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Lu Y..Graph-2021art

:= *стандартное библиографическое описание**:
[Y. Lü и др., “GraphPEG,” *ACM Transactions on Architecture and Code Optimization*, т. 18, № 3, с. 1—24, сент. 2021. DOI: 10.1145/3450440. url: https://doi.org/10.1145/3450440]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Lucas B..aIteraIRTwaAtSV-1981art

:= *стандартное библиографическое описание**:
[B. Lucas и T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI),” англ., *Proceedings of Imaging Understanding Workshop*, с. 121—130, 1981]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Luhmann T..CloseRPa3DI-2018bk

:= *стандартное библиографическое описание**:
[T. Luhmann и др., *Close-Range Photogrammetry and 3D Imaging: 2nd Edition*. Berlin: De Gruyter, 2018]

⇒ *аннотация**:
[В данной книге подробно описаны фотограмметрические методы трехмерной реконструкции, приводятся сведения из смежных областей. Уделяется внимание как математическим, так и физическим принципам оптических систем компьютерного зрения. В главе 8 описаны примеры прикладных кибернетических систем, основанных на методах трехмерной реконструкции.]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *§ 4.4.1. Прикладные задачи трехмерной реконструкции*

Lundberg S.M..Advan iNIPS-2017art

:= *стандартное библиографическое описание**:
[S. M. Lundberg и S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” в *Advances in Neural Information Processing Systems*, I. Guyon и др., ред., т. 30, Curran Associates, Inc., 2017. url: `https://proceedings.neurips.cc/paper/2017/file/%208a20a8621978632d76c43dfd28b67767-Paper.pdf`]

⇒ *аннотация**:
[We present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Lupea M.aTheorPfCaRDL-2002art

:= *стандартное библиографическое описание**:
[M. Lupea, “A THEOREM PROVER FOR CONSTRAINED AND RATIONAL DEFAULT LOGICS,” англ., янв. 2002]

⇒ *аннотация**:
[Default logics represent an important class of the nonmonotonic formalisms. Using simple by powerful inference rules, called defaults, these logic systems model reasoning patterns of the form "in the absence of infor-mation to the contrary of. . . and thus formalize the default reasoning, a special type of nonmonotonic reasoning. In this paper we propose an auto-mated system, called DARR, with two components: a propositional theorem prover and a theorem prover for constrained and rational propositional default logics. A modified version of semantic tableaux method is used to implement the propositional prover. Also, this theorem proving method is adapted for computing extensions because one of its purpose is to produce models, and extensions are models of the world described by default theories.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Lutska N..OntolMoDTiM-2022art

:= *стандартное библиографическое описание**:
[N. Lutska и др., “Ontological Model of Digital Twin in Manufacturing,” Springer International Publishing, 2022, с. 310—335, ISBN: 978-3-031-15881-0. DOI: `10.1007/978-3-031-15882-7_16`. url: `http://dx.doi.org/10.1007/978-3-031-15882-7_16`]

⇒ *аннотация**:
[In this article, an approach to the continuous automation development of the processes of creation, evolvement, and usage of standards, based on ontological networks, is proposed. The existing international standards, reports, and guidelines in the field of Industry 4.0 and Industrial Internet of Things in the direction of digital twins (DT) are considered and analyzed.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

Mack R..KnowlPatEDKW-2001art

:= *стандартное библиографическое описание**:
[R. Mack, Y. Ravin и R. J. Byrd, “Knowledge portals and the emerging digital knowledge workplace,” *IBM systems journal*, т. 40, № 4, с. 925—955, 2001]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний*

Mair E..Adapt aGCDBotAST-2010art

:= *стандартное библиографическое описание**:

[E. Mair и др., “Adaptive and Generic Corner Detection Based on the Accelerated Segment Test,” англ., в *Computer Vision – ECCV 2010*, Springer Berlin Heidelberg, 2010, с. 183—196. DOI: 10.1007/978-3-642-15552-9_14]

⇐ *библиографическая ссылка**:

- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Manin Y.I.SemanS-2016art

:= *стандартное библиографическое описание**:

[Y. I. Manin и M. Marcolli, “Semantic Spaces,” англ., *Mathematics in Computer Science*, т. 10, с. 459—477, 2016]

⇒ *аннотация**:

[Any natural language can be considered as a tool for producing large databases (consisting of texts, written, or discursive). This tool for its description in turn requires other large databases (dictionaries, grammars etc.). Nowadays, the notion of database is associated with computer processing and computer memory. However, a natural language resides also in human brains and functions in human communication, from interpersonal to intergenerational one. We discuss in this survey/research paper mathematical, in particular geometric, constructions, which help to bridge these two worlds. In particular, in this paper we consider the Vector Space Model of semantics based on frequency matrices, as used in Natural Language Processing. We investigate underlying geometries, formulated in terms of Grassmannians, projective spaces, and flag varieties. We formulate the relation between vector space models and semantic spaces based on semic axes in terms of projectability of subvarieties in Grassmannians and projective spaces. We interpret Latent Semantics as a geometric flow on Grassmannians. We also discuss how to formulate Gärdenfors’ notion of “meeting of minds” in our geometric setting.]

⇐ *библиографическая ссылка**:

- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Manin Y.I.ZipfsLaLLPD-2014art

:= *стандартное библиографическое описание**:

[Y. I. Manin, “Zipf’s law and L. Levin’s probability distributions,” англ., *Functional Analysis and its Applications*, т. 48, № 2, с. 116—127, 2014. DOI: https://doi.org/10.1007/s10688-014-0052-1]

⇒ *аннотация**:

[Zipf’s law in its basic incarnation is an empirical probability distribution governing the frequency of usage of words in a language. As Terence Tao recently remarked, it still lacks a convincing and satisfactory mathematical explanation. In this paper I suggest that, at least in certain situations, Zipf’s law can be explained as a special case of the a priori distribution introduced and studied by L. Levin. The Zipf ranking corresponding to diminishing probability appears then as the ordering by growing Kolmogorov complexity. One argument justifying this assertion is the appeal to a recent interpretation by Yu. Manin and M. Marcolli of asymptotic bounds for error-correcting codes in terms of phase transition. In the respective partition function, the Kolmogorov complexity of a code plays the role of its energy.]

⇐ *библиографическая ссылка**:

- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

MapleB-el

:= *стандартное библиографическое описание**:

[*Maplesoft. Books [Electronic resource]*, англ., Mode of access: http://www.maplesoft.com/books/index.aspx. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

MaplePH-el

:= *стандартное библиографическое описание**:

[*Maple Product History [Electronic resource]*, англ., Mode of access: https://www.maplesoft.com/products/maple/history/. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Marrs T.Json aWPDIftW-2017bk

:= *стандартное библиографическое описание**:
[T. Marrs, *JSON at work: practical data integration for the web*, англ. O'Reilly Media, Inc., 2017]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.6. Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*

Masaharu T..aRevie otECTCE-2018art

:= *стандартное библиографическое описание**:
[M. Tsujimoto и др., "A review of the ecosystem concept — Towards coherent ecosystem design," *Technological Forecasting and Social Change*, т. 136, с. 49—58, 2018, ISSN: 0040-1625. DOI: https://doi.org/10.1016/j.techfore.2017.06.032]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

MatheИВ-эл

:= *стандартное библиографическое описание**:
[*Mathematica. История версий [Электронный ресурс]*, Режим доступа: http://ru.wikibooks.org/wiki/Mathematica/РЧСҒСЂР«СҐРчCS_РўРхСҐСҒРчРе. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

MatheQRH-el

:= *стандартное библиографическое описание**:
[*Mathematica Quick Revision History [Electronic resource]*, англ., Mode of access: http://www.wolfram.com/mathematica/quick-revision-history.html. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*

Matsukawa T..Devel otCDIoLK-1991art

:= *стандартное библиографическое описание**:
[T. Matsukawa и E. Yokota, "Development of the Concept Dictionary Implementation of Lexical Knowledge," англ., в *Lexical Semantics and Knowledge Representation*, 1991. url: https://aclanthology.org/W91-0219]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

McAulay R..SpeecASBoaSR-1986art

:= *стандартное библиографическое описание**:
[R. McAulay и T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, т. 34, № 4, с. 744—754, 1986]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

McCooI R.Rethi tSWP2-2006art

:= *стандартное библиографическое описание**:
[R. McCooI, "Rethinking the semantic web. Part 2," *IEEE Internet Computing*, т. 10, № 1, с. 93—96, 2006]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

McCrae J..InterLRotSW-2012art

:= *стандартное библиографическое описание**:
[J. McCrae и др., "Interchanging lexical resources on the Semantic Web," англ., *Language Resources and Evaluation*, т. 46, с. 701—719, дек. 2012. DOI: 10.1007/s10579-012-9182-3]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

McCrae J.P..OneOtBTA-2015art

:= *стандартное библиографическое описание**:

[J. P. McCrae и др., “One Ontology to Bind Them All: The META-SHARE OWL Ontology for the Interoperability of Linguistic Datasets on the Web,” в *The Semantic Web: ESWC 2015 Satellite Events*, F. Gandon и др., ред., Cham: Springer International Publishing, 2015, с. 271—282, ISBN: 978-3-319-25639-9**]**

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

McJones P.ParalLCMS-2015art

:= *стандартное библиографическое описание**:

[P. McJones. “Parallel Lisps: Connection Machine Lisp (StarLisp).” англ. Mode of access: `https://www.softwarepreservation.org/projects/LISP/parallel#Connection_Machine_\protect\discretionary{\protect\protect\leavevmode@ifvmode\kern+.1667em\relax\OMS/cmsy/m/n/10\char2}{}{}Lisp_(StarLisp)`. — Date of access: 29.12.2018, Computer History Museum. (дек. 2015)**]**

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

McKay B.D.NautyUGV24-2007art

:= *стандартное библиографическое описание**:

[B. D. McKay, “Nauty user’s guide (version 2.4),” *Computer Science Dept., Australian National University*, с. 225—239, 2007**]**

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.4.3. Расширение Программного интерфейса информационно-поисковых методов в Реализации sc-памяти в ostis-платформе. Реализация изоморфного поиска конструкций в sc-памяти по заданному графу-образцу*

Melcuk I.Langu fMtT-2016bk

:= *стандартное библиографическое описание**:

[I. Melcuk, *Language: from Meaning to Text*, англ., D. Beck, ред. Moscow & Boston: Academic Studies Press, апр. 2016, с. 270, ISBN: 978-1618114563**]**

⇒ *аннотация**:

[This volume presents a sketch of the Meaning-Text linguistic approach, richly illustrated by examples borrowed mainly, but not exclusively, from English. Chapter 1 expounds the basic idea that underlies this approach—that a natural language must be described as a correspondence between linguistic meanings and linguistic texts—and explains the organization of the book. Chapter 2 introduces the notion of linguistic functional model, the three postulates of the Meaning-Text approach (a language is a particular meaning-text correspondence, a language must be described by a functional model and linguistic utterances must be treated at the level of the sentence and that of the word) and the perspective “from meaning to text” for linguistic descriptions. Chapter 3 contains a characterization of a particular Meaning-Text model: formal linguistic representations on the semantic, the syntactic and the morphological levels and the modules of a linguistic model that link these representations. Chapter 4 covers two central problems of the Meaning-Text approach: semantic decomposition and restricted lexical cooccurrence (lexical functions); particular attention is paid to the correlation between semantic components in the definition of a lexical unit and the values of its lexical functions. Chapter 5 discusses five select issues: 1) the orientation of a linguistic description must be from meaning to text (using as data Spanish semivowels and Russian binominative constructions); 2) a system of notions and terms for linguistics (linguistic sign and the operation of linguistic union; notion of word; case, voice, and ergative construction); 3) formal description of meaning (strict semantic decomposition, standardization of semantemes, the adequacy of decomposition, the maximal block principle); 4) the Explanatory Combinatorial Dictionary (with a sample of complete lexical entries for Russian vocables); 5) dependencies in language, in particular—syntactic dependencies (the criteria for establishing a set of surface-syntactic relations for a language are formulated). Three appendices follow: a phonetic table, an inventory of surface-syntactic relations for English and an overview of all possible combinations of the three types of dependency (semantic, syntactic, and morphological). The book is supplied with a detailed index of notions and terms, which includes a linguistic glossary.**]**

⇐ *библиографическая ссылка**:

- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

Melekhova O..aDecenSfCD-2018art

≔ *стандартное библиографическое описание**:
[O. Melekhova и др., “A decentralised solution for coordinating decisions in large-scale autonomic systems,” в *MATEC Web of Conferences*, EDP Sciences, т. 161, 2018, с. 03 024]
⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.4. Комплекс свойств, определяющих качество физической оболочки кибернетической системы*

Memduhoglu M..PossiCoSSMaTtMRSDP-2018art
≔ *стандартное библиографическое описание**:
[A. Memduhoglu и M. Basaraner, “POSSIBLE CONTRIBUTIONS OF SPATIAL SEMANTIC METHODS AND TECHNOLOGIES TO MULTI-REPRESENTATION SPATIAL DATABASE PARADIGM,” *International Journal of Engineering and Geosciences*, окт. 2018. DOI: 10.26833/ijeg.413473. url: https://doi.org/10.26833/ijeg.413473]
⇒ *аннотация**:
[Today, the amount and variety of spatial data have increased dramatically. In addition, the web has made it easier to disseminate and share this kind of data. Therefore, spatial data integration and interoperability have gained more importance. Spatial data are collected from different sources and often heterogeneous in terms of the levels of detail and the points of view. To able to meet the demands of different spatial applications, multi-source and heterogeneous spatial datasets need to be integrated as well as the consistency of these datasets needs to be maintained. In this context, multirepresentation spatial database (MRSDB) paradigm has been suggested by researchers. However, the heterogeneity constitutes a significant barrier in this respect and hence the implementations have so far been remained within a rather narrow scope. In this article, it is mainly discussed about the possible contributions of basic methods and technologies of spatial semantics such as ontologies, semantic web and linked data to the data integration for creating a MRSBD. Some examples are also given to illustrate the concept.]
⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Meurisch C..ExploUEoPAI-2020art
≔ *стандартное библиографическое описание**:
[C. Meurisch, C. A. Mihale-Wilson и др., “Exploring User Expectations of Proactive AI Systems,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, т. 4, № 4, дек. 2020. DOI: 10.1145/3432193. url: https://doi.org/10.1145/3432193]
⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

Meurisch C..ReferMoNGDP-2017art
≔ *стандартное библиографическое описание**:
[C. Meurisch, M.-D. Ionescu и др., “Reference Model of Next-Generation Digital Personal Assistant: Integrating Proactive Behavior,” в *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, сер. UbiComp ’17, Maui, Hawaii: Association for Computing Machinery, 2017, с. 149—152, ISBN: 9781450351904. DOI: 10.1145/3123024.3123145. url: https://doi.org/10.1145/3123024.3123145]
⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.4. Персональные ostis-ассистенты пользователей*

Mnih V..HumanLCtDRL-2015art
≔ *стандартное библиографическое описание**:
[V. Mnih и др., “Human-Level Control through Deep Reinforcement Learning,” *Nature*, т. 518, с. 529—533, 2015]
⇒ *аннотация**:
[The theory of reinforcement learning provides a normative account, deeply rooted in psychological and neuroscientific perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations. Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems, the former evidenced by a wealth of neural data revealing notable parallels between the phasic signals emitted by dopaminergic neurons and temporal difference reinforcement learning algorithms. While reinforcement learning

agents have achieved some successes in a variety of domains, their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. Here we use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. We tested this agent on the challenging domain of classic Atari 2600 games. We demonstrate that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a professional human games tester across a set of 49 games, using the same algorithm, network architecture and hyperparameters. This work bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*

Moens M..TempoOiNL-1987art

:= *стандартное библиографическое описание**:
[M. Moens и M. Steedman, “Temporal Ontology in Natural Language,” англ., в *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics*, сер. ACL ’87, Stanford, California: Association for Computational Linguistics, 1987, с. 1—7. DOI: 10.3115/981175.981176. url: https://doi.org/10.3115/981175.981176]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Mohseni M..aModelDAfSW-2021art

:= *стандартное библиографическое описание**:
[M. Mohseni, M. Sohrabi и M. Dorrigiv, “A model-driven approach for semantic web service modeling using web service modeling languages,” *Journal of Software: Evolution and Process*, т. 33, июнь 2021. DOI: 10.1002/smr.2364]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

Moldovan D..SNAPPPAtAI-1992art

:= *стандартное библиографическое описание**:
[D. Moldovan и др., “SNAP: Parallel Processing Applied to AI,” *Computer*, с. 39—49, 1992]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Moldovan D.I..SNAP aVLSIAfAIP-1985art

:= *стандартное библиографическое описание**:
[D. I. Moldovan и Y.-W. Tung, “SNAP: A VLSI architecture for artificial intelligence processing,” *Journal of Parallel and Distributed Computing*, т. 2, № 2, с. 109—131, 1985, ISSN: 0743-7315. DOI: https://doi.org/10.1016/0743-7315(85)90031-0. url: https://www.sciencedirect.com/science/article/pii/0743731585900310]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.3. Вариант мелкозернистой архитектуры ассоциативных семантических компьютеров*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Montero M..Adapt aAWBES-2005bk

:= *стандартное библиографическое описание**:
[M. Montero и E. Gaudioso, *Adaptable and Adaptive Web-Based Educational Systems*, англ. UK: Liverpool John Moores University, 2005, с. 8—11. DOI: 10.4018/978-1-59140-562-7.ch002]

⇒ *аннотация**:
[This article present how to obtain adaptable and adaptive systems. Next, briefly present how to combine both types of personalization in PDINAMET, a WES for Physics, describe some future trends and conclusions.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Moon D.A.SymboA-1987art

:= *стандартное библиографическое описание**:
[D. A. Moon, “Symbolics Architecture,” *Computer*, т. 20, № 1, с. 43—52, 1987. url: doi:10.1109/MC.1987.1663356]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Mousavinasab E..IntelTSaSRoC-2018art

:= *стандартное библиографическое описание**:
[E. Mousavinasab и др., “Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods,” *Interactive Learning Environments*, т. 29, № 1, с. 142—163, дек. 2018. DOI: 10.1080/10494820.2018.1558257. url: https://doi.org/10.1080/10494820.2018.1558257]

⇒ *аннотация**:
[Работа содержит описания интеллектуальных обучающих систем]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Moussa M..ArchiSaMfANNI-2013el

:= *стандартное библиографическое описание**:
[M. Moussa, A. Savich и S. Areibi, *Architecture, system and method for artificial neural network implementation*, июнь 2013]

⇒ *аннотация**:
[Патент на вычислительную архитектуру для обработки вычислительных нейронных сетей.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

MathWMtLoT-el

:= *стандартное библиографическое описание**:
[*MathWorks. MATLAB. The Language of Technical Computing [Electronic resource]*, англ., Mode of access: http://www.mathworks.com/products/matlab/. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Myers B.A..Surve oUIP-1992art

:= *стандартное библиографическое описание**:
[R. M. Myers B.A., “Survey on User Interface Programming,” в *Proceedings SIGCHI’92: Human Factors in Computing Systems*, Monterrey, CA, 1992, с. 195—202]

⇒ *аннотация**:
[This paper reports on the results of a survey of user interface programming. The survey was widely distributed, and we received 74 responses. The results show that in today’s applications, an average of 48% of the code is devoted to the user interface portion. The average time spent on the user interface portion is 45% during the design phase, 50% during the implementation phase, and 37% during the maintenance phase. 34% of the systems were implemented using a toolkit, 27% used a UIMS, 14% used an interface builder, and 26% used no tools. This appears to be because the toolkit systems had more sophisticated user interfaces. The projects using UIMSs or interface builders spent the least percent of time and code on the user interface (around 41%) suggesting that these tools are effective. In general, people were happy with the tools they used, especially the graphical interface builders. The most common problems people reported when developing a user interface included getting users’ requirements, writing help text, achieving consistency, learning how to use the tools, getting acceptable performance, and communicating among various parts of the program.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.7. Реализация интерпретатора sc-моделей пользовательских интерфейсов*

Myers B.aBriefHoHCIT-2001art

:= *стандартное библиографическое описание**:
[B. Myers, “A Brief History of Human Computer Interaction Technology,” т. 5, 2001. DOI: 10.1145/274430.274436]

⇒ *аннотация**:
[This article summarizes the historical development of major advances in human-computer interaction technology, emphasizing the pivotal role of university research in the advancement of the field.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*

MyHDL-2023el

:= *стандартное библиографическое описание**:
[*MyHDL*, English, Mode of access: https://www.myhdl.org/. — Date of access: 29.03.2023]

⇒ *аннотация**:
[Официальный сайт проекта MyHDL для языка Python. MyHDL превращает Python в язык описания и проверки оборудования.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*

Nacer H..SemanWSSAC-2014art

:= *стандартное библиографическое описание**:
[H. Nacer и D. Aissani, “Semantic web services: Standards, applications, challenges and solutions,” *Journal of Network and Computer Applications*, т. 44, с. 134—151, 2014]

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*

Nagendra Prasad M.V..LearnSSCiCMAS-1999art

:= *стандартное библиографическое описание**:
[M. V. Nagendra Prasad и V. R. Lesser, “Learning Situation-Specific Coordination in Cooperative Multi-agent Systems,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 2, № 2, с. 173—207, 1999]

⇒ *аннотация**:
[В статье рассмотрен еще один вариант координации агентов.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Neiva F.W..TowarPItSC-2016art

:= *стандартное библиографическое описание**:
[F. W. Neiva и др., “Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature,” *Information and Software Technology*, т. 72, с. 137—150, 2016, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2015.12.013]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

Neo4jGDPGDMS-2023el

:= *стандартное библиографическое описание**:
[“Neo4j Graph Database Platform | Graph Database Management System.” англ. (сент. 2023), url: https://neo4j.com/]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Neumann J.FirstDoaRotEDVAC-1993art

:= *стандартное библиографическое описание**:
[J. von Neumann, “First draft of a report on the EDVAC,” *IEEE Annals of the History of Computing*, т. 15, № 4, с. 27—75, 1993. DOI: 10.1109/85.238389]

⇒ *аннотация**:
[Переиздание работы Дж. фон Неймана, описывающей основные принципы предложенной им архитектуры.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.2. Методы и средства реализации ostis-систем*

Nevzorova O..OntolDPoUT-2019art

:= *стандартное библиографическое описание**:
[O. Nevzorova и V. Nevzorov, “Ontology-Driven Processing of Unstructured Text,” в *Artificial Intelligence*, S. O. Kuznetsov и A. I. Panov, ред., Cham: Springer International Publishing, 2019, с. 129—142, ISBN: 978-3-030-30763-9]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Nilsson N.J.HumanLAIBS-2005art

:= *стандартное библиографическое описание**:
[N. J. Nilsson, “Human-level artificial intelligence? Be serious!” *AI magazine*, т. 26, № 4, с. 68—68, 2005]

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

NLTKNLPL-2022el

:= *стандартное библиографическое описание**:
[“NLTK NLP library.” англ. Mode of access: https://www.nltk.org/. — Date of access: 24.10.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Norton J.D.aDemon otIoCoII-2019art

:= *стандартное библиографическое описание**:
[J. D. Norton, “A Demonstration of the Incompleteness of Calculi of Inductive Inference,” *The British Journal for the Philosophy of Science*, т. 70, № 4, с. 1119—1144, 2019. DOI: 10.1093/bjps/axx004. url: https://doi.org/10.1093/bjps/axx004]

⇒ *аннотация**:
[Статья, в которой рассматривается индуктивный вывод.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Omicini A..Coord fIAD-1999art

:= *стандартное библиографическое описание**:
[A. Omicini и F. Zambonelli, “Coordination for Internet Application Development,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 2, № 2, с. 251—269, июнь 1999]

⇒ *аннотация**:
[В статье предложена идея, заключающаяся в том, чтобы вводить в сеть агентов коммуникационные центры, регулирующие общение агентов.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Omidvar O..NeuraSfC-1997bk

:= *стандартное библиографическое описание**:
[O. Omidvar и D. Elliott, *Neural Systems for Control*, англ. New York: Academic Press, 1997]

⇒ *аннотация**:
[Neural Systems for Control represents the most up-to-date developments in the rapidly growing aplication area of neural networks and focuses on research in natural and artifical neural systems directly applicable to control or making use of modern control theory. The book covers such important new developments in control systems such as intelligent sensors in semiconductor wafer manufacturing; the relation between muscles and cerebral neurons in speech recognition; online compensation of reconfigurable control for spacecraft aircraft and other systems; applications to rolling mills, robotics and process control; the usage of past output data to identify nonlinear systems by neural networks; neural approximate optimal control; model-free nonlinear control; and neural control based on a regulation of physiological investigation/blood pressure control. All researchers and students dealing with control systems will find the fascinating Neural Systems for Control of immense interest and assistance.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

OpenCL-2023el

:= *стандартное библиографическое описание**:
[*OpenCL*, English, Mode of access: `https://www.khronos.org/opencl/`. — Date of access: 29.03.2023]

⇒ *аннотация**:
[Официальный сайт проекта OpenCL]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Orlov M.K..NonprPSMiN-2022art

:= *стандартное библиографическое описание**:
[M. K. Orlov и A. P. Vasilevskaya, “Non-procedural problem-solving models in next-generation intelligent computer systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 6, Minsk : BSUIR, 2022, с. 161—172]

⇒ *аннотация**:
[В работе рассматривается подход к проектированию решателей задач интеллектуальных систем на основе непроцедурных моделей. Разрабатываемый подход позволяет интегрировать любые модели решения задач, в том числе принципы логического вывода, для решения задач на основе общей формальной модели.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Orlov M.K.ComprLoRSC-2022art

:= *стандартное библиографическое описание**:
[M. K. Orlov, “Comprehensive library of reusable semantically compatible components of next-generation intelligent computer systems,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 6, Minsk : BSUIR, 2022, с. 261—272]

⇒ *аннотация**:
[В работе рассматривается подход к проектированию систем, управляемых знаниями, ориентированный на использование совместимых многократно используемых компонентов, что существенно сокращает трудоемкость разработки таких систем.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.3. Понятие многократно используемого компонента ostis-систем*

OSTISSR-el

:= *стандартное библиографическое описание**:
[*OSTIS Standard Repository [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/ostis-standard`. — Date of access: 01.03.2023]

⇐ *библиографическая ссылка**:
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.4.3. Язык представления исходных текстов баз знаний на основе языка LaTeX*

Ouksel A.M..SemanIiGIS-1999art

:= *стандартное библиографическое описание**:
[A. M. Ouksel и A. Sheth, “Semantic Interoperability in Global Information Systems,” *SIGMOD Rec.*, т. 28, № 1, с. 5—12, март 1999]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *§ 1.1.2. Понятие интеллектуальной многоагентной системы*

OWLI-2023el

:= *стандартное библиографическое описание**:
[“OWL Implementations [Electronic resource],” англ., *World Wide Web Consortium (W3C)*, Mode of access: `https://www.w3.org/2001/sw/wiki/OWL/Implementations/`. — Date of access: 23.01.2023]

⇒ *аннотация**:
[Официальный сайт, описывающий применение языка OWL]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

OWL2WOLDO-2023el

:= *стандартное библиографическое описание**:
[“OWL 2 Web Ontology Language document overview [Electronic resource],” англ., *World Wide Web Consortium (W3C)*, Mode of access: `http://www.w3.org/TR/owl2-overview`. — Date of access: 23.01.2023]

⇒ *аннотация**:
[Официальный сайт, описывающий язык OWL 2]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Pais S..NLPBPaaSaBR-2022art

:= *стандартное библиографическое описание**:
[S. Pais, J. Cordeiro и M. L. Jamil, “NLP-based platform as a service: a brief review,” англ., *Journal of Big Data*, т. 9, апр. 2022. DOI: `10.1186/s40537-022-00603-5`]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Paola A..ConteAfMSDF-2016art

:= *стандартное библиографическое описание**:
[A. De Paola и др., “Context-awareness for multi-sensor data fusion in smart environments,” т. 10037, 29 нояб. 2016, с. 377—391, ISBN: 978-3-319-49129-5. DOI: `10.1007/978-3-319-49130-1_28`]

⇒ *аннотация**:
[Multi-sensor data fusion is extensively used to merge data collected by heterogeneous sensors deployed in smart environments. However, data coming from sensors are often noisy and inaccurate, and thus probabilistic techniques, such as Dynamic Bayesian Networks, are often adopted to explicitly model the noise and uncertainty of data. This work proposes to improve the accuracy of probabilistic inference systems by including context information, and proves the suitability of such an approach in the application scenario of user activity recognition in a smart home environment. However, the selection of the most convenient set of context information to be considered is not a trivial task. To this end, we carried out an extensive experimental evaluation which shows that choosing the right combination of context information is fundamental to maximize the inference accuracy.]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Papasalouros A..AutomGoMCQ-2008art

:= *стандартное библиографическое описание**:

[A. Papasalouros, K. Kanaris и K. Kotis, “Automatic Generation Of Multiple Choice Questions From Domain Ontologies.,” *e-Learning*, т. 1, с. 427—434, 2008]

⇒ *аннотация**:

[Работа содержит описание метода генерации вопросов на выбор]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.1 Автоматическая генерация тестовых вопросов*

Paterno F..MARIAaUDMALL-2009art

:= *стандартное библиографическое описание**:

[F. Paternò, C. Santoro и L. Spano, “MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments,” *ACM Trans. Comput.-Hum. Interact.*, т. 16, нояб. 2009. DOI: 10.1145/1614390.1614394]

⇒ *аннотация**:

[The article discussing how a novel model-based UIDL can provide useful support both at design and runtime for applications]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Paulheim H..UI20AFOoUIaI-2013art

:= *стандартное библиографическое описание**:

[H. Paulheim и F. Probst, “UI2Ont—A Formal Ontology on User Interfaces and Interactions,” с. 1—24, 2013. DOI: 10.1007/978-1-4471-5301-6_1]

⇒ *аннотация**:

[In this article discussing the UI 2 Ont ontology, an ontology of user interfaces and interactions, which reuses many concepts defined in different user interface description languages and grounds them in the formal top level ontology DOLCE. Discussing the rationales of developing the ontology, give an overview of its basic concepts, and show its application in a framework for application integration on the user interface level.]

⇐ *библиографическая ссылка**:

- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Pau L.F.KnowlBP-1990art

:= *стандартное библиографическое описание**:

[L. F. Pau, “Knowledge-Based Processing,” англ., в *Computer vision for electronics manufacturing*, Boston, 1990, с. 127—131]

⇒ *аннотация**:

[Статья посвящена вопросам решения задач в системах, основанных на знаниях.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Pavel M..BehavIaCMiSoP-2015art

:= *стандартное библиографическое описание**:

[M. Pavel и др., “Behavioral Informatics and Computational Modeling in Support of Proactive Health Management and Care,” *IEEE Transactions on Biomedical Engineering*, т. 62, № 12, с. 2763—2775, дек. 2015. DOI: 10.1109/tbme.2015.2484286. url: https://doi.org/10.1109/tbme.2015.2484286]

⇒ *аннотация**:

[Статья посвящена вопросам применения идей бихевиоризма в информатике в контексте решения задач в медицине.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *§ 3.3.8. Актуальные проблемы и перспективы развития технологий разработки гибридных решателей задач*

Pearl C.DesigVUIPoC-2016bk

:= *стандартное библиографическое описание**:

[C. Pearl, *Designing voice user interfaces: principles of conversational experiences*. O’Reilly Media, Inc., 2016]

⇐ *библиографическая ссылка**:

- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Pease A..tSuggeUMOaL-2002art

:= *стандартное библиографическое описание**:
[A. Pease, I. Niles и J. Li, “The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications,” янв. 2002]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Penta M..otRelatBRAaBaD-2020art

:= *стандартное библиографическое описание**:
[M. Di Penta, G. Bavota и F. Zampetti, “On the relationship between refactoring actions and bugs: a differentiated replication,” в *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, с. 556—567]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Petrovsky A..HybriSDBoIH-2011art

:= *стандартное библиографическое описание**:
[A. Petrovsky, E. Azarov и A. Petrovsky, “Hybrid signal decomposition based on instantaneous harmonic parameters and perceptually motivated wavelet packets for scalable audio coding,” *Signal processing*, т. 91, № 6, с. 1489—1504, 2011]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Pileggi S..Ontol iSE-2018art

:= *стандартное библиографическое описание**:
[S. Pileggi, A. Lopez-Lorca и G. Beydoun, “Ontologies in Software Engineering,” англ., нояб. 2018]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Pohl J.Inter atNfISaHP-2004art

:= *стандартное библиографическое описание**:
[J. Pohl, “Interoperability and the Need for Intelligent Software: A Historical Perspective,” сент. 2004]

⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*

Popov V..Fast aLoDTTSw-2020art

:= *стандартное библиографическое описание**:
[V. Popov и др., “Fast and lightweight on-device TTS with Tacotron2 and LPCNet,” в *Proc. Interspeech*, 2020, с. 220—224]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Portner P.H..FormaStER-2008bk

:= *стандартное библиографическое описание**:
[P. Portner и B. Partee, *Formal Semantics: The Essential Readings* (Linguistics: The Essential Readings), англ. Wiley, 2008, ISBN: 9780470758182. url: `https://books.google.by/books?id=ptgUWREtAkMC`]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.2. Формализация денотационной семантики естественных языков*

Povey D..tKaldiSRT-2011art

:= *стандартное библиографическое описание**:
[D. Povey и др., “The Kaldi Speech Recognition Toolkit,” в *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, 2011]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Prakash S.P.Worki wMPA-2022art

:= *стандартное библиографическое описание**:
[S. P. Pradhan, “Working with Microsoft Power Apps,” англ., в *Power Platform and Dynamics 365 CE for Absolute Beginners*, Apress, 2022, с. 79—131. DOI: 10.1007/978-1-4842-8600-5_3. url: https://doi.org/10.1007/978-1-4842-8600-5_3]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Protege-2016el

:= *стандартное библиографическое описание**:
[“Protege [Electronic resource],” англ., Mode of access: http://protege.stanford.edu. — Date of access: 27.05.2016]

⇒ *аннотация**:
[Офицальный сайт редактора онтологий Protege.]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.1 Автоматическая генерация тестовых вопросов*

Puerta A..BeyonDMfAUIG-1994art

:= *стандартное библиографическое описание**:
[A. Puerta и др., “Beyond Data Models for Automated User Interface Generation.,” янв. 1994, с. 353—366]

⇒ *аннотация**:
[The paper presents Mecano, a model-based interface development environment that extends the concept of generating interface specifications from data models. Mecano employs a domain model to generate not only the layout of an interface, but also its dynamic behavior.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Qian L..OntolAfCLID-2020art

:= *стандартное библиографическое описание**:
[L. Qian, M. Sadouski и W. Li, “Ontological Approach for Chinese Language Interface Design,” в *Communications in Computer and Information Science*, Springer International Publishing, 2020, с. 146—160. DOI: 10.1007/978-3-030-60447-9_9. url: https://doi.org/10.1007/978-3-030-60447-9_9]

⇒ *аннотация**:
[The natural language user interface is a subclass of user interfaces that allows user and system to communicate using natural language. It is the development direction of the user interface of the intelligent system. The key technology for implementation of natural language user interface is the computer processing of natural language text. Due to the diversity and complexity of natural language, its understanding hasn't completely achieved yet. By comparing Chinese language with other European languages, this article describes the characteristics of Chinese language and the difficulties in Chinese language processing. After an analysis of current mainstream natural language processing methods, it was shown that the knowledge base plays an important role in the natural language processing model. The knowledge base is the basis for natural language processing. This article proposes a method of computer processing of Chinese language text based on Chinese linguistic ontology and domain ontologies. The ontologies are used to build a unified semantic model of Chinese linguistic knowledge and domain knowledge for the processing of Chinese language text. In this way the Chinese linguistic knowledge is integrated in the Chinese language processing model, the application of Chinese linguistic knowledge makes the Chinese language processing model more interpretative.]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

- *Подпункт 7.5.3.2.3 Проверка ответов на объективные вопросы*
- *Подпункт 7.5.3.2.4 Проверка ответов на субъективные вопросы*

RDB tRML-2012el

:= *стандартное библиографическое описание**:
[*RDB to RDF Mapping Language*, англ., 2012. url: `https://www.w3.org/TR/r2rml/`]

⇒ *аннотация**:
[В работе рассматривается принцип работы R2RML. R2RML(RDB to RDF Mapping Language) — это язык маппинга для преобразования данных из реляционных баз данных (RDB) в RDF, используемый в протоколе SPARQL.]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами*

EasilGHKG-2022el

:= *стандартное библиографическое описание**:
[*Easily generate high-quality knowledge graphs with RML.io*, англ., 2022. url: `https://rml.io/`]

⇒ *аннотация**:
[R2RML.io является онлайн-инструментом для создания файлов маппинга R2RML без необходимости знания языка R2RML.]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами*

Radford A..RobusSRvL-2022art

:= *стандартное библиографическое описание**:
[A. Radford и др., “Robust speech recognition via large-scale weak supervision,” Tech. Rep., Technical report, OpenAI, тех. отч., 2022]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Rasheed B..NetwoGDUDI-2019art

:= *стандартное библиографическое описание**:
[B. Rasheed и A. Popov, “Network Graph Datastore Using DISC Processor,” в *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, янв. 2019, с. 1582—1587. DOI: `10.1109/EIConRus.2019.8656749`]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

RDFCAS-2023el

:= *стандартное библиографическое описание**:
[“RDF 1.1 concepts and abstract syntax [Electronic resource],” англ., *World Wide Web Consortium*, Mode of access: `https://www.w3.org/TR/rdf11-concepts`. — Date of access: 18.01.2023]

⇒ *аннотация**:
[Официальный сайт, описывающий язык RDF 1.1]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.2. Принципы интеграции Экосистемы OSTIS со структурированными информационными ресурсами*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Reiter R.aLogic fDR-1980cm

:= *стандартное библиографическое описание**:
[R. Reiter, “A Logic for Default Reasoning,” англ., *Artificial Intelligence*, т. 13, № 13, с. 81—132, 1980]

⇒ *аннотация**:
[Ключевая статья, посвязенная логике умолчаний.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Ribeiro M.T..WhySITY-2016art

:= *стандартное библиографическое описание**:
[M. T. Ribeiro, S. Singh и C. Guestrin, *Why Should I Trust You?: Explaining the Predictions of Any Classifier*, 2016. DOI: 10.48550/ARXIV.1602.04938. url: https://arxiv.org/abs/1602.04938]

⇒ *аннотация**:
[In this work, we propose LIME, a novel explanation technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. We also propose a method to explain models by presenting representative individual predictions and their explanations in a non-redundant way, framing the task as a submodular optimization problem. We demonstrate the flexibility of these methods by explaining different models for text (e.g. random forests) and image classification (e.g. neural networks)]

⇐ *библиографическая ссылка**:
- *Глава 3.6. Конвергенция и интеграция искусственных нейронных сетей с базами знаний в ostis-системах*

Robinson I..GraphD-2015bk

:= *стандартное библиографическое описание**:
[Ian Robinson, Jim Webber and Emil Eifrem, *Graph databases*, англ. O'Reilly Media, Inc., 2015]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Rosen A.MalveTAaaNT-2021art

:= *стандартное библиографическое описание**:
[A. Rosen. "Malvertising Takes Aim at a New Target: IoT Devices Connected to Smart Home Networks." англ. (авг. 2021), url: https://www.geoedge.com/malvertising-attack-iot-devices/]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Rosten E..Faste aBaMLA-2010art

:= *стандартное библиографическое описание**:
[E. Rosten, R. Porter и T. Drummond, "Faster and Better: A Machine Learning Approach to Corner Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, т. 32, № 1, с. 105—119, 2010. DOI: 10.1109/tpami.2008.275]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Rosten E..MachiLfHS-2006art

:= *стандартное библиографическое описание**:
[E. Rosten и T. Drummond, "Machine Learning for High-Speed Corner Detection," в *Computer Vision – ECCV 2006*, Springer Berlin Heidelberg, 2006, с. 430—443. DOI: 10.1007/11744023_34]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Rublee E..ORB:aEAtS-2011art

:= *стандартное библиографическое описание**:
[E. Rublee и др., "ORB: An efficient alternative to SIFT or SURF," в *2011 International Conference on Computer Vision*, IEEE, 2011. DOI: 10.1109/ICCV.2011.6126544]

⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Rujiang W..Revie oCPTaM-2011art

≔ *стандартное библиографическое описание**:
[W. X. H. Z. B. Rujiang и L. Yuping, “Review on Concepts, Processes, Tools and Methods of Ontology Integration,” *Library and Information Service*, т. 55, № 16, с. 119, 2011]
⇒ *аннотация**:
[Работа содержит описание методов слияния онтологий]
⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Rumbell T..Emoti iAACA-2012art
≔ *стандартное библиографическое описание**:
[T. Rumbell и др., “Emotions in autonomous agents: comparative analysis of mechanisms and functions,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 25, № 1, с. 1—45, июнь 2012]
⇒ *аннотация**:
[Подход к регулированию деятельности агентов на основе эмоций]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Rybina G..Metho aAfVoKB-2007art
≔ *стандартное библиографическое описание**:
[G. Rybina и V. Smirnov, “Methods and Algorithms for Verification of Knowledge Bases in Integrated Expert Systems,” *Journal of Computer and Systems Sciences International*, т. 46, с. 590—601, янв. 2007. DOI: 10.1134/S1064230707040090]
⇐ *библиографическая ссылка**:
- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы*

Sadouski M..Metho aTfCID-2022art
≔ *стандартное библиографическое описание**:
[M. Sadouski и A. Zhmyrko, “Methodology and Tools for Component Interface Design of Next-generation Intelligent Computer Systems,” англ., в *Open semantic technologies for intelligent systems*, Minsk : BSUIR, 2022, с. 279—284]
⇒ *аннотация**:
[In the article, the methodology of designing interfaces for next-generation computer systems is considered. The stages of designing adaptive intelligent multimodal user interfaces and the usage of these stages in the context of the OSTIS Technology are described.]
⇐ *библиографическая ссылка**:
- *Глава 5.4. Методика и средства компонентного проектирования интерфейсов ostis-систем*
- *§ 5.4.1. Анализ методик проектирования пользовательских интерфейсов*

Sadouski M.OntolAttBoSM-2021art
≔ *стандартное библиографическое описание**:
[M. Sadouski, “Ontological approach to the building of semantic models of user interfaces,” англ., в *Open semantic technologies for intelligent systems*, Minsk : BSUIR, 2021, с. 105—116]
⇒ *аннотация**:
[The article is dedicated to the description of the ontological approach to the building of the user interface based on the OSTIS Technology. The existing approaches in the field of the building of user interfaces are considered and an ontological model of support of the building process is described.]
⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*

Sadouski M.SemanDoaAUI-2022art
≔ *стандартное библиографическое описание**:
[M. Sadouski, “Semantic-Based Design of an Adaptive User Interface,” в *International Conference on Open Semantic Technologies for Intelligent Systems*, Springer, 2022, с. 165—191]
⇒ *аннотация**:
[An approach to the building of adaptive user interfaces (UIs) based on their semantic model (semantic-based design) is described in the article. A variant of the implementation of a framework for building UIs is proposed within this approach, which is based on the OSTIS Technology. The existing approaches to the building of UIs are also

considered; examples and methods of designing UI components for the proposed implementation variation are shown.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*
- *§ 4.1.5. Действия и внутренние агенты пользовательского интерфейса ostis-системы*
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.7.1. Основные компоненты Реализации интерпретатора sc-моделей пользовательских интерфейсов*

Sadouski M.tSrtuct oNICSI-2022art

:= *стандартное библиографическое описание**:
[M. Sadouski, “The structure of next-generation intelligent computer system interfaces,” англ., в *Open semantic technologies for intelligent systems*, Minsk : BSUIR, 2022, с. 199—208]

⇒ *аннотация**:
[This article deals with the structure of adaptive multimodal interfaces of next-generation intelligent computer systems, which provide a transition from the paradigm of literate user to the paradigm of equal cooperation between the user and the intelligent system, which will increase the efficiency of human-machine interaction.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.2. Предлагаемый подход к организации интерфейсов ostis-систем*

Safawi A.R..tDecisPoAI-2015art

:= *стандартное библиографическое описание**:
[S. Abdul Rahman и др., “The Decision Processes of Abductive Inference,” англ., *Advanced Science Letters*, т. 21, с. 1754—1757, июнь 2015. DOI: 10.1166/asl.2015.6193]

⇒ *аннотация**:
[A vast number of problem solving frameworks have been established in the realms of management, business, mathematics, healthcare, aerospace, law and etcetera towards facilitating effective decision making in normative form in particular. It was learnt that those problem solving frameworks are originated mainly from practical exercises of practitioners and/or improvement/revision of an earlier frameworks. Against this bottom-up approach in the establishment of problem solving frameworks, this paper continues promoting the establishment of a problem solving framework following top-down approach by synthesizing the deductive inference i.e., the second inference process of Pragmatism’s scientific method. The synthesis has resulted in the identification of the decision sub-processes of Pragmatism’s deductive inference namely analysis and demonstration. This paper posits that these two decision processes are aligned with both the key and strategies of the decision processes of the renowned problem solving frameworks in practice in management, business, mathematics, healthcare and law and hence, the decision processes of deductive inference in particular and Pragmatism’s philosophical method in general is an alternative for decision making processes and problem solving frameworks following the top-down approach.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Saha D..Athen aOSfNL-2016art

:= *стандартное библиографическое описание**:
[D. Saha и др., “ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores,” англ., *Proc. VLDB Endow.*, т. 9, № 12, с. 1209—1220, авг. 2016, ISSN: 2150-8097. DOI: 10.14778/2994509.2994536. url: https://doi.org/10.14778/2994509.2994536]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Samodumkin S.A.SemanToIGSD-2019art

:= *стандартное библиографическое описание**:
[S. A. Samodumkin, “Semantic Technology of Intellectual Geoinformation Systems Development,” англ., в *Open semantic technologies for intelligent systems*, сер. Iss. 3, Minsk : BSUIR, 2019, с. 231—236]

⇒ *аннотация**:
[Работа содержит описание семантических технологий развития интеллектуальных геоинформационных систем.]

⇐ *библиографическая ссылка**:
- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Samodumkin S.Next-IGS-2022art

:= *стандартное библиографическое описание**:

[S. Samodumkin, “Next-generation intelligent geoinformation systems,” 2022]

⇒ *аннотация**:

[In the article, an approach to the building of intelligent geoinformation systems based on the OSTIS Technology is considered. The formal ontology of the syntax of the map language is explicitly set, which, in turn, allows establishing the types of map objects and setting spatial semantic relations; the formal ontology of the denotation semantics of the map language is set, which, in turn, allows establishing the semantics of displaying geo-entities on maps depending on the types of terrain objects; the formal ontology of terrain objects is set as a necessary condition for integration with subject domains in interests of GIS.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Samodumkin S.SemanToIGSD-2019art

:= *стандартное библиографическое описание**:

[S. Samodumkin, “Semantic Technology of Intellectual Geoinformation Systems Development,” 2019]

⇒ *аннотация**:

[This paper is devoted to the creation of a special technology for designing intelligent geo-information systems that use knowledge of terrain objects to solve applied tasks in problem areas.]

⇐ *библиографическая ссылка**:

- *Глава 7.8. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла интеллектуальных геоинформационных систем различного назначения*

Sapasgozar S..aSysteCRoAI-2020art

:= *стандартное библиографическое описание**:

[S. Sepasgozar и др., “A Systematic Content Review of Artificial Intelligence and the Internet of Things Applications in Smart Home,” *Applied Sciences*, т. 10, № 9, 2020, ISSN: 2076-3417. DOI: 10.3390/app10093074. url: https://www.mdpi.com/2076-3417/10/9/3074]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

Scalabrino S..ImproCRMwTF-2016art

:= *стандартное библиографическое описание**:

[S. Scalabrino и др., “Improving code readability models with textual features,” в *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, IEEE, 2016, с. 1—10]

⇒ *аннотация**:

[Code reading is one of the most frequent activities in software maintenance; before implementing changes, it is necessary to fully understand source code often written by other developers. Thus, readability is a crucial aspect of source code that may significantly influence program comprehension effort. In general, models used to estimate software readability take into account only structural aspects of source code, e.g., line length and a number of comments. However, source code is a particular form of text; therefore, a code readability model should not ignore the textual aspects of source code encapsulated in identifiers and comments. In this paper, we propose a set of textual features aimed at measuring code readability. We evaluated the proposed textual features on 600 code snippets manually evaluated (in terms of readability) by 5K+ people. The results demonstrate that the proposed features complement classic structural features when predicting code readability judgments. Consequently, a code readability model based on a richer set of features, including the ones proposed in this paper, achieves a significantly higher accuracy as compared to all of the state-of-the-art readability models.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Schalley A.C.Ontol aOMiL-2019art

:= *стандартное библиографическое описание**:

[A. C. Schalley, “Ontologies and ontological methods in linguistics,” *Language and Linguistics Compass*, т. 13, № 11, e12356, 2019, e12356 LNCO-0634.R3. DOI: https://doi.org/10.1111/lnc3.12356. eprint: https://compass.onlinelibrary.wiley.com/doi/pdf/10.1111/lnc3.12356. url: https://compass.onlinelibrary.wiley.com/doi/abs/10.1111/lnc3.12356]

⇐ *библиографическая ссылка**:

- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Schiitze H.tPrositL-1991art

:= *стандартное библиографическое описание**:
[H. Schiitze, “The PROSIT Language v0.4,” *Manuscript, Center for the Study of Language and Information, Stanford University, Stanford, CA*, 1991]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Schlungbaum E.IndivUIaMU-1997art

:= *стандартное библиографическое описание**:
[E. Schlungbaum, “Individual User Interfaces and Model-Based User Interface Software Tools,” в *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, сер. IUI ’97, Orlando, Florida, USA: Association for Computing Machinery, 1997, с. 229—232, ISBN: 0897918398. DOI: 10.1145/238218.238330. url: https://doi.org/10.1145/238218.238330]

⇒ *аннотация**:
[Point of the paper is to use an additional user model to create individual user interfaces.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

Schuster S.A..RAP2aAPfDaIA-1979art

:= *стандартное библиографическое описание**:
[S. Schuster и др., “RAP.2 - an associative processor for databases and its applications,” *IEEE Trans. on Computers*, № 6, с. 446—458, 1979]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Semantic S.ScienPSRbK-2022el

:= *стандартное библиографическое описание**:
[S. Schoolar. “Scientific papers search results by keyword "Speech Technology"and Data Range filter 2017-2022.” (), url: https://www.semanticscholar.org/search?%5C%5Cyear%5C%5B0%5C%5D=2017&year%5C%5B1%5C%5D=2022&q=speech%5C%20technology&sort=relevance]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Scott D.LattiTDTaF-1972art

:= *стандартное библиографическое описание**:
[D. Scott, *Lattice Theory, Data Types and Formal Semantics, Formal Semantics of Programming Languages*. Prentice-Hall, Englewood Cliffs, NJ, 1972, с. 65—106]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Пункт 3.2.4.1. Синтаксис программ в ostis-системах*

Scott M.L.ProgrLP-2006bk

:= *стандартное библиографическое описание**:
[M. L. Scott, *Programming Language Pragmatics*. Morgan Kaufmann publications, 2006, с. 856]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *Пункт 3.2.4.1. Синтаксис программ в ostis-системах*

Sebesta R.W.Conce oPL-2012bk

:= *стандартное библиографическое описание**:
[R. W. Sebesta, *Concepts of Programming Languages*. 10th ed. — Pearson/Addison-Wesley, 2012, с. 814]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*
- *Пункт 3.2.4.1. Синтаксис программ в ostis-системах*

Sellitto G.tUnder tIoRoPC-2020art

:= *стандартное библиографическое описание**:
[G. Sellitto и др., “Toward Understanding the Impact of Refactoring on Program Comprehension,” в *29th International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2022, с. 1—12]

⇒ *аннотация**:
[Software refactoring is the activity associated with developers changing the internal structure of source code without modifying its external behavior. The literature argues that refactoring might have beneficial and harmful implications for software maintainability, primarily when performed without the support of automated tools. This paper continues the narrative on the effects of refactoring by exploring the dimension of program comprehension, namely the property that describes how easy it is for developers to understand source code. We start our investigation by assessing the basic unit of program comprehension, namely program readability. Next, we set up a large-scale empirical investigation – conducted on 156 open-source projects – to quantify the impact of refactoring on program readability. First, we mine refactoring data and, for each commit involving a refactoring, we compute (i) the amount and type(s) of refactoring actions performed and (ii) eight state-of-the-art program comprehension metrics. Afterwards, we build statistical models relating the various refactoring operations to each of the readability metrics considered to quantify the extent to which each refactoring impacts the metrics in either a positive or negative manner. The key results are that refactoring has a notable impact on most of the readability metrics considered.]

⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

SemanW-2022el

:= *стандартное библиографическое описание**:
[““Semantic Web”: W3C’s vision of the Web of linked data.” англ. Mode of access: `https://www.w3.org/standards/semanticweb/`. — Date of access: 18.10.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Serra X..aSystefSATS-1990art

:= *стандартное библиографическое описание**:
[X. Serra, “A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition,” Stanford University, 1990]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.3. Предметная область и онтология моделей параметрического представления сигнала*

Sethy S.S.MediaIS-2021art

:= *стандартное библиографическое описание**:
[S. S. Sethy, “Mediate Inference (Syllogism),” англ., в июнь 2021, с. 95—124, ISBN: 978-981-16-2688-3. DOI: `10.1007/978-981-16-2689-0_8`. url: `https://www.researchgate.net/publication/352359427`]

⇒ *аннотация**:
[In this article will be discussed mediate inference (i.e. syllogism) in detail with suitable examples. Authors will analyse the moods and figures of syllogism, describe the rules for the syllogism, and apply the rules of syllogism to each mood by linking them to figures of syllogism to find out the valid moods and valid arguments of the syllogism. In the end, they will list out the valid moods of the syllogism corresponding to the four figures of syllogism.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Shahmirzadi O..aTextSiVSM-2019art

:= *стандартное библиографическое описание**:
[O. Shahmirzadi, A. Lugowski и K. Younge, “Text similarity in vector space models: a comparative study,” в *2019 18th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, 2019, с. 659—666]

⇒ *аннотация**:
[В работе рассмотрен подход на основе VSM для вычисления подобия текстов]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*
- *Подпункт 7.5.3.1.2 Автоматическая проверка ответов пользователей*

Shamsfard M..LearnOfNLT-2004art

≔ *стандартное библиографическое описание**:
[M. Shamsfard и A. A. Barforoush, “Learning ontologies from natural language texts,” англ., *International Journal of Human-Computer Studies*, т. 60, № 1, с. 17—63, 2004, ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2003.08.001. url: https://www.sciencedirect.com/science/article/pii/S1071581903001368]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Sherif Y.S..CompuSDQAMaM-1988art
≔ *стандартное библиографическое описание**:
[Y. S. Sherif, E. Ng и J. Steinbacher, “Computer software development: Quality attributes, measurements, and metrics,” *Naval Research Logistics (NRL)*, т. 35, с. 425—436, 1988]
⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.3. Комплекс свойств, определяющий общий уровень качества кибернетической системы*

Shi X..GraphPoGPUs-2018art
≔ *стандартное библиографическое описание**:
[X. Shi и др., “Graph Processing on GPUs,” *ACM Computing Surveys*, т. 50, № 6, с. 1—35, нояб. 2018. DOI: 10.1145/3128571. url: https://doi.org/10.1145/3128571]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Shunkevich D.V..OntolAttDoaSM-2021art
≔ *стандартное библиографическое описание**:
[D. Shunkevich, D. Koronchik, “Ontological approach to the development of a software model of a semantic computer based on the traditional computer architecture,” англ., в *Open semantic technologies for intelligent systems*, BSUIR, Minsk, 2021, с. 75—92]
⇒ *аннотация**:
[The paper considers an ontological approach to the development of a software model of a platform for interpreting semantic models of intelligent computer systems (a software model of a semantic computer). The architecture of the specified software model and its components are considered in detail, the principles of their implementation and the advantages of the decisions made in comparison with analogues are indicated. A distinctive feature of the work is the demonstration of the usage of the ontological approach to the development of software products on the example of the specified software model of a semantic computer.]
⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем*

Shunkevich D.V.OntolDoHPS-2022art
≔ *стандартное библиографическое описание**:
[D. Shunkevich, “Ontology-Based Design of Hybrid Problem Solvers,” англ., в *Open semantic technologies for intelligent systems*, Cham: Springer International Publishing, 2022, с. 101—131, ISBN: 978-3-030-60446-2]
⇒ *аннотация**:
[The creation of problem solvers for intelligent computer systems based on the OSTIS Technology is discussed from an ontological perspective in this study. It is now possible to describe the ideas of a problem-solving model and a problem solver on the basis of the formal interpretation of terms like action, problem, class of actions, class of problems, method, and skill. The collected results will increase the effectiveness of the component approach to problem solving and automation tools for problem solving.]
⇐ *библиографическая ссылка**:
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*

Shunkevich.D.V.AgentMMaToC-2018art
≔ *стандартное библиографическое описание**:
[D. Shunkevich, “Agent-oriented models, method and tools of compatible problem solvers development for intelligent systems,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 2, BSUIR, Minsk, 2018, с. 119—132]
⇒ *аннотация**:

[Работа содержит описание агентно-ориентированных моделей, методов и средств разработки совместимых решателей задач интеллектуальных систем]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*
- *§ 4.2.3. Методика и средства разработки естественно-языковых интерфейсов*
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.2. Принципы, лежащие в основе, и структура предлагаемой Программной платформы ostis-систем*
- *Глава 5.3. Методика и средства компонентного проектирования решателей задач ostis-систем*
- *Глава 3.1. Формализация понятий действия, задачи, метода, средства, навыка и технологии*
- *§ 3.1.8. Понятие модели решения задач*

Siekmann J.UniveU-1984art

:= *стандартное библиографическое описание**:

[J. H. Siekmann, “Universal unification,” англ., в *International Conference on Automated Deduction*, Springer, 1984, с. 1—42]

⇒ *аннотация**:

[This article surveys what is presently known about first order unification theory.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*

Sims M..AutomODfMS-2008art

:= *стандартное библиографическое описание**:

[M. Sims, D. Corkill и V. Lesser, “Automated organization design for multi-agent systems,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 16, № 2, с. 151—185, июнь 2008]

⇒ *аннотация**:

[В статье рассматривается KB-ORG: полностью автоматизированный конструктор организаций, основанный на знаниях, для многоагентных систем.]

⇐ *библиографическая ссылка**:

- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Bhatia A..AutomGoMCQ-2013art

:= *стандартное библиографическое описание**:

[A. S. Bhatia, M. Kirti и S. K. Saha, “Automatic Generation of Multiple Choice Questions Using Wikipedia,” в *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, с. 733—738. DOI: 10.1007/978-3-642-45062-4_104. url: https://doi.org/10.1007/978-3-642-45062-4_104]

⇒ *аннотация**:

[Работа содержит описание использования Википедии для генерации вопросов на выбор]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Siri-2022el

:= *стандартное библиографическое описание**:

[“Siri.” англ. Mode of access: https://www.apple.com/siri/. — Date of access: 15.09.2022. ()]

⇐ *библиографическая ссылка**:

- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Smith S.M..SUSAN aNAtLL-1997art

:= *стандартное библиографическое описание**:

[S. Smith и J. Brady, “SUSAN—A New Approach to Low Level Image Processing,” *International Journal of Computer Vision*, т. 23, № 1, с. 45—78, 1997. DOI: 10.1023/a:1007963824710]

⇐ *библиографическая ссылка**:

- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

Smith S.tLMI LTS-1984art

:= *стандартное библиографическое описание**:

[S. Smith, *The LMI Lambda Technical Summary. Technical report, LMI Inc.* Los Angeles, CA, 1984]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Smorodin V..AdaptCoRPS-2019art

:= *стандартное библиографическое описание**:

[V. P. V. Smorodin, “Adaptive Control Of Robotic Production Systems,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2019, с. 161—166]

⇒ *аннотация**:

[The purpose of the work, that is presented in this paper, is to develop a method for adaptive control of a technological production cycle based on a software and hardware system that includes indicators of the hardware units states, parameters of the technological production cycle operation, simulation model of the probabilistic technological process and a built-in decision-making system. Operational interaction of the software and hardware system components and construction of the feedback control connections is implemented through the control parameters and variables of the simulation model based on the output of the neuroregulator model. To address the described problem, tasks related to implementation of the neural network technologies when constructing architecture and mathematical model of the neuroregulator were solved. The mathematical model of the neuroregulator is based on parameters of operation of the physical prototype, construction of the feedback connections for the real-time control (adaptive control) is based on the procedure of training of a recurrent neural network that includes LSTM-cells.]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Smorodin V..Appli oNMfAC-2019art

:= *стандартное библиографическое описание**:

[V. Smorodin и V.Prokhorenko, “Application of Neuro-Controller Models for Adaptive Control,” в *Recent Developments in Data Science and Intelligent Analysis of Information. ICDSIAI 2018. Advances in Intelligent Systems and Computing*, С. О. и др., ред., Cham. Springer, 2019, с. 30—38]

⇒ *аннотация**:

[In this paper, a method for constructing a model of a controller based on recurrent neural network architecture for implementation of control for the optimal trajectory finding problem is considered. A type of a neuro-controller based on recurrent neural network architecture with long short-term memory blocks as a knowledge base on the external environment and previous states of the controller is being proposed. The formalization of the technological cycle of a special type for adaptive control of a production process using the model of the neuro-controller is given.]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Smorodin V..Contr oaTCoPPB-2019art

:= *стандартное библиографическое описание**:

[V. P. V. Smorodin, “Control Of A Technological Cycle Of Production Process Based On A Neuro-Controller Model,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2019, с. 251—256]

⇒ *аннотация**:

[In this paper a method for constructing a model of a neuro-controller for implementation of control in the presence of external disturbances for the optimal trajectory finding on the phase plane of system states for technological cycle of a production process is proposed. A type of a neuro-controller based on recurrent neural networkarchitecture with long short-term memory blocks as a knowledge base on the external environment, previous states of the controller and control actions is being used.]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Smorodin V..SoftwCfACoaP-2022art

:= *стандартное библиографическое описание**:

[V. P. V. Smorodin, “Software-Technological Complex For Adaptive Control Of A Production Cycle Of Robotic Manufacturing,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2022, с. 401—404]

⇒ *аннотация**:

[An approach is being proposed for constructing a new generation intellectual system based on OSTIS technology for decision making during realization of adaptive control procedures for technological cycle of robotic manufacturing based on the means of software-hardware coupling. At the basis of the decision making intellectual system lays the idea of using neural network controllers that solve the task of searching for optimal maintenance strategy for a technological cycle of robotic manufacturing. A formalization of such a system is being proposed based on OSTIS technology implementation.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*
- *Подпункт 7.7.1.2.9 Примеры реализации систем адаптации управления*

SoftwIoSNS-el

:= *стандартное библиографическое описание**:
[*Software implementation of semantic network storage and processing [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/sc-machine`. — Date of access: 6.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем*

SoftwIoWOUI-el

:= *стандартное библиографическое описание**:
[*Software implementation of web-oriented user interfaces sc-models [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/sc-web`. — Date of access: 6.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.7. Реализация интерпретатора sc-моделей пользовательских интерфейсов*

SoftwIoSLP-el

:= *стандартное библиографическое описание**:
[*Software implementation of scn latex plugin [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/scn-latex-plugin`. — Date of access: 01.03.2023]

⇐ *библиографическая ссылка**:
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.4.3. Язык представления исходных текстов баз знаний на основе языка LaTeX*

SoftwIoStST-el

:= *стандартное библиографическое описание**:
[*Software implementation of scn-tex to scs translator [Electronic resource]*, англ., Mode of access: `https://github.com/ostis-ai/tex2scs-translator`. — Date of access: 01.03.2023]

⇐ *библиографическая ссылка**:
- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.4.3. Язык представления исходных текстов баз знаний на основе языка LaTeX*

Solovov A.V..Desig aOotECC-2023art

:= *стандартное библиографическое описание**:
[A. V. Solovov и A. A. Menshikova, “Designing an ontology of the e-learning course content,” англ., *Ontology of designing*, т. 13, № 1 (47), с. 99—112, март 2023. DOI: `10.18287/2223-9537-2023-13-1-99-112`]

⇐ *библиографическая ссылка**:
- *§ 7.5.1. Общие принципы автоматизации образовательной деятельности с помощью ostis-систем*

Somsubhra S.ReconSP-2006el

:= *стандартное библиографическое описание**:
[S. Somsubhra, *Reconfigurable semantic processor*, окт. 2006]

⇒ *аннотация**:
[Патент на реконфигурируемый семантический процессор.]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Son L.H..PictuISaNF-2017art

:= *стандартное библиографическое описание**:
[L. Son, P. Van Viet и P. Van Hai, “Picture inference system: a new fuzzy inference system on picture fuzzy set,” англ., *Applied Intelligence*, с. 652—669, 2017. url: https://doi.org/10.1007/s10489-016-0856-1]

⇒ *аннотация**:
[In this paper, authors propose a novel fuzzy inference system on picture fuzzy set called picture inference system (PIS) to enhance inference performance of the traditional fuzzy inference system. In PIS, the positive, neutral and negative degrees of the picture fuzzy set are computed using the membership graph that is the combination of three Gaussian functions with a common center and different widths expressing a visual view of degrees. Then, the positive and negative defuzzification values, synthesized from three degrees of the picture fuzzy set, are used to generate crisp outputs. Learning in PIS including training centers, widths, scales and defuzzification parameters is also discussed. The system is adapted for all architectures such as the Mamdani, the Sugeno and the Tsukamoto fuzzy inferences. Experimental results on benchmark UCI Machine Learning Repository datasets and an example in control theory - the Lorenz system are examined to verify the advantages of PIS.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Song W..NovelGPAPSaR-2016art

:= *стандартное библиографическое описание**:
[W. S. Song и др., “Novel graph processor architecture, prototype system, and results,” в *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, сент. 2016. DOI: 10.1109/hpec.2016.7761635. url: https://doi.org/10.1109/hpec.2016.7761635]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Sosnin P.QuestMfCDoS-2007art

:= *стандартное библиографическое описание**:
[P. Sosnin, “Question-answer means for collaborative development of software intensive systems,” в *Complex Systems Concurrent Engineering: Collaboration, Technology Innovation and Sustainability*, Springer, 2007, с. 151—158]

⇐ *библиографическая ссылка**:
- *Глава 3.4. Язык вопросов для ostis-систем*

Sowa.J.F.Top-lOC-1995art

:= *стандартное библиографическое описание**:
[J. F. Sowa, “Top-level ontological categories,” англ., *Intern. J. of Human-Computer Studies*, т. 43, № 5/6, с. 669—685, 1995]

⇒ *аннотация**:
[Категории верхнего уровня онтологии получены из контрастирующих признаков, которые отличают объекты предметной области. Каждая отличительная особенность связана с аксиомами, которые наследуются каждой сущностью или категорией сущностей, которые имеют эту особенность.]

⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*

SpaCyNL-2022el

:= *стандартное библиографическое описание**:
[“spaCy NLP library.” англ. Mode of access: https://spacy.io/. — Date of access: 24.10.2022. ()]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

SPARQLO-2023el

:= *стандартное библиографическое описание**:
[“SPARQL 1.1 Overview [Electronic resource],” англ., *World Wide Web Consortium*, Mode of access: https://www.w3.org/TR/sparql11-overview. — Date of access: 18.01.2023]

⇒ *аннотация**:
[Официальный сайт, описывающий язык SPARQL 1.1]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*

- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Stanley K..EvolvNNtAT-2002art

:= *стандартное библиографическое описание**:

[R. Stanley Kenneth O.; Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, т. 10(2), с. 99—127, 2002. DOI: 10.1162/106365602320169811]

⇒ *аннотация**:

[An important question in neuroevolution is how to gain an advantage from evolving neural network topologies along with weights. We present a method, NeuroEvolution of Augmenting Topologies (NEAT), which outperforms the best fixed-topology method on a challenging benchmark reinforcement learning task. We claim that the increased efficiency is due to (1) employing a principled method of crossover of different topologies, (2) protecting structural innovation using speciation, and (3) incrementally growing from minimal structure. We test this claim through a series of ablation studies that demonstrate that each component is necessary to the system as a whole and to each other. What results is significantly faster learning. NEAT is also an important contribution to GAs because it shows how it is possible for evolution to both optimize and complexify solutions simultaneously, offering the possibility of evolving increasingly complex solutions over generations, and strengthening the analogy with biological evolution.]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Steele G.L..ConneMLFP-1986art

:= *стандартное библиографическое описание**:

[J. Guy L. Steele и W. D. Hillis, "Connection Machine Lisp: fine-grained parallel symbolic processing," англ., в *Proceedings of the 1986 ACM conference on LISP and functional programming (LFP '86)*, ACM, New York, 1986, с. 279—297]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Stephen-2023el

:= *стандартное библиографическое описание**:

[*Stephen Wolfram. Books [Electronic resource]*, англ., Mode of access: http://www.stephenwolfram.com/publications/books/. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:

- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Stojkoska B.R..aRevie oIoTfSH-2017art

:= *стандартное библиографическое описание**:

[B. R. Stojkoska и K. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *Journal of Cleaner Production*, т. 140, с. 1454—1464, 2017]

⇒ *аннотация**:

[Although Internet of Things (IoT) brings significant advantages over traditional communication technologies for smart grid and smart home applications, these implementations are still very rare. Relying on a comprehensive literature review, this paper aims to contribute towards narrowing the gap between the existing state-of-the-art smart home applications and the prospect of their integration into an IoT enabled environment. We propose a holistic framework which incorporates different components from IoT architectures/frameworks proposed in the literature, in order to efficiently integrate smart home objects in a cloud-centric IoT based solution. We identify a smart home management model for the proposed framework and the main tasks that should be performed at each level. We additionally discuss practical design challenges with emphasis on data processing, as well as smart home communication protocols and their interoperability. We believe that the holistic framework ascertained in this paper can be used as a solid base for the future developers of Internet of Things based smart home solutions.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.4. Подсистемы умного дома*
- *Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью*

Strubell E..Energ aPCfDL-2019art

≔ *стандартное библиографическое описание**:
[E. Strubell, A. Ganesh и A. McCallum, “Energy and policy considerations for deep learning in NLP,” англ., *arXiv preprint arXiv:1906.02243*, 2019]
⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Studer.R..OntologCPSM-1996art
≔ *стандартное библиографическое описание**:
[R. Studer и др., *Ontologies and the Configuration of Problem-Solving Methods*, нояб. 1996]
⇒ *аннотация**:
[В работе говорится о необходимости создания библиотеки многократно используемых методов решения задач, упоминается проект PROTÉGÉ-II, включающий библиотеку таких многократно используемых методов. кратко говорится о структуре библиотеки многократно используемых компонентов систем, основанных на знаниях, о необходимости поиска нужного компонента в библиотеке, основное внимание уделяется проблеме конфигурирования компонентов.

Формулируется проблема совместимости многократно используемых методов в рамках единой системы, которая решается вручную разработчиком. Для упрощения процесса согласования (конфигурирования) множества многократно используемых методов предлагается использовать иерархию онтологий, описывающих методы решения задач различных классов и их подметоды (методы решения подзадач исходной задачи).

Метод делится на подзадачи (явно сказано, что задача и метод ее решения не противопоставляются), каждая из которых, в свою очередь, может делиться на более простые, вплоть до таких подзадач, для которых существует тривиальный ("зашитый") способ их решения. Такой способ в работе назван *элементарным логическим действием* (elementary inference action).

Тем не менее, в работе не дается описания какой-либо предметно-независимой общей онтологии методов в виде семейства классов и отношений. Предложенный подход иллюстрируется на примере онтологии для конкретного класса задач (простая настольная игра в виде поля с клетками, по которому перемещается фишка), формулируется проблема сопоставления (mapping) методов и решаемых ими классов задач. Говорится, что для описания онтологий используется формальный язык KARL (описан, например, в работе *Fensel.D..TheKnowledgeARL-1998art*), но не приводятся примеры описаний.

Основным результатом работы авторы считают введение понятия онтологии общего метода решения задач и онтологий подзадач (подметодов), что позволяет локализовать внимание разработчика на соответствующих частных онтологиях. Таким образом, речь в работе идет не о некоторой общей онтологии методов, а о самой идее разработки онтологий, соответствующих методам решения задач и формирования иерархических систем таких взаимосвязанных онтологий, соответствующих методам и их подметодам.]
⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Study E.KürzeWiKG-1905art
≔ *стандартное библиографическое описание**:
[E. Study, “Kürzeste Wege im komplexen Gebiet,” нем., *Mathematische Annalen (in German).*, т. 60, № 3, с. 321—378, 1905]
⇐ *библиографическая ссылка**:
- *Глава 2.2. Универсальный язык смыслового представления знаний в памяти ostis-систем — SC-код (Semantic Computer Code)*
- *§ 2.2.3. Смысловое пространство ostis-систем*

SuggestedUMO-2016el
≔ *стандартное библиографическое описание**:
[“Suggested Upper Merged Ontology (SUMO) [Electronic resource].” англ. Mode of access: `http://www.adampease.org/OP/`. — Date of access: 21.11.2016. ()]
⇒ *аннотация**:
[Официальный сайт Semantic Suggested Upper Merged Ontology (SUMO).]
⇐ *библиографическая ссылка**:
- *Глава 2.5. Структура баз знаний ostis-систем: иерархическая система предметных областей и соответствующих им онтологий*
- *§ 2.5.6. Онтологии верхнего уровня*

Sun Z..aBenchSoEEA-2020art
≔ *стандартное библиографическое описание**:

[Z. Sun и др., “A benchmarking study of embedding-based entity alignment for knowledge graphs,” англ., *arXiv preprint arXiv:2003.07743*, 2020]

⇒ *аннотация**:

[Работа содержит описание алгоритма выравнивания графа знаний на основе встраивания]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Sutton R.S..ReinfLaI-1998bk

:= *стандартное библиографическое описание**:

[R. S. Sutton и A. G. Barto, *Reinforcement Learning: An Introduction*, англ. The MIT Press, 1998]

⇒ *аннотация**:

[Reinforcement learning, one of the most active research areas in artificial intelligence, is a computational approach to learning whereby an agent tries to maximize the total amount of reward it receives when interacting with a complex, uncertain environment. In Reinforcement Learning, Richard Sutton and Andrew Barto provide a clear and simple account of the key ideas and algorithms of reinforcement learning. Their discussion ranges from the history of the field’s intellectual foundations to the most recent developments and applications. The only necessary mathematical background is familiarity with elementary concepts of probability.The book is divided into three parts. Part I defines the reinforcement learning problem in terms of Markov decision processes. Part II provides basic solution methods: dynamic programming, Monte Carlo methods, and temporal-difference learning. Part III presents a unified view of the solution methods and incorporates artificial neural networks, eligibility traces, and planning; the two final chapters present case studies and consider the future of reinforcement learning.]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*

SysteCCP-2023el

:= *стандартное библиографическое описание**:

[*SystemC Community Portal*, English, Mode of access: `https://systemc.org/`. — Date of access: 29.03.2023]

⇒ *аннотация**:

[Официальный сайт проекта по разработке языка SystemC – языка для системного проектирования, высокоуровневого синтеза, моделирования и проверки.]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*

Taberko V.V..Desig oBMEitC-2018art

:= *стандартное библиографическое описание**:

[V. Taberko и other, “Design of batch manufacturing enterprises in the context of Industry 4.0,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., сер. 2, BSUIR, Minsk, 2018, с. 307—320. url: `https://libeldoc.bsuir.by/handle/123456789/30398`]

⇒ *аннотация**:

[This paper presents an evolution of an ontology-based approach to design batch manufacturing enterprises]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*

Taberko V..OntolAtBEwI-2022art

:= *стандартное библиографическое описание**:

[V. Taberko и D. Ivaniuk, “Ontological approach to batch enterprise within Industry 4.0,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2022, с. 395—400. url: `https://libeldoc.bsuir.by/handle/123456789/49386`]

⇐ *библиографическая ссылка**:

- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.2.1.3 Проблемы стандартизации в области производственной деятельности*

Taberko V..OntolAfSDwI-2020art

:= *стандартное библиографическое описание**:

[V. Taberko, D. Ivaniuk и др., “Ontological Approach for Standards Development within Industry 4.0,” англ., в *Communications in Computer and Information Science*, Springer International Publishing, 2020, с. 64—80. DOI: `10.1007/978-3-030-60447-9_5`. url: `https://doi.org/10.1007/978-3-030-60447-9_5`]

⇒ *аннотация**:
[In this paper we propose an approach to automating the processes of creating, developing and applying standards based on OSTIS Technology. The problems of modern approaches to the development, maintenance and application of standards are considered in detail, special attention is paid to standards in the field of Industry 4.0, such as ISA-88 and ISA-95, their role in the context of Industry 4.0 and problems specific to standards in this field are considered. The paper proposes an approach to the development of standards based on the ontological approach and involving the transformation of the standard into a knowledge base developed by a distributed team of developers directly in the process of its use. It is proposed to use OSTIS Technology as the basis for building this kind of system. We consider a prototype information system for employees of a batch production enterprise that implements the proposed approach, as well as examples of the integration of such a system with production systems.]
⇐ *библиографическая ссылка**:
- *§ 7.2.2. Структура, назначение, особенности и достоинства Стандарта OSTIS*

Taberko V..Princ fEtDaUoS-2020art
:= *стандартное библиографическое описание**:
[V. Taberko и other, “Principles for enhancing the development and use of standards within Industry 4.0,” англ., в *Open semantic technologies for intelligent systems*, V. Golenkov, ред., BSUIR, Minsk, 2020, с. 167—174. url: `https://libeldoc.bsuir.by/handle/123456789/38690`]
⇒ *аннотация**:
[In this paper, we propose an approach to automating the processes of creating, developing and applying standards based on OSTIS Technology. The problems of modern approaches to the organization of these processes are considered, the role of standards in the framework of the Industry 4.0 concept is shown, examples of formalization of standards within the framework of the proposed approach are given.]
⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.2 Создание математической модели производственной системы в рамках концепции Industry 4.0*

TextEI-2022el
:= *стандартное библиографическое описание**:
[*Text Encoding Initiative*, Available at: `https://tei-c.org/`, (accessed 2022, October)]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

tGenerOoLD-2022el
:= *стандартное библиографическое описание**:
[*The General Ontology of Linguistic Description*, Available at: `http://linguistics-ontology.org/info/about`, (accessed 2022, October)]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Tin ..TowarSOPL-1995art
:= *стандартное библиографическое описание**:
[E. Tin, V. Akman и M. Ersan, “Towards situation-oriented programming languages,” *ACM Sigplan Notices*, т. 30, № 1, с. 27—36, 1995]
⇒ *аннотация**:
[Recently, there have been some attempts towards developing programming languages based on situation theory. These languages employ situation-theoretic constructs with varying degrees of divergence from the ontology of the theory. In this paper, we review three of these programming languages.]
⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*

Tomasseti M.aAnaly otPoWiVP-2021art
:= *стандартное библиографическое описание**:
[M. Tomasetti, “An Analysis of the Performance of Websockets in Various Programming Languages and Libraries,” англ., *Available at SSRN 3778525*, 2021]
⇒ *аннотация**:
[As the demand for real-time data increases, so does the use of websockets. It is crucial to consider the speed along with the reliability of a language and websocket library before implementing it in an application. This study aims to

benchmark various websocket servers in order to determine which one offers the fastest round trip time of a request, as well as the reliability of the websocket server under load.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.6. Реализация подсистемы сетевого взаимодействия с sc-памятью на основе языка JSON в ostis-платформе*

Trajanov ..Surve oNLPiPMT-2022art

≔ *стандартное библиографическое описание**:
[D. Trajanov и др., “Survey of NLP in Pharmacology: Methodology, Tasks, Resources, Knowledge, and Tools,” англ., авг. 2022]

⇐ *библиографическая ссылка**:
- *Глава 4.2. Естественно-языковые интерфейсы ostis-систем*

Tran H-N.. aSurve oGPoGPU-2018art

≔ *стандартное библиографическое описание**:
[H.-N. Tran и E. Cambria, “A survey of graph processing on graphics processing units,” *The Journal of Supercomputing*, т. 74, № 5, с. 2086—2115, янв. 2018. DOI: 10.1007/s11227-017-2225-1. url: https://doi.org/10.1007/s11227-017-2225-1]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

tSemanRoPM-2023el

≔ *стандартное библиографическое описание**:
[*The Semantic Representation of Pure Mathematics [Electronic resource]*, Mode of access: https://blog.wolfram.com/2016/12/22/the-semantic-representation-of-pure-mathematics/. — Date of access: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Samson W.T..OntolBCoPS-1995art

≔ *стандартное библиографическое описание**:
[S. W. Tu и др., “Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: application of PROTÉGÉ-II to protocol-based decision support,” т. 7, № 3, с. 257—289, июнь 1995. DOI: 10.1016/0933-3657(95)00006-r. url: https://doi.org/10.1016/0933-3657(95)00006-r]

⇒ *аннотация**:
[Достаточно подробно рассмотрена работа средств семейства PROTEGE II при разработке конкретной системы, обеспечивающей поддержку принятия решений на основе протоколов в области лечения ВИЧ-инфицированных пациентов. Для решения поставленной задачи используется метод, предполагающий декомпозицию задачи на подзадачи, который строится из многократно используемых компонентов.

Кроме того, рассматривается прикладная онтология, уточняющая систему понятий предметной области, а также онтологии, описывающие знанияя, специфичные для конкретного метода. На основе этой онтологии строится инструмент извлечения знаний, специфичный для предметной области.

Общая цель подхода PROTÉGÉ-II состоит в том, чтобы производить системы и компоненты, которые легко поддерживать и использовать повторно. По этой причине методы решения задач строятся из набора более мелких методов и механизмов. Это также объясняет, почему инструменты извлечения знаний привязаны к предметной области и автоматически генерируются на основе онтологий. Оценка успешности предложенных решений все еще является предварительной, тем не менее в работе показано, как указанных цели могут быть достигнуты для прикладной задачи обеспечения поддержки принятия решений.

Близкие работы: *Studer.R..OntologCPSM-1996art*.

Работа содержит много ссылок на другие работы, описывающие архитектуры, в которых методы решения задач строятся из более мелких многократно используемых компонентов.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*

Turner ..Towar aPLO-2007bk

:= *стандартное библиографическое описание**:
[R. Turner и A. H. Eden, *Towards a programming language ontology*. na, 2007]
⇒ *аннотация**:
[We examine the role of semantic theory in determining the ontology of programming languages. We explore how different semantic perspectives result in different ontologies. In particular, we compare the ontological implications of set-theoretic versus type-theoretic semantics.]
⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*
- *§ 3.2.2. Существующие онтологии языков программирования*

Turner ..ProgrLaTA-2014art
:= *стандартное библиографическое описание**:
[R. Turner, "Programming languages as technical artifacts," *Philosophy & technology*, т. 27, № 3, с. 377—397, 2014]
⇒ *аннотация**:
[Taken at face value, a programming language is defined by a formal grammar. But, clearly, there is more to it. By themselves, the naked strings of the language do not determine when a program is correct relative to some specification. For this, the constructs of the language must be given some semantic content. Moreover, to be employed to generate physical computations, a programming language must have a physical implementation. How are we to conceptualize this complex package? Ontologically, what kind of thing is it? In this paper, we shall argue that an appropriate conceptualization is furnished by the notion of a technical artifact.]
⇐ *библиографическая ссылка**:
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.2. Существующие онтологии языков программирования*

Uehara ..FuzzyIIPaP-2017art
:= *стандартное библиографическое описание**:
[K. Uehara и K. Hirota, "Fuzzy Inference: Its Past and Prospects," англ., *Journal of Advanced Computational Intelligence and Intelligent Informatics*, т. 21, с. 13—19, янв. 2017. DOI: 10.20965/jaciii.2017.p0013]
⇒ *аннотация**:
[Fuzzy inference in the past and its future prospects are described to further promote research in the field: First, the basic methods of fuzzy inference are introduced. Then, the progress of fuzzy inference is reviewed, showing its remarkable achievements, especially in industries. A consideration of fuzzy inference is presented from operational viewpoints. It provides a key to creating fuzzy-inference methods in the future. The growing research area of fuzzy inference is also introduced in order to discuss a current direction, reflecting the consideration mentioned above. Moreover, some future prospects on fuzzy inference are presented, which are expected to stimulate research.]
⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

USBA-2022el
:= *стандартное библиографическое описание**:
["USB Accelerator | Coral." англ. (дек. 2022), url: https://coral.ai/products/accelerator/]
⇒ *аннотация**:
[Официальный сайт USB-ускорителя Coral, обеспечивающего аппаратное ускорение процессов машинного обучения и построенного на базе сопроцессора Google Edge TPU.]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*
- *Глава 6.1. Универсальная модель интерпретации логико-семантических моделей ostis-систем*
- *§ 6.1.1. Уточнение понятия платформенной независимости и анализ современных подходов к ее обеспечению*

Valdez O.How tDaDEaPF-2019art
:= *стандартное библиографическое описание**:
[O. Valdez-De-Leon, "How to develop a digital ecosystem: A practical framework," англ., *Technology Innovation Management Review*, т. 9, № 8, 2019]
⇒ *аннотация**:
[Рассмотрено понятие цифровой экосистемы, предложен фреймворк для создания цифровых экосистем.]
⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

- *§ 7.4.1. Общие принципы интеграции Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *Пункт 7.4.1.1. Принципы интеграции Экосистемы OSTIS с разнородными сервисами*

Van B..KnowlSiaENoP-2005art
:= *стандартное библиографическое описание**:
[P. Van Baalen, J. Bloemhof-Ruwaard и E. Van Heck, “Knowledge Sharing in an Emerging Network of Practice:: The Role of a Knowledge Portal,” *European Management Journal*, т. 23, № 3, с. 300—314, 2005]
⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.2. Семантически совместимые интеллектуальные ostis-порталы знаний*

van der Leun V.Intro tJVML-2017bk
:= *стандартное библиографическое описание**:
[V. van der Leun, *Introduction to JVM Languages*, англ. Packt Publishing, июнь 2017]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Vasconcelos W.W..NormaCRiMA-2009art
:= *стандартное библиографическое описание**:
[W. W. Vasconcelos, M. J. Kollingbaum и T. J. Norman, “Normative conflict resolution in multi-agent systems,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 19, № 2, с. 124—152, июнь 2009]
⇒ *аннотация**:
[В статье предлагается подход к регулированию поведения агентов на основе неких утвержденных "законов".]
⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

VerbNet-2022el
:= *стандартное библиографическое описание**:
[“VerbNet: A Computational Lexical Resource for Verbs.” англ. (окт. 2022), url: `https://verbs.colorado.edu/verbnet/`]
⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

Verdie Y..TILDE aTILD-2015art
:= *стандартное библиографическое описание**:
[Y. Verdie и др., “TILDE: A Temporally Invariant Learned DEtector,” в *2015 IEEE Conference on Computer Vision and Pattern Recognition CVPR*, июнь 2015. DOI: `10.1109/cvpr.2015.7299165`]
⇐ *библиографическая ссылка**:
- *Глава 4.4. 3D-модель внешней среды в ostis-системах*
- *Пункт 4.4.8.2. Локальные признаки изображений*

VHDPl-2023el
:= *стандартное библиографическое описание**:
[*VHDPlus*, English, Mode of access: `https://vhdplus.com/`. — Date of access: 29.03.2023]
⇒ *аннотация**:
[Официальный сайт проекта VHDPlus – инструмента для работы с FPGA при помощи набора высокоуровневых языков и IDE.]
⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *Пункт 6.2.4.1. Архитектура ассоциативного семантического компьютера на базе фон-Неймановской архитектуры*

Vicknair C..aCompa oaGDaaRD-2010art
:= *стандартное библиографическое описание**:
[C. Vicknair и др., “A comparison of a graph database and a relational database: a data provenance perspective,” англ., в *Proceedings of the 48th annual Southeast regional conference*, 2010, с. 1—6]
⇒ *аннотация**:

[Relational databases have been around for many decades and are the database technology of choice for most traditional data-intensive storage and retrieval applications. Retrievals are usually accomplished using SQL, a declarative query language. Relational database systems are generally efficient unless the data contains many relationships requiring joins of large tables. Recently there has been much interest in data stores that do not use SQL exclusively, the so-called NoSQL movement. Examples are Google's BigTable and Facebook's Cassandra. This paper reports on a comparison of one such NoSQL graph database called Neo4j with a common relational database system, MySQL, for use as the underlying technology in the development of a software system to record and query data provenance information.]

⇐ *библиографическая ссылка**:
- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*

Volkel S..What iIiIUI-2020art

:= *стандартное библиографическое описание**:
[S. Volkel и др., "What is "Intelligent"in Intelligent User Interfaces? A Meta-Analysis of 25 Years of IUI," сер. IUI '20, New York, NY, USA: Association for Computing Machinery, 2020, с. 477—487. DOI: 10.1145/3377325.3377500. url: https://doi.org/10.1145/3377325.3377500]

⇒ *аннотация**:
[Библиотеки подпрограмм создаются и используются уже много лет, однако, и сейчас в этой области существуют еще не решенные проблемы. В качестве одной из форм организации вычислений на ЭВМ авторами предлагалось использование библиотеки подпрограмм, и уже тогда была показана эффективность такого метода программирования.]

⇐ *библиографическая ссылка**:
- *Глава 4.1. Общие принципы организации интерфейсов ostis-систем*
- *§ 4.1.1. Анализ и проблемы существующих принципов организации интерфейсов*

VOSK-2022el

:= *стандартное библиографическое описание**:
["VOSK Offline Speech Recognition API." (), url: https://alphacephei.com/vosk/]

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *4.3. Введение в Главу 4.3.*

Wan C..aRevie oTSCM-2019art

:= *стандартное библиографическое описание**:
[C. Wan, Y. Yang и F. Deng, "A review of text similarity calculation methods," *Information science*, т. 37, № 33, с. 158—168, 2019]

⇒ *аннотация**:
[В работе представлен метод вычисления подобия текстов]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Wan HR..RevieoRPoTS-2019art

:= *стандартное библиографическое описание**:
[H. Wan и Y. Zhang, "Review on Research Progress of Text Similarity Calculation," англ., *Journal of Beijing Information Science (Technology University)*, т. 34, № 01, с. 68—74, 2019]

⇒ *аннотация**:
[Работа содержит обзор хода исследований по вычислению подобия текстов]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Waters J..GlobaIUSS-2009art

:= *стандартное библиографическое описание**:
[Jeff Waters and Brenda J. Powers and Marion G. Ceruti, "Global Interoperability Using Semantics, Standards, Science and Technology (GIS3T)," *Computer Standards & Interfaces*, т. 31, № 6, с. 1158—1166, 2009]

⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*

- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*

Watkins C..QL-1992art

:= *стандартное библиографическое описание**:
[C. Watkins и P. Dayan, “Q-learning,” *Machine Learning*, т. 8, с. 279—292, 1992]

⇒ *аннотация**:
[Q-learning (Watkins, 1989) is a simple way for agents to learn how to act optimally in controlled Markovian domains. It amounts to an incremental method for dynamic programming which imposes limited computational demands. It works by successively improving its evaluations of the quality of particular actions at particular states.This paper presents and proves in detail a convergence theorem forQ-learning based on that outlined in Watkins (1989). We show thatQ-learning converges to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. We also sketch extensions to the cases of non-discounted, but absorbing, Markov environments, and where manyQ values can be changed each iteration, rather than just one.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.6 Алгоритмы адаптации управления технологическим циклом при использовании нейросетевого моделирования*

Weydert E.DefauLaPaTP-2022art

:= *стандартное библиографическое описание**:
[E. Weydert, “Defaults, Logic and Probability – A theoretical perspective,” англ., *KI – Künstliche Intelligenz, v.4/01, 44–49 (2001)*, дек. 2022]

⇒ *аннотация**:
[The complexity of the real world and the restricted availability of knowledge call for powerful logical frameworks to deal with uncertainty, reaching from qualitative default formalisms to quantitative probability logics. In particular, reasoning under uncertainty requires defeasible inference notions. Author takes a general look at these issues, with a special emphasis on quasi-probabilistic approaches to default reasoning.]

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Weyns D..Envir aaFCAiM-2007art

:= *стандартное библиографическое описание**:
[D. Weyns, A. Omicini и J. Odell, “Environment as a first class abstraction in multiagent systems,” англ., *Autonomous Agents a. Multi-Agent Systems*, т. 14, № 1, с. 5—30, февр. 2007]

⇒ *аннотация**:
[Обзор сред взаимодействия в МАС.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

Xu G.P..Resea oITS-2009art

:= *стандартное библиографическое описание**:
[G. P. Xu, W. H. Zeng и C. L. Huang, “Research on intelligent tutoring system,” англ., *Application research of computers*, т. 26, № 11, с. 4020—4030, нояб. 2009]

⇒ *аннотация**:
[В работе представлено понятие интеллектуальных обучающих систем и связанных с ними технологий]

⇐ *библиографическая ссылка**:
- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

What-2016el

:= *стандартное библиографическое описание**:
[“What is WordNet? [Electronic resource].” англ. Mode of access: `https://wordnet.princeton.edu/`. — Date of access: 25.11.2016. ()]

⇒ *аннотация**:
[Официальный сайт WordNet.]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*

White D.A..Handb oICNF-1992bk

:= *стандартное библиографическое описание**:
[D. White и D. Sofge, *Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches*, англ. Van Nostrand Reinhold, 1992]

⇒ *аннотация**:
[Handbook of Intelligent Control provides a hands-on approach to integrating new intelligent control techniques with existing control techniques for the nonlinear control of complex multivariate systems.]

⇐ *библиографическая ссылка**:
- *Глава 7.7. Автоматизация производственной деятельности в рамках Экосистемы OSTIS*
- *Подпункт 7.7.1.2.3 Разработка моделей нейрорегуляторов для решения задач адаптивного управления в условиях Экосистемы OSTIS*

Wilkes M.V.Prepa oPfaEDC-1951bk

:= *стандартное библиографическое описание**:
[M. Wilkes, *The Preparation of Programs for an Electronic Digital Computer: With Special Reference to the EDSAC and the Use of a Library of Subroutines* (Addison-Wesley mathematics series), англ. Addison-Wesley Press, 1951. url: `https://books.google.by/books?id=PzVVAAAAMAAJ`]

⇒ *аннотация**:
[Ее часто считают первой книгой по компьютерному программированию. Она была написана для компьютера EDSAC (автоматический калькулятор электронного хранения с задержкой), который начал работать в 1949 году как первый в мире регулярно работающий компьютер с хранимой программой. Идея библиотеки подпрограмм была разработана для EDSAC и описана в этой книге. Морис Уилкс руководил разработкой EDSAC.]

⇐ *библиографическая ссылка**:
- *Глава 5.1. Комплексная библиотека многократно используемых семантически совместимых компонентов ostis-систем*
- *§ 5.1.1. Анализ современных библиотек многократно используемых компонентов*

Winter Y.Eleme oFS-2016bk

:= *стандартное библиографическое описание**:
[Y. Winter, *An Introduction to the Mathematical Theory of Meaning in Natural Language*, англ. Edinburgh: Edinburgh University Press, 2016, ISBN: 9780748677771. DOI: `doi:10.1515/9780748677771`. url: `https://doi.org/10.1515/9780748677771`]

⇐ *библиографическая ссылка**:
- *Глава 2.7. Языковые средства формального описания синтаксиса и денотационной семантики естественных языков в ostis-системах*
- *§ 2.7.2. Формализация денотационной семантики естественных языков*

Wolfram S.NewKoS-2002bk

:= *стандартное библиографическое описание**:
[S. Wolfram, *New kind of science*, англ. 2002]

⇐ *библиографическая ссылка**:
- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.2. Анализ существующих архитектур вычислительных систем*

Wolfram-2018el

:= *стандартное библиографическое описание**:
[*Wolfram Alpha [Electronic resource]*, англ., Mode of access: `http://www.wolframalpha.com/`. — Date of access: 22.05.2018]

⇒ *аннотация**:
[Официальный сайт вопросно-ответной системы, базы знаний и набора вычислительных алгоритмов WolframAlpha]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*
- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.3. Пример интеграции прототипа обучающей ostis-системы по дискретной математике и Wolfram Mathematica*

Wolfram-2023эл

:= *стандартное библиографическое описание**:
[*Wolfram Mathematica. Наиболее полная система для современных технических вычислений в мире [Электронный ресурс]*, Режим доступа: `http://www.wolfram.com/mathematica`. — Дата доступа: 8.04.2023]

⇐ *библиографическая ссылка**:
- *Глава 7.4. Интеграция Экосистемы OSTIS с современными сервисами и информационными ресурсами*

- *§ 7.4.2. Интеграция инструментов компьютерной алгебры в приложения OSTIS*
- *Пункт 7.4.2.2. Сравнительный обзор наиболее известных систем компьютерной алгебры*

Wooldridge M.aIntro tMS-2009bk

:= *стандартное библиографическое описание**:
[M. Wooldridge, *An introduction to multiagent systems*, англ., 2nd. Chichester: J. Wiley, 2009]

⇒ *аннотация**:
[Основные цели книги:

- □ Познакомить учащихся с понятием агента и многоагентной системы, а также с основными приложениями, для которых они подходят.
- □ Познакомить с основными проблемами, связанными с проектированием интеллектуальных агентов.
- □ Познакомить с основными проблемами, связанными с проектированием многоагентных сообществ.
- □ Представить ряд типичных приложений для агентной технологии.

]

⇐ *библиографическая ссылка**:
- *§ 7.9.2. Библиография OSTIS*
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.2. Предлагаемый подход к разработке гибридных решателей задач ostis-систем и обработке информации в ostis-системах*

World-2018el

:= *стандартное библиографическое описание**:
[“World Economic Forum, Digital Transformation Initiative.” (окт. 2018), url: `http://reports.weforum.org/digital-transformation/wp-content/blogs.dir/94/mp/files/pages/files/dti-executive-summary-website-version.pdf`]

⇐ *библиографическая ссылка**:
- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.1. Анализ существующих подходов к созданию умных домов*

World-2023el

:= *стандартное библиографическое описание**:
[“World Wide Web Consortium [Electronic resource].” англ. Mode of access: `http://www.w3.org`. — Date of access: 18.01.2023. ()]

⇒ *аннотация**:
[Официальный сайт консорциума W3C.]

⇐ *библиографическая ссылка**:
- *Глава 3.3. Агентно-ориентированные модели гибридных решателей задач ostis-систем*
- *Пункт 3.3.1.1. Современное состояние технологий разработки решателей задач и требования, предъявляемые к гибридным решателям задач*

Yaghoobirafi K..aAppro fSIiADIS-2022art

:= *стандартное библиографическое описание**:
[K. Yaghoobirafi и A. Farahani, “An approach for semantic interoperability in autonomic distributed intelligent systems,” *Journal of Software: Evolution and Process*, т. 34, февр. 2022. DOI: `10.1002/smr.2436`]

⇐ *библиографическая ссылка**:
- *Глава 7.1. Структура и проблемы организации и комплексной автоматизации человеческой деятельности*
- *Глава 1.2. Интеллектуальные компьютерные системы нового поколения*
- *§ 1.2.1. Требования, предъявляемые к интеллектуальным компьютерным системам нового поколения*

Yuxuan Z..MissiEAKGIItDGLaT-2022art

:= *стандартное библиографическое описание**:
[Z. Yuxuan и др., “Missing-edge aware knowledge graph inductive inference through dual graph learning and traversing,” англ., в окт. 2022. DOI: `10.1016/j.eswa.2022.118969`. url: `https://www.researchgate.net/publication/364404289`]

⇒ *аннотация**:
[Knowledge graph (KG) is a kind of structured human knowledge of modeling the relations between real-world entities. This paper studies the KG inductive inference problem, i.e., predicting the relations for out-of-KG entities.

However, due to the incomplete nature of the KGs, the connections of some relations are missing. This makes existing differentiable rule learning methods unable to represent some possible rule candidates, which will further affect the inductive inference result. To solve this challenge, author's research hypothesis is that the semantics of relation's argument can be well used to reflect the possible connections between relations. There is proposed a KG inductive inference model, RuleNet, which consists of two parts. Firstly, a query-dependent dual graph construction method is proposed, which is able to learn the relation connections using the information of the relation's argument. Secondly, a dual graph traversing method is proposed, which is able to traverse all possible rule candidates even if some rules cannot be formed due to the missing edges. Performance of the proposed methods is evaluated using the FB15K237 (10%–20%), WN18RR (10%–20%) and YAGO3-10 (10%–20%) benchmarks. Experimental results show that RuleNet achieves a superior performance compared with many strong baselines. Ablation studies have verified the effectiveness of the proposed network components. Qualitative analysis shows that RuleNet can learn meaningful dual graph and logic rules.**]**

⇐ *библиографическая ссылка**:
- *Глава 3.5. Логические, продукционные и функциональные модели решения задач в ostis-системах*
- *§ 3.5.1. Операционная семантика логических языков, используемых ostis-системами*

Zagorskiy A..Facto tDtLoIoCS-2022art

:= *стандартное библиографическое описание**:
[A. Zagorskiy, "Factors that determine the level of intelligence of cybernetic systems," 2022**]**

⇐ *библиографическая ссылка**:
- *§ 1.1.1. Понятие интеллектуальной кибернетической системы*
- *Пункт 1.1.1.8. Комплекс свойств, определяющих уровень интеллекта кибернетической системы*

Zagorskiy A..Princ fItEoNGICS-2022art

:= *стандартное библиографическое описание**:
[A. Zagorskiy, "Principles for implementing the ecosystem of next-generation intelligent computer systems," 2022**]**

⇐ *библиографическая ссылка**:
- *Глава 7.3. Структура Экосистемы OSTIS*
- *§ 7.3.1. Иерархическая система взаимодействующих ostis-сообществ*

Zahariev V.A..ConveSABotFRotML-2021cm

:= *стандартное библиографическое описание**:
[V. A. Zahariev и др., "Conversational speech analysis based on the formalized representation of the mental lexicon," англ., в *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, ред., BSUIR, Minsk, 2021, с. 141—158**]**

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*

Zahariev V.A..IntelVABoOST-2020art

:= *стандартное библиографическое описание**:
[V. Zahariev и др., "Intelligent Voice Assistant Based on Open Semantic Technology," в *Open semantic technologies for intelligent systems*, V. Golenkov и др., ред., Cham. Springer, 2020, с. 121—145. DOI: `https://doi.org/10.1007/978-3-030-60447-9_8`**]**

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*

Zahariev V.A..SemanAoVMBoaFC-2019cm

:= *стандартное библиографическое описание**:
[V. A. Zahariev и др., "Semantic analysis of voice messages based on a formalized context," англ., в *Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh system [Open semantic technologies for intelligent systems]*, V. Golenkov, ред., BSUIR, Minsk, 2019, с. 103—112**]**

⇐ *библиографическая ссылка**:
- *Глава 4.3. Аудиоинтерфейс ostis-систем*
- *§ 4.3.1. Применение принципов онтологического проектирования при разработке аудиоинтерфейсов*

Zapata J..Ontol iSEATGKA-2010art

:= *стандартное библиографическое описание**:
[C. M. Zapata Jaramillo, G. L. Giraldo и G. A. Urrego Giraldo, "Ontologies in software engineering: approaching two great knowledge areas," *Revista Ingenierias Universidad de Medellin*, т. 9, № 16, с. 91—99, 2010**]**

⇒ *аннотация**:

[Ontology concepts have been traditionally linked to knowledge engineering and software engineers have not applied them to solve problems of this area. It is necessary that software engineers use these ontologies, since they provide a common language, which can contribute to the solution of some common software engineering problems like difficulties in communication between the analyst and the interested person in order to define a system requirements, the low components re-use, and scarce automatic generation in code generation, among others. In this paper, a first encounter between ontologies and software engineering by means of a state-of-the-art analysis related to the use of ontologies in several phases of software development life cycle is presented.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *§ 6.3.1. Существующие подходы к проектированию систем автоматизации проектирования и реализации программных компьютерных систем*
- *Глава 3.2. Семантическая теория программ для ostis-систем*
- *§ 3.2.1. Проблемы текущего состояния в области разработки и применения языков программирования*

Zeng K..aComprSoEAfKG-2021art

≔ *стандартное библиографическое описание**:

[K. Zeng и др., “A comprehensive survey of entity alignment for knowledge graphs,” англ., *AI Open*, т. 2, с. 1—13, 2021]

⇒ *аннотация**:

[Работа содержит описание алгоритма согласования графа знаний]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Zhang J..AutomPoTPfTiEGItHIPT-1995art

≔ *стандартное библиографическое описание**:

[J.-Z. Zhang, S.-C. Chou и X.-S. Gao, “Automated production of traditional proofs for theorems in Euclidean geometry I. The Hilbert intersection point theorems,” англ., *Annals of Mathematics and Artificial Intelligence*, т. 13, № 1, с. 109—137, 1995]

⇒ *аннотация**:

[Работа содержит описание процесса автоматического доказательства геометрических вопросов]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Jialiang Z..Boost tPoFPGABGPUHMCaCfBFS-2017art

≔ *стандартное библиографическое описание**:

[J. Zhang, S. Khoram и J. J. Li, “Boosting the Performance of FPGA-based Graph Processor using Hybrid Memory Cube: A Case for Breadth First Search,” *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017]

⇐ *библиографическая ссылка**:

- *Глава 6.2. Ассоциативные семантические компьютеры для ostis-систем*
- *§ 6.2.1. Современное состояние работ в области разработки компьютеров для интеллектуальных систем*

Zhang J..Self-EAPBoPGftPoWMI-2019art

≔ *стандартное библиографическое описание**:

[J. Zhang, X. Peng и M. Chen, “Self-evident automated proving based on point geometry from the perspective of Wu’s method identity,” англ., *Journal of Systems Science and Complexity*, т. 32, № 1, с. 78—94, 2019]

⇒ *аннотация**:

[Работа содержит описание процесса автоматического доказательства геометрических вопросов]

⇐ *библиографическая ссылка**:

- *Глава 7.5. Автоматизация образовательной деятельности в рамках Экосистемы OSTIS*
- *§ 7.5.3. Семантические модели и средства контроля знаний пользователей в ostis-системах*

Zhang D.KnowlBVIaA-2023art

≔ *стандартное библиографическое описание**:

[D. Zhang, “Knowledge base verification: issues and approaches,” март 2023]

⇐ *библиографическая ссылка**:

- *Глава 5.2. Методика и средства проектирования и анализа качества баз знаний ostis-систем*
- *§ 5.2.4. Логико-семантическая модель ostis-системы обнаружения и анализа ошибок и противоречий в базе знаний ostis-системы*

Zhmyrko A.Famil oELoNGCSCttLoISRoK-2022art

:= *стандартное библиографическое описание**:

[A. Zhmyrko, “Family of external languages of next-generation computer systems, close to the language of the internal semantic representation of knowledge,” англ., с. 65—80, 2022]

⇒ *аннотация**:

[In the article, the concepts of external and internal languages of next-generation intelligent computer systems are considered. External languages of knowledge representation within the OSTIS Technology are described, namely SCg-code, SCs-code, SCn-code. For each of the external languages, its syntax and denotational semantics are considered in detail.]

⇐ *библиографическая ссылка**:

- *Глава 2.3. Семейство внешних языков ostis-систем, близких языку внутреннего смыслового представления знаний*
- *Пункт 2.3.4.2. Денотационная семантика SCn-кода*

Zhou B..SmartHEMSCCaSS-2016art

:= *стандартное библиографическое описание**:

[B. Zhou и др., “Smart home energy management systems: Concept, configurations, and scheduling strategies,” *Renewable and Sustainable Energy Reviews*, т. 61, № C, с. 30—40, 2016. DOI: 10.1016/j.rser.2016.03.04. url: https://ideas.repec.org/a/eee/rensus/v61y2016icp30-40.html]

⇒ *аннотация**:

[With the arrival of smart grid era and the advent of advanced communication and information infrastructures, bidirectional communication, advanced metering infrastructure, energy storage systems and home area networks would revolutionize the patterns of electricity usage and energy conservation at the consumption premises. Coupled with the emergence of vehicle-to-grid technologies and massive distributed renewable energy, there is a profound transition for the energy management pattern from the conventional centralized infrastructure towards the autonomous responsive demand and cyber-physical energy systems with renewable and stored energy sources. Under the sustainable smart grid paradigm, the smart house with its home energy management system (HEMS) plays an important role to improve the efficiency, economics, reliability, and energy conservation for distribution systems. In this paper, a brief overview on the architecture and functional modules of smart HEMS is presented. Then, the advanced HEMS infrastructures and home appliances in smart houses are thoroughly analyzed and reviewed. Furthermore, the utilization of various building renewable energy resources in HEMS, including solar, wind, biomass and geothermal energies, is surveyed. Lastly, various home appliance scheduling strategies to reduce the residential electricity cost and improve the energy efficiency from power generation utilities are also investigated.]

⇐ *библиографическая ссылка**:

- *Глава 7.6. Подсистема Экосистемы OSTIS, обеспечивающая поддержку жизненного цикла умных домов*
- *§ 7.6.4. Подсистемы умного дома*
- *Пункт 7.6.4.4. Подсистема управления энергопотреблением и энергоэффективностью*

Zotov N.SoftwPfNGICS-2022art

:= *стандартное библиографическое описание**:

[N. Zotov, “Software platform for next-generation intelligent computer systems,” 2022]

⇒ *аннотация**:

[Данная работа является Формальной спецификацией Программной платформы для ostis-систем.]

⇐ *библиографическая ссылка**:

- *Глава 6.3. Программная платформа ostis-систем*
- *Пункт 6.3.2.1. Принципы, лежащие в основе Программной платформы ostis-систем*

Zotov N.SemanToPiN-2022art

:= *стандартное библиографическое описание**:

[N. Zotov, “Semantic theory of programs in next-generation intelligent computer systems,” 2022]

⇒ *аннотация**:

[Несмотря на активное развитие и использование языков программирования, общей теории программ, на основе которой можно было бы проектировать и разрабатывать прикладные системы, на данный момент не существует. Данная работа предлагает Семантическую теорию программ для интеллектуальных компьютерных систем нового поколения.]

⇐ *библиографическая ссылка**:

- *Глава 3.2. Семантическая теория программ для ostis-систем*

Научное издание

Технология комплексной поддержки
жизненного цикла семантически совместимых
интеллектуальных компьютерных систем
нового поколения

Ответственный за выпуск *С. Л. Бочкарева*

В авторской редакции

Подписано в печать 10.04.2023. Формат 60x84 1/8.
Бумага офсетная. Печать цифровая.
Усл. печ. л. 123,8. Уч.-изд. л. 80,59
Тираж 99 экз. Заказ 96.

Издатель и полиграфическое исполнение
УП «Бестпринт». Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий № 1/160 от 27.01.14.
Ул. Филатова, д. 9, к. 1, 220026, г. Минск.