OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

СРЕДСТВА ПОДДЕРЖКИ КОМПОНЕНТНОГО ПРОЕКТИРОВАНИЯ СИСТЕМ, УПРАВЛЯЕМЫХ ЗНАНИЯМИ

Шункевич Д.В., Давыденко И.Т., Корончик Д.Н., Жуков И.И., Паркалов А.В.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

shu.dv@tut.by

ir.davydenko@gmail.com

deniskoronchik@gmail.com

В работе рассматривается подход к проектированию систем, управляемых знаниями, ориентированный на использование совместимых многократно используемых компонентов, что существенно сокращает трудоемкость разработки таких систем. Работа по реализации данного подхода ведется в рамках открытого проекта OSTIS.

Ключевые слова: системы, управляемых знаниями; семантические сети; базы знаний; интеллектуальный решатель задач.

Введение

Как показывает анализ современных информационных технологий, что наряду с достижениями они имеют целый ряд серьезных недостатков, связанных с трудоемкостью их разработки и сопровождения [Голенков 2013]. В частности, к таким недостаткам можно отнести следующие:

- Отсутствует общее унифицированное решение проблемы семантической совместимости компьютерных систем, что порождает высокую трудоемкость создания комплексных интегрированных компьютерных систем. Наиболее остро проблема совместимости компьютерных систем проявляет себя при разработке web-ориентированных систем.
- Высока степень зависимости архитектур компьютерных систем от платформ, на которых они реализованы, что порождает высокую трудоемкость переноса компьютерных систем на новые платформы.
- Современные информационные технологии не ориентированы на широкий круг разработчиков прикладных компьютерных систем.

Для устранения указанных недостатков разрабатывается *Технология OSTIS* [IMS], в основе которой лежит унифицированное представление любых видов знаний при помощи семантических сетей с теоретико-множественной интерпретацией. Одной из основных задач данной технологии является обеспечение возможности быстро создавать системы, управляемые знаниями, с использованием готовых совместимых компонентов. Данная работа рассматривает типологию такого рода компонентов, их особенности и достоинства.

1. Описание Метасистемы IMS

К средствам поддержки компонентного проектирования систем, управляемых знаниями, относится сама *Метасистема IMS*, база знаний которой включает *Библиотеку многократно используемых компонентов OSTIS*, которая будет подробнее рассмотрена далее.

Все системы, построенные по *Технологии OSTIS* имеют общую унифицированную структуру базы знаний. *Метасистема IMS* также строится по *Технологии OSTIS*, и имеет следующую структуру базы знаний:

База знаний IMS
<= *декомпозиция раздела**:
{

- *Документация. Технология OSTIS*
- *Документация. IMS*
- *Контекст Технологии OSTIS и Метасистемы IMS в рамках Глобальной базы знаний*
- *История развития IMS*

- *Документация. Проект IMS*

}

Раздел *Документация. IMS* представляет собой описание собственно самой IMS: ее базы знаний, машины обработки знаний, пользовательского интерфейса.

Раздел *Контекст Технологии OSTIS и Метасистемы IMS в рамках Глобальной базы знаний* содержит знания, которые непосредственно не входят в описание *Технологии OSTIS*, текущего состояния *Метасистемы IMS* и процесса ее развития, но тесно связаны с этими знаниями в рамках *Глобальной базы знаний* (*Глобального смыслового пространства*).

Раздел *История развития IMS* содержит историю изменений базы знаний IMS и других ее частей, то есть, по сути, прошлое системы.

Раздел *Документация. Проект IMS* содержит описание того, каким видится будущее системы, то есть описание планов по ее развитию, текущих задач, исполнителей и т.д.

Раздел *Документация. Технология OSTIS*, в свою очередь, имеет следующую структуру:

Документация. Технология OSTIS
<= *декомпозиция раздела**:

{

- *Обоснование разработки. Технология OSTIS*
- *Раздел. Унифицированные семантические сети и различные варианты их представления*
- *Раздел. Базовая модель обработки унифицированных семантических сетей*
- *Раздел. Унифицированные логико-семантические модели интеллектуальных систем*
- *Раздел. Семейство платформ интерпретации унифицированных логико-семантических моделей интеллектуальных систем*
- *Раздел. Средства проектирования унифицированных логико-семантических моделей интеллектуальных систем*
- *Раздел. Методика проектирования интеллектуальных систем по Технологии OSTIS*
- *Раздел. Библиотека OSTIS*

}

Далее более подробно будет рассмотрено содержание Раздела *Библиотека OSTIS*, как одного из ключевых разделов *Документации OSTIS*, обеспечивающих поддержку компонентного проектирования систем, управляемых знаниями.

2. Библиотека многократно используемых компонентов OSTIS

Библиотека многократно используемых компонентов OSTIS
= Библиотека OSTIS
= многократно используемый компонент OSTIS
= многократно используемый компонент интеллектуальных систем, построенных по Технологии OSTIS

Под ***многократно используемым компонентом OSTIS*** понимается компонент некоторой интеллектуальной системы, который может быть использован в рамках другой интеллектуальной системы. Для этого необходимо выполнение как минимум двух условий:

- есть техническая возможность встроить компонент в другую систему, либо путем физического копирования, переноса и встраивания его в проектируемую систему, либо использования компонента, размещенного в исходной системе наподобие сервиса, то есть без явного копирования и переноса компонента. Трудоемкость встраивания зависит, в том числе, от реализации компонента;
- использование компонента в каких-либо системах, кроме материнской, является целесообразным, то есть компонентом не может быть частное решение, ориентированное на узкий круг задач. Стоит, однако, отметить, что в общем случае практически каждое решение может быть использовано в каких-либо других системах, круг которых определяется степенью общности и предметной зависимостью такого решения.

С формальной точки зрения каждый ***многократно используемый компонент OSTIS*** представляет собой *sc-структуру*, которая содержит все те (и только те) *sc-элементы*, которые необходимы для функционирования компонента в *дочерней sc-системе* и, соответственно, должны быть в нее скопированы при включении компонента в одну из таких систем. Конкретный состав данной *sc-структуры* зависит от типа компонента и уточняется для каждого типа отдельно. По сути, данная *sc-структура* представляет собой эталон или образец, который копируется при включении соответствующего компонента в дочернюю систему.

Каждый ***многократно используемый компонент OSTIS*** может быть атомарным, либо неатомарным, то есть состоять из более простых самодостаточных компонентов.

В зависимости от типа компонента в его составе, т.е. в составе соответствующей *sc-структуры*, могут дополнительно вводиться роли некоторых *sc-элементов*, если это необходимо. Например, в случае *многократно используемого sc-агента,* сам *sc-узел*, обозначающий *sc-агент*, будет являться *ключевым sc-элементом'* в рамках компонента.

В каждый момент времени в текущем состоянии *sc-памяти* каждый многократно используемый компонент может представлен полностью, т.е. в памяти явно присутствуют все *sc-дуги принадлежности*, соединяющие соответствующую компоненту *sc-структуру* и все ее элементы, или представлен неявно, например, при помощи указания *ключевых sc-элементов'* данного компонента или путем задания декомпозиции данного компонента на более частные.

Каждый ***многократно используемый компонент OSTIS*** имеет формальную спецификацию, то есть некоторую *sc-окрестность*, характеризующую данный компонент. На основе формальной спецификации осуществляется поиск подходящего компонента в библиотеке, сравнение его с другими компонентами и т.д.

Данная спецификация включает, как минимум:

- Информацию об авторстве компонента, то есть связь компонента со знаком автора (физического лица, коллектива и т.д.) при помощи отношения *автор**;
- Информацию о типе компонента, посредством указания принадлежности компонента какому-либо классу многократно используемых компонентов;
- Описание назначения компонента, его особенностей;
- Историю изменений компонента по версиям;
- При необходимости сведения об открытости компонента и возможностях его использования в различных системах с точки зрения проприетарности;
- И др.

Не следует путать понятия **версии компонента** и **модификации компонента.** Версии отражают историю изменений компонента (как правило, какие-либо улучшения или устранения ошибок). Модификации представляют собой функционально эквивалентные, но разные варианты реализации одного и того же компонента, которые могут быть синтаксически эквивалентны (то есть быть реализованными при помощи одних и тех же языковых средств). В качестве примера синтаксически не эквивалентной модификации можно привести реализацию одного и того же *sc-агента* на одном и том же языке но с отличиями в алгоритме, в качестве синтаксически эквивалентной модификации – платформенно-зависимую и платформенно-независимую реализацию одного и того же *sc-агента*.

Библиотека многократно используемых компонентов OSTIS постоянно развивается. Появляются абсолютно новые компоненты, у существующих компонентов появляются новые модификации и новые версии.

В общем случае система *IMS* [Корончик 2014], [Давыденко 2014], [Шункевич 2014] как материнская система взаимодействует со всеми своими *дочерними sc-системами* (с системами, построенными по *Технологии OSTIS*), обеспечивая в дочерних системах автоматическое обновление версий *многократно используемых компонентов OSTIS*. Любая дочерняя система, построенная по *Технологии OSTIS*, в том числе, выполняет роль посредника между разработчиком такой системы и системой *IMS*. Разработчик имеет возможность выбрать интересующий его компонент или набор компонентов в одной из библиотек, и включить их в разрабатываемую дочернюю систему. Таким образом, можно говорить о том, что **разработчик** систем, построенных по *Технологии OSTIS*, является **конечным пользователем** системы *IMS*. При обеспечении такого механизма взаимодействия между системами, построенными на основе *Технологии OSTIS*, указанные системы формируют *Глобальную базу знаний*, в пределах которой различные системы могут координироваться и решать более глобальные задачи, нежели это может делать одна отдельно взятая система. В случае намеренной изоляции какой-либо из систем из такого коллектива, в частности, потери связи с системой *IMS*, она теряет возможность получать своевременные обновления используемых компонентов, а также использовать знания, накопленные в других системах для решения стоящих перед ней задач. Речь в данном случае идет не только о физической изоляции рассматриваемой системы, которая может быть легко устранена, а о рассогласовании знаний данной системы и других систем, что не позволит безболезненно интегрировать компоненты из библиотек в такую систему. Таким образом, разработчик каждой системы обязан следить за тем, чтобы его система находилась в постоянном согласовании с глобальным смысловым пространством, что позволит пользователям в полной мере использовать все возможности коллектива 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 *сопутствующий компонент**. 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***, которые целесообразно или

необходимо использовать в дочерней системе вместе. При этом каждая связка направляется от зависящего компонента к зависимому. Каждая такая связка может дополнительно быть снабжена *sc-комментарием* или *sc-пояснением*, отражающим суть указываемой зависимости.

Включение компонента в *дочернюю sc-систему* в самом общем случае состоит из следующих этапов:

- поиск подходящего компонента (или набора компонентов) во множестве библиотек, входящих в состав *IMS*. Для облегчения задачи поиска могут быть реализованы специализированные поисковые агенты. В любом случае, поиск и выделение компонента будет осуществляться на основе спецификации компонента. Данный этап с точки зрения пользователя не зависит от типа компонента и особенностей его реализации. Конкретные действия на следующих этапах сильно зависят от реализации и типа компонента и будут более детально описаны при рассмотрении подклассов ***многократно используемых компонентов OSTIS***;
- выделение компонента (или набора компонентов) в рамках *IMS* в виде, удобном для транспортировки в *дочернюю sc-систему* (при необходимости – создание физической копии компонента);
- транспортировка выделенного компонента в *дочернюю sc-систему*;
- интеграция компонента в *дочернюю sc-систему*. Если в системе уже использовалась более старая версия компонента, то необходимо произвести либо локальное обновление, либо полную замену устаревшей версии компонента. Дальнейший процесс интеграции зависит от типа компонента, например, в случае добавления нового *sc-агента* он должен быть помечен как *активный sc-агент* и т.п.

Для обеспечения возможности встраивания ***многократно используемых компонентов OSTIS*** в дочернюю систему, каждая такая система обязана иметь в своем составе средства, обеспечивающие интеграцию новых компонентов в систему и, при необходимости, удаление устаревших версий этих компонентов (или автоматического локального обновления компонентов до более новой версии).

3. Классификация компонентов

Рассмотрим классы *многократно используемых компонентов OSTIS*.

3.1. Классификация компонентов по характеру их копируемости

Библиотека многократно используемых компонентов OSTIS
<= разбиение:*
{
- *копируемый компонент OSTIS*
- *шаблон типового компонента OSTIS*

}

шаблон типового компонента OSTIS
= образец типового компонента OSTIS
= атомарная логическая формула, описывающая структуру аналогичных (чаще всего изоморфных) компонентов баз знаний sc-систем

В процессе использования ***шаблона типового компонента OSTIS*** при формировании баз знаний проектируемых *sc-систем* вместо *sc-переменных*, входящих в состав компонента, подставляются их значения.

В случае ***копируемых компонентов OSTIS*** в каждой дочерней системе создается конструкция, полностью синонимичная исходному компоненту.

На множестве ***копируемых компонентов OSTIS*** задано отношение *семантическое включение**, т.е. связь более детального описания с менее детальным. Это значит, что степень детализации каких-либо типовых знаний, включаемых в состав баз знаний различных проектируемых sc-систем, может быть различной.

3.2. Классификация компонентов по признаку атомарности

Библиотека многократно используемых компонентов OSTIS
<= разбиение:*
{
- *атомарный многократно используемый компонент OSTIS*
- *неатомарный многократно используемый компонент OSTIS*

}

Под ***атомарным многократно используемым компонентом OSTIS*** подразумевается компонент, который в текущем состоянии библиотеки компонентов рассматривается как неделимый, то есть не содержит в своем составе других компонентов, представленных в какой-либо из библиотек компонентов в рамках *IMS*. 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.

Под ***неатомарным многократно используемым компонентом OSTIS*** подразумевается компонент, который в текущем состоянии библиотеки компонентов содержит в своем составе более простые компоненты, представленные в каких-либо библиотеках компонентов в рамках *IMS*. 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.

3.3. Классификация компонентов по признаку платформенной независимости

Библиотека многократно используемых компонентов OSTIS
<= разбиение:*
{
- *платформенно-зависимый многократно используемый компонент OSTIS*
- *платформенно-независимый многократно используемый компонент OSTIS*

}

Под ***платформенно-зависимым многократно используемым компонентом OSTIS*** понимается компонент, частично или полностью реализованный при помощи каких-либо сторонних с точки зрения *Технологии OSTIS* средств. Основной недостаток платформенно-зависимых компонентов состоит в том, что их интеграция в интеллектуальные системы может сопровождаться дополнительными трудностями, зависящими от конкретных средств реализации компонента. В качестве возможного преимущества ***платформенно-зависимых многократно используемых компонентов OSTIS*** 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*** может поставляться как в виде набора исходных кодов, так и бинарном виде, например в виде скомпилированной библиотеки.

Процесс интеграции ***платформенно-зависимого многократно используемого компонента OSTIS*** в дочернюю систему, разработанную по *Технологии OSTIS*, сильно зависит от технологий реализации данного компонента и в каждом конкретном случае может состоять из различных этапов.

Для того чтобы ***платформенно-зависимый многократно используемый компонент OSTIS*** мог быть успешно встроен в дочернюю систему, необходимо выполнение следующих условий:

- в состав *sc-структуры*, соответствующей компоненту, должны входить *sc-ссылки*, содержащие сведения о местонахождении исходных текстов компонента или уже собранной его версии, то есть ссылки на внешние ресурсы или явно включенные в систему файлы компонента в виде указанных *sc-ссылок*;
- каждый ***платформенно-зависимый многократно используемый компонент OSTIS*** должен иметь соответствующую подробную, корректную и понятную инструкцию по его установке и внедрению в дочернюю систему;

Под ***платформенно-независимым многократно используемым компонентом OSTIS*** понимается компонент, который целиком и полностью представлен в *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***.

Процесс интеграции ***платформенно-зависимого многократно используемого компонента OSTIS*** в дочернюю систему, разработанную по *Технологии OSTIS*, существенно упрощается за счет использования общей унифицированной формальной основы представления и обработки знаний.

В случае ***платформенно-независимого многократно используемого компонента OSTIS*** процесс интеграции конкретизируется до следующих этапов:

- формирование *sc-структуры*, явно содержащей все sc-элементы, входящие в состав компонента, а также все *sc-элементы*, входящие в спецификацию компонента, необходимую для его функционирования в дочерней системе. В случае ***неатомарного многократно используемого компонента OSTIS*** в указанную *sc-структуру* должны быть полностью включены и все более частные компоненты;
- транспортировка компонента в дочернюю систему. В худшем случае может быть осуществлена путем выгрузки всей *sc-структуры* компонента в какой-либо формат, например *SCs-код*, с последующим переносом файла в дочернюю систему разработчиком вручную. В общем случае, дочерняя система по команде разработчика должна

самостоятельно обращаться к родительской и осуществлять загрузку необходимых компонентов;

- интеграция нового компонента в дочернюю sc-систему либо обновление компонента с более старой версии. Если функция обновления не поддерживается, то интерация может проводиться в два этапа – удаление старой версии компонента и добавление в систему более новой версии. В зависимости от типа компонента (*многократно используемый компонент баз знаний* или *многократно используемый компонент машин обработки знаний*) интеграция осуществляется по-разному.

3.4. Классификация компонентов по их месту в архитектуре системы, построенной по Технологии OSTIS

Библиотека многократно используемых компонентов OSTIS
*=> включение**:

- *Библиотека платформ реализации sc-систем*
- *Библиотека многократно используемых компонентов sc-моделей баз знаний*
- *Библиотека многократно используемых компонентов sc-машин*
- *Библиотека многократно используемых компонентов sc-моделей интерфейсов*
- *Библиотека типовых подсистем интеллектуальных систем*

Библиотека платформ реализации sc-моделей
= Библиотека интерпретаторов унифицированных логико-семантических моделей интеллектуальных систем
= интерпретатор унифицированных логико-семантических моделей интеллектуальных систем
= платформа реализации sc-моделей

Под ***платформой реализации sc-моделей*** понимается некоторая реализация платформы интерпретации указанных моделей, которая в общем случае включает в себя:

- хранилище *sc-текстов* (*sc-хранилище*);
- файловую память *sc-машины*;
- средства, обеспечивающие взаимодействие *sc-агентов* над общей памятью;
- базовые средства интерфейса для взаимодействия системы с внешним миром (пользователем или другими системами). Указанные средства включают в себя, как минимум, редактор, транслятор (в sc-память и из нее) и визуализатор для одного из базовых универсальных вариантов представления *SC-кода* (*SCg-код, SCs-код, SCn-код*), средства, позволяющие задавать системе вопросы из некоторого универсального класса (например, запрос семантической окрестности некоторого объекта);
- *абстрактную scp-машину*, то есть интерпретатор *scp-программ*.

При необходимости, в ***платформу реализации sc-моделей*** могут быть заранее на платформенно-зависимом уровне включены какие-либо компоненты машин обработки знаний или баз знаний, например, с целью упрощения создания первой версии дочерней системы на основе *Технологии OSTIS*.

Реализация платформы может осуществляться на основе произвольного набора существующих технологий, включая аппаратную реализацию каких-либо ее частей. С точки зрения *Технологии OSTIS* любая ***платформа реализации sc-моделей*** является ***платформенно-зависимым многократно используемым компонентом.***

Библиотека многократно используемых компонентов sc-моделей баз знаний
= многократно используемый компонент sc-моделей баз знаний

Каждый ***многократно используемый компонент sc-моделей баз знаний*** представляет собой *sc-структуру*, либо явно представленную в текущем состоянии *sc-памяти*, либо не полностью сформированную *sc-структуру*, которая при необходимости может быть полностью сформирована путем объединения своих частей, указанных при помощи какого-либо *отношения декомпозиции*, например *разбиение**, или отношения *включение**.

Интеграция ***многократно используемого компонента sc-моделей баз знаний*** в дочернюю систему сводится к склеиванию ключевых узлов по идентификаторам и устранению возможных дублирований и противоречий, которые могли возникнуть в случае, если разработчик дочерней системы вручную вносил какие-либо изменения в ее базу знаний.

К основным типам компонентов баз знаний, хранящихся в библиотеке компонентов баз знаний, относятся:

- онтологии различных предметных областей, которые могут быть самыми различными по содержанию, однако должны быть семантически совместимыми;
- базовые фрагменты теорий, соответствующие различным уровням знания пользователя, начиная от базового школьного до профессионального;
- различные *sc-окрестности* различных объектов;
- спецификации формальных языков описания различных предметных областей.

Для обеспечения семантической совместимости компонентов баз знаний, которые являются унифицированными семантическими моделями, необходимо

- согласовать семантику всех используемых ключевых узлов;
- согласовать *системные идентификаторы** ключевых узлов, используемых в разных компонентах. После этого интеграция всех компонентов, входящих в состав библиотеки, и в любых комбинациях осуществляется автоматически, без вмешательства разработчика.

Для включения компонента в библиотеку необходимо его специфицировать по следующим критериям:

- предметная область, описание которой содержится в компоненте;
- класс (тип) компонента базы знаний;
- состав компонента;
- количественные характеристики ключевых узлов компонента;
- информация о разработчиках компонента;
- дата создания компонента;
- информация о верификации компонента;
- версия компонента;
- условия распространения компонента базы знаний;
- сопровождающая информация.

К средствам проектирования баз знаний также относятся:

- средства верификации баз знаний, включающие пополняемую библиотеку команд и *sc-агентов* верификации баз знаний;
- средства анализа качества баз знаний, позволяющие определить такие характеристики баз знаний как полнота, связность;
- средства редактирования баз знаний, решающие проблему синхронизации редактирования семантической модели базы знаний и соответствующего фрагмента ее исходного текста несколькими разработчиками;
- средства поддержки коллективной разработки баз знаний.

Библиотека многократно используемых компонентов sc-машин
= многократно используемый компонент sc-машин обработки знаний

Если ***многократно используемый компонент sc-машин*** является *платформенно-зависимым многократно используемым компонентом OSTIS*, то его интеграция производится в соответствии с инструкцией, как и для любого компонента такого рода. В противном случае, процесс интеграции можно конкретизировать в зависимости от подклассов данного типа компонентов.

Библиотека многократно используемых компонентов sc-машин
<= разбиение:*
{
- *Библиотека многократно используемых sc-агентов*
- *Библиотека многократно используемых программ обработки sc-текстов*

}

Библиотека многократно используемых программ обработки sc-текстов
= многократно используемая программа обработки sc-текстов

Под ***многократно используемой программой обработки sc-текстов*** подразумевается компонент, соответствующий программе, записанной на произвольном языке программирования, которая ориентирована именно на обработку знаний, то есть с точки зрения *Технологии OSTIS*, обработку sc-конструкций. 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 и рецепторных *sc-агентов*).

Библиотека многократно используемых программ обработки sc-текстов
*=> включение**:
Библиотека многократно используемых scp-программ

Библиотека многократно используемых sc-агентов
= многократно используемый sc-агент

Под ***многократно используемым sc-агентом*** подразумевается компонент, соответствующий некоторому *абстрактному sc-агенту*, который может быть использован в других системах, возможно, в составе более сложных *неатомарных абстрактных sc-агентов*. Указанный абстрактный sc-агент входит в соответствующую компоненту sc-структуру под атрибутом *ключевой sc-элемент'*. Каждый ***многократно используемый sc-агент*** должен содержать всю информацию, необходимую для функционирования соответствующего *sc-агента* в дочерней системе.

Таким образом, соответствующая ***многократно используемому sc-агенту*** *sc-структура* формируется следующим образом:

1) 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;

2) в случае, если входящий в ***многократно используемый sc-агент*** *абстрактный sc-агент* рассматривается как *неатомарный абстрактный sc-агент*, то ***многократно используемый sc-агент*** будет содержать *sc-узлы*, обозначающие все более частные *абстрактные sc-агенты*, а также все их спецификации согласно п.1. Для каждого включенного в ***многократно используемый sc-агент*** *абстрактного sc-агента* необходимо выполнить п.2 и п.3;

3) для каждого *атомарного абстрактного sc-агента*, знак которого вошел в ***многократно используемый sc-агент*** 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 ***многократно используемый sc-агент*** sc-узел, обозначающий указанную реализацию, а также знаки всех программ, входящие во множество, связанное с указанной реализацией отношением *программа sc-агента**. Выбранная реализация включается в ***многократно используемый sc-агент*** под атрибутом *ключевой sc-элемент'*.

4) для всех *scp-программ*, знаки которых включены в ***многократно используемый sc-агент***, необходимо включить в него полный текст *scp-программы*, то есть все параметры с указанием типа параметра, *scp-операторы*, входящие в ее состав, все соответствующие им операнды с указанием всех выполняемых ими ролей, связи отношения *следующий оператор** для этой операторов этой *scp-программы* и т.д.;

5) в ***многократно используемый sc-агент*** включаются также все связки отношений, указанных в п.1-4, связывающие уже включенные в его состав sc-элементы, а также сами знаки этих отношений (например, *включение**, *программа sc-агента** и т.д.);

После того, как ***многократно используемый sc-агент*** был скопирован в дочернюю систему, необходимо сгенерировать *sc-узел*, обозначающий конкретный *sc-агент*, работающий в данной системе, принадлежащий выбранной реализации *абстрактного sc-агента* и добавить его во множество *активных sc-агентов* при необходимости.

Также каждую *scp-программу*, попавшую в дочернюю систему при копировании ***многократно используемого sc-агента***, необходимо добавить ее во множество *корректных scp-программ* (корректность верифицируется при попадании в библиотеку компонентов в рамках *IMS*).

Библиотека многократно используемых scp-программ

= многократно используемая scp-программа

Под ***многократно используемой scp-программой*** понимается компонент, соответствующий некоторой универсальной *scp-программе*, которая может быть использована в составе сразу нескольких *sc-агентов*.

В ***многократно используемую scp-программу*** включается полный текст *scp-программы*, то есть все параметры с указанием типа параметра, все *scp-операторы*, входящие в ее состав, все соответствующие им операнды с указанием всех выполняемых ими ролей, связи отношения *следующий оператор** для этой операторов этой *scp-программы* и т.д. Сам *sc-узел*, обозначающий *scp-программу*, входит в соответствующий компонент под атрибутом *ключевой sc-элемент'*.

После того, как ***многократно используемая scp-программа*** была скопирована в дочернюю систему, необходимо добавить ее во множество *корректных scp-программ* (корректность верифицируется при попадании в библиотеку компонентов в рамках *IMS*).

Библиотека типовых подсистем интеллектуальных систем

= многократно используемая типовая подсистема интеллектуальных систем

Каждая ***многократно используемая типовая подсистема интеллектуальных систем*** в общем случае представляет собой совокупность *sc-агентов* и фрагментов баз знаний, которые объединены для решения какого-либо класса задач. Таким образом, указанная совокупность не может быть отнесена ни к *многократно используемым компонентам sc-моделей баз знаний*, ни к *многократно используемым компонентам sc-машин*.

Библиотека многократно используемых компонентов sc-моделей интерфейсов

= многократно используемый компонент sc-моделей интерфейсов

Каждый ***многократно используемый компонент sc-моделей интерфейсов*** может быть отнесен также к классу *многократно используемый компонент sc-моделей баз знаний* либо *многократно используемый компонент sc-машин*, однако такие компоненты обладают своей спецификой и поэтому выделяются в отдельный класс. ***Многократно используемые компоненты sc-моделей интерфейсов*** отвечают за взаимодействие интеллектуальной системы с внешней средой, включая также файлы *sc-машины*.

4. Средства автоматизации проектирования систем, управляемых знаниями

Средства автоматизации проектирования систем, управляемых знаниями. можно разделить на средства, входящие в состав *Метасистемы IMS* и средства, которые содержатся в самой *дочерней sc-системе* и представляют собой *многократно используемую типовую подсистему интеллектуальных систем.*

В рамках интеллектуальной *Метасистемы IMS* выделяется подсистема для поддержки проектирования различных компонентов и различных классов интеллектуальных систем.

В рамках рассматриваемой подсистемы необходимо выделить *sc-агент сборки многократно используемого компонента*.

Задачей ***sc-агента сборки многократно используемого компонента*** является формирование *sc-структуры*, соответствующей выбранному *многократно используемому компоненту*, то есть явную генерацию всех *sc-дуг принадлежности* из указанной *sc-структуры* в ее элементы. После завершения работы данного агента рассматриваемая *sc-структура* становится полностью сформированным множеством, а после копирования компонента в дочернюю систему с целью экономии памяти сгенерированные дуги могут снова быть удалены до следующего аналогичного запроса. Данный агент декомпозируется на более частные абстрактные sc-агенты.

sc-агент сборки многократно используемого компонента
<= *декомпозиция абстрактного sc-агента**:
{
- *sc-агент сборки многократно используемого компонента sc-моделей баз знаний*
- *sc-агент сборки многократно используемого sc-агента*
- *sc-агент сборки многократно используемой scp-программы*

}

В свою очередь, в составе каждой дочерней sc-системы выделяется подсистема поддержки проектирования этой системы. В ее состав входит *sc-агент импорта многократно используемого компонента в дочернюю систему*

sc-агент импорта многократно используемого компонента в дочернюю систему
<= *декомпозиция абстрактного sc-агента**:
{
- *sc-агент импорта многократно используемого компонента sc-моделей баз знаний в дочернюю систему*
- *sc-агент импорта многократно используемого sc-агента в дочернюю систему*
- *sc-агент импорта многократно используемой scp-программы в дочернюю систему*

}

Заключение

В работе рассмотрена типология многократно используемых компонентов, используемых в рамках открытой семантической технологии проектирования интеллектуальных систем.

Библиографический список

[Голенков, 2013] Голенков, В.В., Гулякина Н.А. Открытый проект, направленный на создание технологии компонентного проектирования интеллектуальных систем./В.В. Голенков, Н.А. Гулякина// Открытые семантические технологии проектирования интеллектуальных систем (OSTIS-2013): материалы III междунар. науч.-техн.конф. – Минск: БГУИР, 2013 – с.55-78

[IMS] Документация. Технология OSTIS. [электронный ресурс]. – Режим доступа: http://ims.ostis.net/

[Корончик 2014] Корончик, Д.Н. Пользовательский интерфейс интеллектуальной метасистемы поддержки проектирования интеллектуальных систем./ Д.Н. Корончик// Открытые семантические технологии интеллектуальных систем (Ostis-2014): Материалы IV международной науч.-тех. конф. – Минск: БГУИР, 2014 – с.79-82

[Давыденко 2014] Гракова Н.В., Давыденко И.Т., Русецкий К.В. База знаний интеллектуальной метасистемы поддержки проектирования интеллектуальных систем./ Н.В. Граоква, И.Т., Давыденко, К.В. Русецкий// Открытые семантические технологии интеллектуальных систем (Ostis-2014): Материалы IV международной науч.-тех. конф. -- Минск: БГУИР,2014. – с.83-92

[Шункевич 2014] Шункевич, Д.В. Машина обработки знаний интеллектуальной метасистемы поддержки проектирования интеллектуальных систем. /Д.В. Шункевич// Открытые семантические технологии интеллектуальных систем (Ostis-2014): Материалы IV международной науч.-тех. конф -- Минск: БГУИР,2014. – с.93-96

SUPPORT TOOLS KNOWLEDGE-BASED SYSTEMS COMPONENT DESIGN

Shunkevich D.V., Davydenko I.T., Koronchik D.N., Zukov I.I., Parkalov A.V.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

shu.dv@tut.by

ir.davydenko@gmail.com

deniskoronchik@gmail.com

The paper shows a manage knowledge systems design approach, oriented on using compatible reusable components, which help to reduce laboriousness in systems development. The work is carried out within the framework of OSTIS open project.

Key words: manage knowledge systems, semantic networks, knowledge bases, intelligent task resolvent.

Introduction

Analisation of modern information technology demonstrates a number of serios drawbackes assosiated whith laboriousness of their development and maintenance. In particular, there are the following disadvantages:

- There is no common unified problem solution of the computer systems semantic compatibility. It creates difficalty of laboriosness in comprehensive integrated computer systems design, which is particularly acute in development of web-oriented systems.
- The computer system's architecture is platform-dependent and hardly transfered.
- Modern information technology is not focused on a wide range of applied computer systems development engineeres.

To eliminate these shortcomings the OSTIS technology is developed, based on unified knolege representation using the sematic networks whith set-theoretic interpretation. One of the main goal of this technology is to insure knowledge based systems production applying the ready-made compossible components.

Main Part

Metasystem IMS is a supporting tools for manage knowledge systems component design, which knowledge base includes *Library of reusable components OSTIS*, an it will be further showed.

All systems based on *OSTIS Technology*, have the general structure of knowledge base. *Metasystem IMS* is also based on *OSTIS Technology* have the following structure of knowledge base:

Knowledge base IMS
<= *section decomposition**:
{
- *OSTIS Technology documentation*
- *IMS documentation*
- *OSTIS Technology and IMS Metasystem context inside of Global knowledge base.*
- *Historyof IMS evolution*
- *Documentation. IMS Progect*

}

Section *IMS Documentation* is a description of IMS itself: it's knowledge base, knowledge elaboration machine and human-system interface.

Section *OSTIS Technology and IMS Metasystem context inside of Global knowledge base* contain knowledge, which are not included in OSTIS Technology description, but are closely related with knowledge of *Global knowledge base.*

Section *History of IMS evolution* contain the history of IMS knowledge base and it's parts changes.

Section *Documentation. IMS Project* contains the description of its future state.

Conclusion

In the paper described the typology of reusable components, which are used in open semantic technology of intelligent systems design.