OSTIS-2016
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ПРИНЦИПЫ ПЛАТФОРМЕННОЙ НЕЗАВИСИМОСТИ И ПЛАТФОРМЕННОЙ РЕАЛИЗАЦИИ OSTIS

Ивашенко В.П., Татур М.М.

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

ivp@tut.by
tatur@i-proc.com

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.
Ключевые слова: семантические сети, интеллектуальные системы, платформенная независимость, абстрактные машины.

Введение

Для облегчения разработки, развития технологий разработки интеллектуальных систем и самих интеллектуальных систем важным свойством является возможность многократного использования компонентов интеллектуальных систем [Голенков и др, 2015]. Многократность использования разработанных компонентов означает, что сокращаются затраты на разработку нового компонента-аналога в случае необходимости построения более совершенной интеллектуальной системы. Кроме того, наличие в технологии разработки средств интеграции знаний [Ивашенко, 2015], позволяющих добавлять новые компоненты без остановки работы интеллектуальной системы, сводит процесс разработки новой более совершенной интеллектуальной системы к процессу обучения существующей. Таким образом, при наличии механизмов интеграции знаний в процессе работы системы и возможности многократного использования компонентов технология позволяет создавать совершенствуемые и обучаемые интеллектуальные системы. Необходимым свойством таких систем является сохранение основных потребительских свойств, включая производительность и потребление ресурсов.

Для реализации платформы необходима некоторая машина, поэтому при спецификации платформы будем сопоставлять ей некоторую абстрактную машину [Кузьмицкий В.М., 1998].

В общих чертах семейство реализуемых абстрактных машин можно рассмотреть в пространстве следующих признаков (рисунок 1): аппаратная или программная реализация, уровень языка платформы (способность представлять на нём данные сложной структуры и ассоциации), универсальность абстрактной машины (способность её моделировать другие машины) [Kazuhiro, 1984], [Hewitt, 2009].

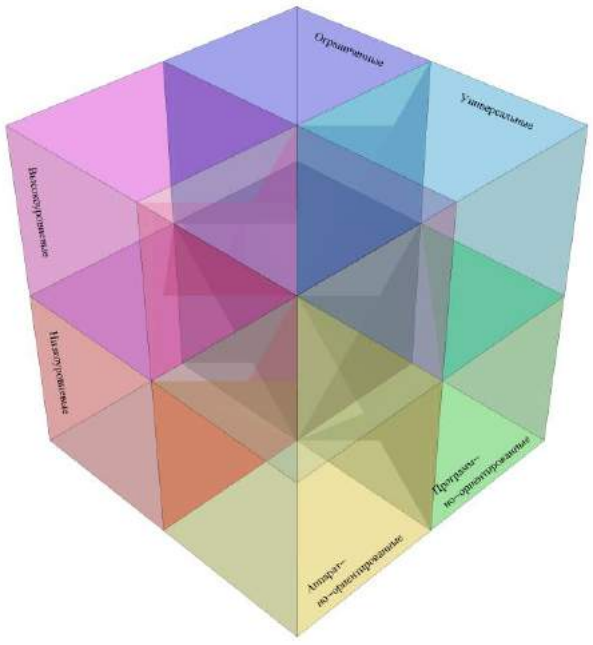

Рисунок 1 – Реализуемые платформы (абстрактные машины)

Далее будут рассмотрены более детально устройство и виды абстрактных машин, подходы к спецификации платформ, отношения между ними и принципы реализации интеллектуальных систем, имеющих платформенно-независимые компоненты.

1. Абстрактные машины

1.1. Память абстрактных машин

Каждая абстрактная машина соответствует формальной модели обработки информации [Голенков и др, 2001], задаваемой языком, начальной информационной конструкцией и множеством операций. Хранимые в памяти абстрактной машины информационные конструкции являются текстами некоторого языка (языка представления). Рассмотрим абстрактные машины, хранящие тексты языков, являющихся подмножествами A^{**}, замыкания замыканий алфавита A.

В зависимости от свойств языка абстрактные машины разделяются на классы:

- абстрактные машины, способные хранить тексты бесконечных языков, такие машины обладают неограниченной памятью, к таким машинам относятся стековые автоматы, машина Тьюринга, универсальные абстрактные машины;
- абстрактные машины, способные хранить только тексты конечного языка, к таким машинам относятся конечные автоматы;
- абстрактные машины, хранящие тексты симметричного языка A^{size}, где $size$ – количество ячеек памяти, память таких машин может рассматриваться как линейно организованная относительно A;
- абстрактные машины с распределённой памятью, язык этих машин является ассоциацией (подмножеством декартова произведения) языков абстрактных машин каждого из её узлов,

1.2. Операции абстрактных машин

Операции (языки исполнения) абстрактной машины являются бинарными отношениями на множестве текстов языка абстрактной машины, ассоциациями языка представления абстрактной машины самого с собой.

В зависимости от свойств можно различать следующие виды операций:

- транзитивные операции,
- симметричные операции,
- детерминированные операции, которые являются функциями, частным случаем детерминированных операций являются алгебраические операции,
- недетерминированные операции.

К свойствам множества операций абстрактной машины можно отнести обратимость (наличие обратных операций), сходимость (наличие аттракторов), взаимную исключительность (отсутствий операций с общими элементами) и независимость, когда ни одна операция не может быть представлена как объединение других операций или их композиций.

Отдельно выделяемым видом операций абстрактной машины являются операции ввода (сенсорные операции), которые позволяют изменить состояние, переходить к новому состоянию памяти абстрактной машины за счёт ввода данных из внешней среды.

2. Платформы

2.1. Виды платформ

Для описания платформ воспользуемся формальными моделями онтологий [Гаврилова и др., 2000], которые задаются тройкой конечных множеств

$$\langle G, R, O \rangle, \tag{1}$$

где G – конечное множество обозначений элементов объекта описания, R – строка отношений (языков структуры) на множестве G, R – строка функций (языков интерпретации) на множестве G.

Каждое состояние памяти абстрактной машины, соответствующей платформе, описывается формальной моделью его онтологии, поэтому для исследования операций абстрактной машины можно применить модель спецификации знаний [Ивашенко, 2015].

Модель спецификации знаний задаётся множеством формальных моделей онтологий Z, множеством морфизмов, соответствий между ними, и множеством отношений между парами формальных моделей и множеством соответствий и морфизмов.

$$\begin{gathered} \left\langle Z \cup 2^{\{\omega(z)|z\in Z\}^2}, 2^{\Theta(\langle Z,Z\rangle)} \right\rangle; \\ \Theta\left(\langle X,Y\rangle\right) = \bigcup_{\langle x,y\rangle \in X\times Y} \left(\{x\}\times\{y\}\right) \times 2^{\omega(x)\times\omega(y)}, \end{gathered} \tag{2}$$

где $2^{\Theta(\langle Z,Z\rangle)}$ – множество отношений модели спецификации; ω – функция элементов онтологической модели

$$\begin{gathered} \omega(z_i) = G_i \cup \{r | r = R_{ij}\} \cup \{o | o = O_{ij}\} \\ \cup \{k | k \in R_{ij}\} \cup \{p | p \in O_{ij}\} \cup \{a | \langle a, v\rangle \in O_{ij}\}. \end{gathered} \tag{3}$$

Таким образом, каждой операции абстрактной машины платформы взаимно-однозначно соответствует отношение модели спецификации знаний, которое каждой паре формальных моделей онтологий (состояний абстрактной машины платформы) сопоставляет множество соответствий и морфизмов между такими онтологиями.

Платформа от абстрактной машины отличается тем, что для неё определены физические характеристики, включающие затраты памяти,

временные затраты, энергоинформационные затраты или потери и другие. Функции, вычисляющие эти затраты представимы в модели оценки качества, которая использует модель спецификации знаний.

Для вычисления мер в соответствии с выделенным языком L и записываемыми в его текстах требованиями используются функции вида

$$\left\{\left\langle L, 2^{\Phi^{\Theta(\langle\zeta(L),\zeta(L)\rangle)}}\right\rangle \middle| L \subseteq A^{*^*}\right\}, \quad (4)$$

где $\Theta(\langle Z, Z\rangle)$ – объединение отношений модели спецификации на Z; ζ – отображение множества текстов языка спецификации на множество их онтологических моделей; Ф – упорядоченное множество оценок.

Таким образом, если определена некоторая детерминированная операция модели $\varphi \in M^M$ или недетерминированная $\psi \in 2^{M\times M}$, то существуют предикаты $\pi_{\Phi\varphi} \in 2^{\Phi^{\Theta(\langle\zeta(M),\zeta(M)\rangle)}}$ и $\pi_{\Phi\psi} \in 2^{\Phi^{\Theta(\langle\zeta(M),\zeta(M)\rangle)}}$ такие, что при $\Phi = \{0,1\}$

$$\pi_{\Phi\psi}\left(\langle\langle x, y\rangle, w\rangle\right) = \begin{cases} 1 \middle| \exists u \exists v \exists x \exists y P\left(\langle u, v, x, y\rangle\right) \\ 0 \middle| \neg\exists u \exists v \exists x \exists y P\left(\langle u, v, x, y\rangle\right) \end{cases}$$

$$P\left(\langle u, v, x, y\rangle\right) = \left(\langle x, y\rangle \in \left(\zeta\left(\{u\}\right)\times\zeta\left(\{v\}\right)\right)\right) \wedge \\ \wedge\left(w \subseteq \omega(x)\times\omega(y)\right)\wedge\left(\langle u, v\rangle \in \psi\right) \quad (5)$$

Затраты или потери в этом случае могут быть выражены:

$$\langle x, y\rangle \in \left(\zeta\left(\{u\}\right)\times\zeta\left(\{v\}\right)\right) \\ \mu_{\Phi'\psi}\left(\langle\langle x, y\rangle, w\rangle\right) = \pi_{\Phi\psi}\left(\langle\langle x, y\rangle, w\rangle\right) * \mu_{\Phi'\psi}\left(\langle u, v\rangle\right), \quad (6)$$

где $\mu_{\Phi'\psi}$ – функция затрат на множестве Ф′ при переходе из состояния u в v.

Для сравнения операций необходимо перейти от формальных моделей онтологий, описывающих состояния, к формальным моделям онтологий, описывающим операции. Когда мощность операций бесконечна, такой переход не возможен без привязки формальной модели онтологии операции к формальным моделям онтологий состояний отношениями модели спецификации знаний. Поэтому для привязки используются три типа отношений модели спецификации знаний: отношения объединения пар формальных моделей онтологий состояний, соответствующих парам состояний операции, отношения реификации соответствия формальных моделей онтологий пар состояний операции и отношения проецирования формальных моделей онтологий реифицированного соответствия пар состояний операции на формальную модель онтологии операции. Тогда в модели оценки качества становится возможным дать оценку каждой паре формальных моделей онтологий операций.

Вид платформы может зависеть как от свойств её абстрактной машины, так и от её характеристик. Можно выделить два виды платформ: масштабируемые, немасштабируемые платформы.

Среди масштабируемых платформ можно различать масштабируемые по множествам затрат или потерь платформы. Среди масштабируемых по множествам затрат или потерь можно различать платформы, масштабируемые по множеству затрат или потерь по входу (входному состоянию), масштабируемые по множеству затрат или потерь по выходу (выходному состоянию), полностью масштабируемые по множеству затрат или потерь. Среди платформ, масштабируемых по множеству затрат или потерь по входу, выходу либо полностью можно выделять экспоненциально, полиномиально, квадратично, линейно, полилогарфмически, логарифмически масштабируемые (по множеству затрат или потерь по входу, выходу либо полностью (или немасштабируемые)) и другие (в соответствии с классом сложности или некоторым условием).

2.2. Модель платформ

Модель платформ рассматривает отношения между платформами. Одним из важных отношений является реализуемость одних платформ на базе других [Кузьмицкий В.М., 1998]. Реализуемость является транзитивным замыканием объединения отношений эмуляции, его обращения и отношения непосредственной реализации.

Непосредственная реализация заключается в переассоциации языка представления (абстрактной машины, виртуальной машины) платформы в язык абстрактной машины реализуемой платформы. Каждая операция (язык исполнения) реализуемой платформы должна быть сводима, (зависима, алгоритмически вычислима) к набору операций реализующей платформы. Переассоциация языков – отношение, являющееся объединением отношения ассоциации языков с композицией отношений объединения диссоциаций языков и ассоциации языков.

Отношение эмуляции платформ сводится к изоморфному вложению языка представления абстрактной машины эмулируемой платформы в язык эмулирующей платформы с точностью до неразличимости структуры элементов алфавита и изоморфному вложению всех операций (языков исполнения) эмулирующей машины в эмулируемую. При этом вложение языков исполнения осуществляется, таким образом, что для всех сопоставленных вложением аргументов их образы соответствующих операций взаимно-однозначно сопоставлены этим изоморфным вложением.

Редукция – отношение между множеством платформ и платформой, где множество языков реализующих платформ переассоциируемо в язык представления реализуемой платформы. Каждая операция реализуемой платформы должна быть алгоритмически сводима (зависима) к операциям

реализующей платформы. Платформы, не имеющие непосредственно реализующей платформы, могут рассматриваться как аппаратно-реализуемые [Ивашенко и др., 2015].

Характеристики всех непосредственно реализуемых платформ ограничиваются характеристиками этой платформы. Так, значения характеристик затрат (времени, памяти и т.п.) или потерь не возрастают для каждых двух пар соответствующих состояний реализуемой и реализующей платформы. При редукции характеристики затрат (памяти или времени) редуцируемой платформы меньше максимальных затрат (памяти или времени) одной из редуцирующих платформ. В случае параллельной редукции временные затраты реализуемой платформы могут быть меньше суммы временных затрат редуцирующих платформ. Поэтому для обеспечения требуемых характеристик реализуемой платформы каждая платформа каждого уровня (непосредственной) реализации должна гарантировать некоторые требования интересующих характеристик. Таким образом, каждый уровень реализации должен предоставлять, давать гарантии для обеспечения требований к реализуемой платформе. Как правило хорошими гарантиями затрат являются затраты менее квадратичных.

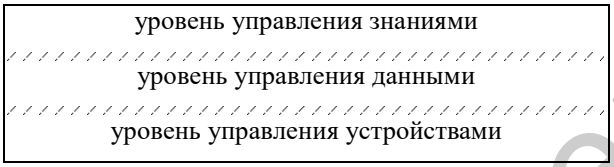

Рисунок 2 – Уровни управления (при включении языков представления друг в друга границы между уровнями могут быть неразделимы, что проиллюстрировано волнистой линией)

Относительно пары выделяемых языка представления данных и языка представления знаний выделяются уровни уровень управления устройствами, уровень управления данными, уровень управления знаниями. В уровень управления устройствами входят платформы, реализующие платформу языка представления данных, в уровень управления данными входят платформы, реализуемые на платформе с языком представления данных и реализующие платформу языка представления знаний, в уровень управления знаниями входят все платформы, реализуемые с использованием языка представления знаний (рисунок 2).

3. Платформенная независимость

3.1. Виды платформенной независимости

Платформа обладает платформенной независимостью, если существует реализующая её платформа, не реализующая её непосредственно и не реализующая её через другие непосредственно реализуемые платформы. Соответственно в зависимости от числа таких платформ платформа может обладать большей или меньшей степенью платформенной независимости.

При платформенной независимости платформ с конечными языками представления обычно важным является различие простых характеристик (затрат времени или памяти при переходе между состояниями), выражаемое в виде отношения или разности соответствующих величин. Для платформ с бесконечными языками представления, как правило, важно сохранение (вида) масштабируемости платформы. В этом случае платформенная независимость различается по сохраняемому виду масштабируемости платформы.

3.2. Принципы построения платформенно независимых систем, управляемых знаниями

Структура системы, управляемой знаниями, в общем виде может быть иллюстрирована схемой (рисунок 3), включающей базу знаний, решатель и (пользовательский) интерфейс. Рассматриваемые системы ориентированы на обработку знаний представленных в SC-коде [Голенков и др, 2001], [Ивашенко, 2015].

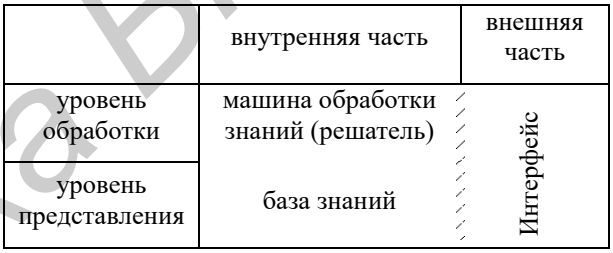

Рисунок 3 – Общая схема системы, управляемой знаниями

Более детализированная схема [Голенков и др, 2013] включает несколько уровней, основанных на агентно-ориентированном подходе (рисунок 4).

В соответствии с методикой компонентного проектирования систем, управляемых знаниями, выделяются следующие основные этапы [Шункевич и др., 2015b]:

- выбор и установка платформы реализации производной (дочерней) sc-системы, включая выбор и установку sc-хранилища и установку scp-машины;
- установка ядра базы знаний производной sc-системы;
- установка Ядра sc-машин, то есть набора базовых многократно используемых компонентов sc-машин, необходимых для работы даже первого прототипа sc-системы;
- установка Ядра sc-моделей интерфейсов, то есть набора базовых многократно используемых компонентов sc-моделей интерфейсов, необходимых для работы даже первого прототипа sc-системы;
- установка ядра подсистемы поддержки проектирования дочерней системы в составе проектируемой производной sc-системы.

К неосновным этапам относятся: расширение базы знаний производной sc-системы; расширение

машины обработки знаний производной sc-системы; расширение возможностей пользовательского интерфейса производной sc-системы.

Каждый агент (sc-агент [Голенков и др, 2015]) в реализации интеллектуальной системы может рассматриваться как некоторая платформа, поэтому к анализу его свойств применимы соответствующие подходы.К платформенно-независимым sc-агентам относятся sc-агенты, реализуемые на некоторой базовой sc-машине (scp-машине).

<table>
<tr><td colspan="2">Коллектив интегрированных интеллектуальных систем</td></tr>
<tr><td>Обрабатываемая база знаний интеллектуальной системы
Система предметных областей, онтологий и формальных теорий
Система семантически целостных фрагментов базы знаний
Система взаимосвязанных sc-элементов</td><td></td></tr>
<tr><td colspan="2">Коллектив интегрированных решателей задач</td></tr>
<tr><td colspan="2">Решатель задач</td></tr>
<tr><td>Знания решателя задач используемые для управления процессом обработки знаний на основании используемой модели решения задач</td><td>Коллектив неатомарных агентов над обрабатываемой базой знаний
Коллектив атомарных агентов над обрабатываемой базой</td></tr>
<tr><td>Хранимые scp-программы, описывающие поведение атомарных агентов над базой знаний
Библиотека scp-программ, описывающих базовые преобразования обрабатываемых знаний</td><td>Неатомарный агент интерпретации хранимых scp-программ
Коллектив атомарных агентов интерпретации хранимых scp-программ</td></tr>
<tr><td>Техническая реализация sc-памяти</td><td>Техническая реализация атомарных агентов интерпретации хранимых scp-программ</td></tr>
</table>

Рисунок 4 – Уровни детализации интеллектуальной системы

Следует отметить, что в зависимости от реализации sc-хранилища [Шункевич и др., 2015a] и мощности соответствующего sc-языка [Ивашенко, 2015], возможны разные (частичные) реализации (версии) scp-машин, не каждая из которых реализуема на другой. Поэтому, говоря о платформенно-независимых компонентах (sc-агентах), следует уточнять о платформенной независимости относительно какой реализации (версии) scp-машины идёт речь.

При платформенной реализации на её разных уровнях решаются разные задачи (рисунок 5).

В случае использования линейно адресуемой памяти к операциям над данными могут быть отнесены: операции над непрерывными участками памяти, содержащими данные (массивами, непрерывными функциями), операции над несвязными (прерывными) участками памяти с данными (списками, деревьями, функциями).

<table>
<tr><td>3</td><td>уровень управления знаниями</td><td>операции обработки знаний</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">уровень управления данными</td><td rowspan="2">операции над данными</td></tr>
<tr><td>перевыделение участков памяти</td></tr>
<tr><td>1</td><td>уровень управления памятью (устройствами)</td><td>выделение и высвобождение участков памяти</td></tr>
</table>

Рисунок 5 – Задачи, решаемые на разных уровнях управления

Следует отметить, что важными принципами [] реализации операций обработки знаний (рисунок 5) для интеллектуальных систем, к которым относятся системы, управляемые знаниями относятся: принцип учёта НЕ-факторов, принцип семантического протоколирования процесса работы системы, принцип распределённой обработки потока данных (знаний).

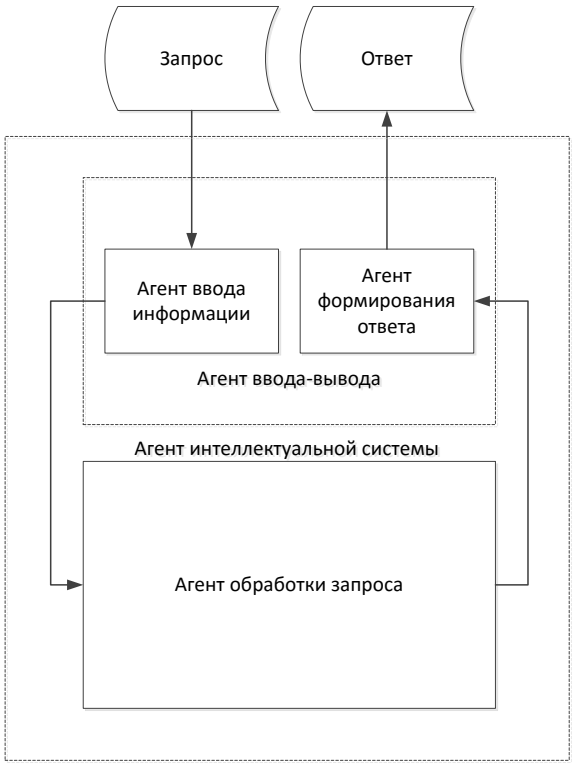

Рисунок 6 – Схема взаимодействия агентов в системе, управляемой знаниями

С точки зрения агентно-ориетинрованного подхода (в соответствии с рисунком 3) в упрощённом виде схема системы включает sc-агенты (рисунок 6): агент ввода-вывода (декомпозированный на агент ввода информации и агент формирования ответа (вывода)) и агент обработки запроса (решатель).

Агент ввода обеспечивает разбор входных данных (запроса) в соответствии с известной агенту грамматикой и формирует условия (команды) запуска агентов обработки запроса. Агент ввода (как и агент вывода) может редуцироваться к другим агентам ввода, если язык входных (выходных) данных является ассоциацией языков.

Агент вывода обеспечивает синтез и отправку ответа пользователя в соответствии с правилами синтеза текстов выходного языка.

Агент ввода-вывода может содержать другие редуцирующие агенты ввода-вывода помимо изображённых агентов ввода и вывода.

Агент обработки запроса (решатель) и его база знаний обеспечивает решение прикладной задачи в соответствии с некоторым планом решения этой задачи являющегося результатом логического вывода. Агент обработки запроса работает на уровне управления знаниями, который так же может быть разбит на уровни: уровень управления систематикой (первичных) элементов предметной области, уровень управления отношениями предметной области, уровень управления утверждениями о предметной области, уровень управления задачами предметной области и т.д. Для каждого уровня строится своя онтология (база знаний), обеспечивающая решение задачи управления на данном уровне.

Агент обработки запроса (решатель) в общем случае может содержать не только агент или агенты, ориентированные на решение прикладной задачи, но и агенты обеспечивающие реализацию основных принципов: агент интеграции знаний, учитывающий НЕ-факторы; агент семантического протоколирования процесса обработки знаний (работы системы); агент сборки мусора на основе анализа семантического протокола; агенты управления потоком знаний.

Примером системы, построенной с учётом перечисленных принципов, является система решения задач на графах, решающая задачи определения вида графа. Для этого решатель использует методы и алгоритмы канонической разметки графа, поиска мостов, точек сочленения, маршрутов и циклов в графе.

Заключение

В работе рассмотрены виды платформы, отношения между ними, понятие платформенной независимости и принципы реализации интеллектуальных систем в соответствии с технологией OSTIS.

Библиографический список

[Гаврилова и др., 2000] Гаврилова, Т.А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2000.

[Голенков и др, 2001] Голенков, В.В. Представление и обработка знаний в графодинамических ассоциативных машинах /В. В. Голенков[и др.] – Мн. : БГУИР, 2001.

[Голенков и др, 2013] Голенков, В.В., Гулякина Н.А., Открытый проект, направленный на создание технологии компонентного проектирования интеллектуальных систем Материалы. Международной научн.-техн. Конференции OSTIS,2013:Минск, Республика Беларусь, БГУИР 21-23 февраля 2013.

[Голенков и др, 2015] Голенков, В.В., Гулякина Н.А., семантическая технология компонентного проектирования систем, управляемых знаниями Материалы Международной научн.-техн. Конференции OSTIS,2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015.

[Ивашенко, 2015] Ивашенко, В.П. Модели и алгоритмы интеграции знаний на основе однородных семантических сетей. Материалы Международной научн.-техн. Конференции OSTIS,2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015.

[Ивашенко и др., 2015] Ивашенко, В.П. Представление семантических сетей и алгоритмы их организации и семантической обработки на вычислительных системах с массовым параллелизмом. Материалы Международной научн.-техн. Конференции OSTIS,2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015 / Ивашенко, В.П., Вереник Н.Л., Гирель А.И., Сейткулов Е.Н., Татур М.М. // Мн.: БГУИР, 2015.

[Кузьмицкий В.М., 1998] Кузьмицкий, В.М. Принципы построения графодинамического параллельного ассоциативного компьютера, ориентированного на переработку сложноструктурированных знаний / В.М. Кузьмицкий // Интеллектуальные системы : сб. науч. тр. / НАН Беларуси, Ин-т техн. кибернетики ; науч. ред. А.М. Крот. – Минск, 1998. – Вып. 1. – С. 156–166.

[Шункевич и др., 2015a] Шункевич, Д.В. Средства поддержки компонентного проектирования систем, управляемых знаниями. Материалы Международной научн.-техн. Конференции OSTIS,2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015 / Шункевич Д.В., Давыденко И.Т., Корончик Д.Н., Жуков И.И., Паркалов А.В. // Мн.: БГУИР, 2015.

[Шункевич и др., 2015b] Шункевич, Д.В. Методика компонентного проектирования систем, управляемых знаниями. Материалы Международной научн.-техн. Конференции OSTIS,2015:Минск, Республика Беларусь, БГУИР 19-21 февраля 2015 / Шункевич Д.В., Давыденко И.Т., Корончик Д.Н., Губаревич А.В., Борискин А.С. // Мн.: БГУИР, 2015.

[Hewitt, 2009] Carl Hewitt Middle History of Logic Programming: Resolution, Planner, Prolog and the Japanese Fifth Generation Project ArXiv 2009. Mode of access: http://arxiv.org/abs/0904.3036. Date of access: 30.11.2015.

[Kazuhiro, 1984] Kazuhiro Fuchi, Revisiting Original Philosophy of Fifth Generation Computer Systems Project, FGCS 1984, pp. 1-2.

FOUNDATIONS OF PLATFORM INDEPENDENCY AND OSTIS PLATFORM IMPLEMENTATION

Ivashenko V.P., Tatur M.M.

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

ivp@tut.by

tatur@i-proc.com

The paper deals with an approach to the platform specifications , platform comparison, principles of their implementation and types of platform independence. The framework of platform independent components of intelligent systems are described. This framework uses homogenious semantic network with set-theoretic interpretation as the language of knowledge representation.