OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514

ОБЛАЧНЫЕ СЕМАНТИЧЕСКИЕ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СЕРВИСОВ

Грибова В.В., Клещев А.С.

Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук, г. Владивосток, Россия

gribova@iacp.dvo.ru,
kleschev@iacp.dvo.ru

В работе описана технология разработки и сопровождения интеллектуальных систем как интеллектуальных сервисов. В основе предлагаемой технологии разработки интеллектуальных систем лежит принцип четкого разделения между декларативными знаниями (знаниями предметной области) и процедурными (о методе решении задачи). Описаны принципы формирования каждого компонента интеллектуальной системы - базы знаний и данных, решателя задач и пользовательского интерфейса.

Ключевые слова: интеллектуальные системы; облачные технологии; семантические технологии; онтологии; базы знаний.

Введение

Разработка интеллектуальных систем и систем обработки сложно-структурированной информации, является трудоемким процессом, но еще более сложным и трудоемким является сопровождение таких систем. Поэтому исследование методов, средств, технологии разработки и сопровождения остается одной из актуальных задач.

В настоящее время можно выделить две ключевые технологии разработки таких систем: первая технология сводится к тому, что разработчику предлагается готовый решатель, и, в соответствии с представлением информации, на которое он рассчитан, он должен сформировать базу знаний, при этом часть знаний о решении задачи встроить в нее. Эту технологию поддерживают оболочки интеллектуальных систем. В качестве способа представления знаний используются базы правил. Первый опыт их применения показал, что базы знаний, основанные на правилах, исключительно сложно не только разрабатывать, но, прежде всего, сопровождать [Bachant et al., 1984], поскольку эксперты не понимают такое представление. Разработанные подходы, стратегии, методы и процедуры работы с экспертами по извлечению знаний позволили упростить их создание, однако модификация, расширение знаний в процессе эксплуатации системы, осталась трудоемким процессом, для чего такие системы не предназначены, что было отмечено еще в [Compton, 1988], поскольку метод решения задачи частично содержится в базе знаний.

Вторая технология исходит от обратного принципа – при построении интеллектуальных систем наиболее сложным для создания и сопровождения компонентом является база знаний и любая другая сложно-структурированная информация. Поэтому она должна разрабатываться так, чтобы быть повторно-используемой в различных интеллектуальных системах. Этот подход предполагает сначала формирование метастуктуры или онтологии, по которой может быть создан класс различных баз знаний либо другой сложно-структурированной информации, ей соответствующий. Решатель задачи основан на онтологии обрабатываемой информации и, таким образом, может быть повторно использован для различных интеллектуальных систем, имеющих ту же онтологию обрабатываемой информации. Основные преимущества такой технологии – процедурные знания отделены от декларативных, решатель задач является повторно-используемым для класса задач с общей онтологией обрабатываемой информации. Данная технология применена в [Gennari et al., 2003, Клещев и др., 2006а]. В основе технологии Протеже лежит объектно-ориентированный подход к формированию онтологий. В онтологии определяется иерархия классов для представления основных ее объектов, множество слотов для описания их свойств и отношений между ними, а также множество экземпляров классов. Решатель задач состоит из трех основных компонентов:

описания алгоритма, включающее все шаги решения задачи, описания так называемой онтологии метода, которое включает спецификации форматов входных и выходных данных, а также непосредственно программного кода, реализующего метод. Пользовательский интерфейс входит в метод решения задачи. В основе метода решения лежит утверждение, выдвинутое еще Чандрасекараном о том, что в интеллектуальных системах можно выделить классы задач, которые они решают, и для каждого класса задач разработать метод ее решения, который будет повторно использоваться в различных системах (методы могут быть проблемно-зависимыми).

Однако в данном подходе отсутствует четкое разделение между двумя системами понятий. Такой подход, с одной стороны, «позволяет быстро и относительно просто сконструировать небольшую предметную онтологию», однако отсутствует четкое разделение между базой знаний и онтологией [Пивоварова и др., 2007]. Возможность разработки общих для разных онтологий методов решения интеллектуальных задач также не представляется универсальным механизмом, несмотря на отдельные удачные примеры, представленные в работах.

В работах [Клещев и др., 2006b] также используется идеология полного разделения процедурных и декларативных знаний, принцип формирования базы знаний по онтологии. Существенным различием является четкое разделение между уровнями и, таким образом, разделение труда между инженерами по знаниям (формируют онтологию через интерфейс, ориентированный на метаязык) и экспертами (формируют базу знаний через сгенерированный по онтологии интерфейс).

Целью данной работы является описание дальнейшего развития технологии – методов формирования каждого компонента интеллектуальной системы - базы знаний и данных, решателя задач и пользовательского интерфейса.

1. Информационные ресурсы

Интеллектуальные системы имеют специфический архитектурный компонент – базу знаний, поэтому основной частью технологического процесса создания и сопровождения интеллектуальных систем является разработка и модификация базы знаний. При этом технология должна быть такой, чтобы эксперты могли формировать и сопровождать базы знаний любой степени сложности самостоятельно, без посредников. Задача разработки такой технологии может быть разбита на две подзадачи:

− отделение терминологии инженера по знаниям от терминологии эксперта, создание методов формирования и сопровождения баз знаний, ориентированных на экспертов, в понятной им терминологии;

− автоматизация разработки редакторов баз знаний, поскольку редакторы баз знаний являются сложными программными средствами; очевидно, что практически невозможно создавать редактор под формирование каждой базы знаний, поэтому необходимы методы и средства автоматизации разработки редакторов баз знаний, ориентированных на экспертов.

Рассматривается двухуровневая модель информационных ресурсов (под информационными ресурсами в данной работе понимаются базы знаний, базы данных и вообще любая сложным образом организованная информация), предложенная в работе [Клещев и др., 2006b], в соответствии с которой информация порождается или модифицируется под управлением метаинформации. Таким образом, в соответствии с данной моделью любая информация связана с метаинформацией, по которой она порождается или модифицируется. Метаинформация информационного ресурса является представлением графовой грамматики (грамматики, порождающей размеченные графы) языка представления информации.

Метаинформация (МИ) представляет собой пару МИ=(GM, σM), где GM – граф понятий для описания информации (баз знаний и данных), возможно, содержащий циклы и петли, а σM - разметка этого графа. Граф GM есть тройка <Vertexes, Arcs, RootVertex>, где Vertexes - множество вершин графа, Arcs – множество дуг графа, RootVertex - начальная вершина графа. Множество вершин графа $Vertexes = \{Vertexi\}_{i=1}^{vertexescaunt}$ состоит из двух непересекающихся подмножеств – терминальных вершин Terminal_Vertexes и нетерминальных вершин Neterminal_Vertexes. Начальной вершиной графа RootVertex является нетерминальная вершина, RootVertex∈Neterminal_Vertexes.

Множество дуг графа Arcs есть множество пар $\{Arci\}_{i=0}^{arcscount}$, Arc_i=<$VertexFrom_i$, $VertexTo_i$>, где $VertexFrom_i$, $VertexTo_i$ – вершины из которых выходит и в которые входит дуга Arc_i, соответственно, $VertexFrom_i$∈ Neterminal_Vertexes, $VertexTo_i$∈ Vertexes.

Средства разметки σM включают в себя разметку вершин σV и разметку дуг σA. Эта разметка позволяет при описании графа метаинформации задать ограничения на структуру и содержание графа информации.

Разметка вершин σV задается следующим образом: начальная вершина отображается в имя графа метаинформации, т.е. σV(RootVertex) = Имя Графа Метаинформации; нетерминальные вершины графа отображаются в пару: Имена понятий предметной области либо класса предметных областей, имеющих одну и ту же метаинформацию; Тип структурированного набора,

т.е. σV(Neterminal_Vertexes) ∈ (Имена понятий предметной области × Тип структурированного набора). Назовем множество дуг графа, выходящих из одной вершины, структурированным набором дуг. Граф метаинформации имеет два возможных типа структурированных наборов - непустое множество и непустое множество альтернатив, т.е. Тип структурированного набора∈{Непустое множество, Непустое множество альтернатив}.

Терминальные вершины могут иметь разметку двух типов – терминал, описывающий сорт, и терминал, описывающий значение. Терминал, описывающий сорт, есть пара Терминал, описывающий $Сорт_i$=<$Имя_i$, $Значение_i$>, где $Имя_i$ – строка символов конечной длины, $Значение_i$ принадлежит множеству имен базовых сортов. Терминал, описывающий значение, есть тройка Терминал, описывающий $Значение_i$=<$Имя_i$, $Сорт_i$, $Значение_i$>, где $Имя_i$ – строка символов конечной длины, $Сорт_i$ ∈ Имена Базовых сортов, $Значение_i$ ∈ $Сорт_i$. Разметка дуг σA задается следующим образом: σA(Arc_i) = Спецификатор дуги. Спецификатор дуги есть пара <Тип спецификатора, Факультативность>, т.е. Спецификатор дуги=<Тип спецификатора, Факультативность>, Тип спецификатора ∈{копия, единственность, непустое множество}. Все спецификаторы могут использоваться в сочетании со спецификатором «факультативность», однако спецификатор «факультативность» могут иметь лишь дуги, входящие в структурированный набор с типом «непустое множество».

Информация, как и метаинформация, представляет собой пару И=(GI, σI), где GI – граф (сеть) понятий, который в отличие от графа метаинформации, не содержит циклов и петель, σI - разметка графа. Граф информации GI есть тройка <IVertexes, IArcs, IRootVertex>, где множество вершин графа (IVertexes), множество дуг графа (IArcs), корневая вершина (IRootVertex) описываются также как в графе метаинформации, средства разметки σI, в отличие от графа метаинформации, включают в себя только разметку вершин, т.е. σI ≡σV. Все вершины графа информации отображаются в пару σV=(SM, RM), где SM – служебная метка; RM – метка соответствия между информацией и метаинформацией. Служебные метки могут быть двух типов - порожденные вершины (вершины, из которых выполнен шаг порождения) и непорожденные (вершины, из которых еще не выполнен шаг порождения). Метка соответствия RM - это Имя понятия предметной области, которому соответствует либо Имя понятия предметной области, заданное в метаинформации, либо некоторое имя, входящее в класс предметных областей, также заданный в метаинформации.

Соответствие между информацией и метаинформацией задается следующим образом. Каждой вершине в графе информации соответствует некоторая единственная вершина в графе метаинформации, называемая ее вершиной-прототипом. Количество вершин в графе информации определяется типом структурированного набора, заданного в разметке графа метаинформации. Так, если в разметке некоторой вершины-прототипа графа метаинформации структурированный набор задается типом «непустое множество», то в графе информации из вершины, соответствующей этой вершине-прототипу выходит i дуг, где 0≤i≤N, N - количество дуг, выходящих из вершины –прототипа в графе метаинформации; если в разметке вершины-прототипа структурированный набор задается типом «непустое множество альтернатив», то в графе информации из вершины, соответствующей вершине-прототипу, выходит единственная дуга.

Структура и разметка графа информации также зависит от типа спецификатора дуги метаинформации. Если в графе метаинформации дуга имеет Тип спецификатора= копия, то разметка вершины, в которую входит дуга со спецификатором данного типа, сохраняется в графе информации; если в графе метаинформации дуга имеет Тип спецификатора=единственность, то этой дуге в графе информации соответствует только одна дуга, входящая в вершину, разметка которой соответствует Имени предметной области (у нетерминалов) или значению (у терминалов); если в графе метаинформации дуга имеет Тип спецификатора=непустое множество, то этой дуге в графе информации соответствует множество дуг, входящих в вершины, разметка которых соответствует некоторому Имени предметной области (у нетерминалов) или значению (у терминалов). Факультативность, указанная в разметке дуги графа метаинформации означает, что в графе информации может отсутствовать понятие (термин) из метаинформации дуга, и, соответственно, вершина, в которую эта дуга входит.

Граф метаинформации можно рассматривать как порождающую графовую грамматику (грамматику, порождающую графы информации), определяющую язык, в терминах которого может быть порождено множество графов информации на этом языке.

Для компьютерного представления метаинформации и информации разработан метаязык (язык ИРУО), реализующий описанную выше модель, и универсальный редактор (редактор ИРУО) [Клещев и др., 2006b], имеющий два режима. Для формирования метаинформации на языке ИРУО используется режим инженера знаний. С помощью структурного редактора инженер знаний формирует на языке ИРУО размеченный граф метаинформации. По этому графу метаинформации редактор ИРУО в режиме эксперта генерирует интерфейс для эксперта (специалиста) предметной области. Информация, сформированная экспертом с помощью редактора ИРУО в режиме эксперта, также сохраняется на языке ИРУО. Таким

образом, редактор по сути является интерпретатором метаинформации, представленной на языке ИРУО, который формирует информацию также на языке ИРУО. Такой подход обеспечил реализацию единственного редактора как метаинформации, так и информации для любой метаинформации, представленной на языке ИРУО.

2. Решатель задач

Основной задачей при разработке решателей задач является создание универсальных методов их разработки из повторно-используемых компонентов в условиях полного отделения процедурных знаний (знаний о решении задач) от декларативных знаний (баз знаний) для снижения трудоемкости проектирования и сопровождения.

В предлагаемой концепции все информационные ресурсы имеют единое унифицированное представление – семантические сети; в этом случае решатель задач можно рассматривать как программу обработки информационных ресурсов, представленных таким образом. Результатом решения задачи является либо формирование нового информационного ресурса, либо модификация входного информационного ресурса

Вычислительный процесс, выполняемый решателем задач, состоит в построении графа выходной информации по графу выходной метаинформации и входным данным, если граф выходной информации неполностью порожден, либо в модификации этого графа, если он, как входные данные, полностью порожден (возможен и смешанный вариант).

Решатель задач представляет собой совокупность агентов. Агент - повторно используемый программный компонент, взаимодействующий с другими агентами посредством приема и передачи сообщений. Агент состоит из двух частей – декларативной и процедурной. Декларативная часть включает описание множества блоков. Каждый блок есть пара – метаинформация сообщения (шаблон сообщения) и множество продукций. Метаинформация сообщения определяет язык, на котором представлена информация в сообщении, посылаемая этому блоку этого агента.

Продукция состоит из условия (антецедента) и действия (консеквента). Антецеденты продукций блока используются для анализа сообщения, выбора применимых методов и передачи им информации из сообщения; консеквент по сути является вызовом такого метода, который выполняет обработку информации, выбранной антецедентом из сообщения для решения связанной с агентом подзадачи. Для каждого блока продукций определен свой шаблон сообщения. Результатом работы метода может быть формирование и посылка сообщения конкретному агенту, либо подмножеству агентов, ограниченных некоторым условием, а также шаги формирования или модификации некоторых информационных ресурсов.

Агенты могут быть двух основных типов: повторно-используемыми и специализированные. Повторно-используемыми агентами являются:

− проблемно-независимые вычислительные агенты, решающие стандартные часто-используемые вычислительные задачи;

− агенты, обрабатывающие информационные ресурсы; обрабатывающими будем называть агенты, осуществляющие шаги порождения и модификации информационных ресурсов, а также запрос информации из них; повторная используемость агентов этого типа достигается за счет того, что все информационные ресурсы имеют единое унифицированное представление, а при порождение и модификации информационных ресурсов поддерживается соответствие между метаинформацией и информацией;

− интерфейсные агенты, осуществляющие взаимодействие с пользователями.

Специализированные агенты – это вычислительные агенты, решающие узкоспециализированные подзадачи, либо управляющие агенты (через которые осуществляется координация вычислений, выполняемых различными агентами).

В этом случае решатель задач – это совокупность повторно-используемых и, возможно, специализированных агентов. Единственное ограничение на состав агентов внутри решателя – они «должны знать» шаблоны сообщений агентов, которым они посылают и от которых они принимают сообщения. Поскольку решатель имеет формальные параметры (входные и выходные), описываемые через метаинформацию информационных ресурсов, каждый решатель применим к классу задач, связанных с этой метаинформацией.

Проектирование решателя задач состоит в декомпозиции основной задачи формирования или модификации информационных ресурсов на совокупность подзадач, решаемых агентами, с учетом ранее разработанных повторно-используемых агентов, и определении взаимосвязей между этими подзадачами.

Объединение агентов (подзадач) в решатель может осуществляться двумя способами:

− через посылку сообщений; сообщение может быть послано либо по обратному адресу, либо в сообщении указывается конкретный адресат;

– созданием специального информационного ресурса – управляющего графа, через который агенты связываются между собой.

Использование единого формата представления метаинформации и информации (язык ИРУО) позволило разработать конечное множество программных интерфейсов (функций), которые

обеспечивают универсальный доступ к информационным ресурсам для их обработки компьютерными программами (агентами). При этом они устроены таким образом, что поддерживается соответствие между информацией и метаинформацией. Использование стандартного набора программных интерфейсов освобождает программиста от дополнительных усилий (написание программного кода) для обеспечения доступа к представлению информационных ресурсов.

3. Пользовательский интерфейс

При разработке пользовательских интерфейсов для интеллектуальных систем необходимо разработать технологию их создания, удовлетворяющую следующим требованиям:

- реализация пользовательского интерфейса, как web-интерфейса, т.е. поддерживающего взаимодействие пользователя с интеллектуальной системой через браузер;
- поддержка WIMP-интерфейса;
- адаптация пользовательского интерфейса к типам пользователей и характеристикам входных/выходных данных.

Первое и второе требования обусловлены необходимостью создания интеллектуальных систем как интернет-сервисов; при этом важно сохранение традиционного и привычного для большинства пользователей WIMP-интерфейса. Третье требование обусловлено стандартами юзабилити, которые прописывают набор правил организации удобного в использовании и дружественного интерфейса для различных категорий пользователей в зависимости от характеристик входных/выходных данных.

Пользовательский интерфейс реализуется через повторно-используемых агентов. Для упрощения разработки пользовательского интерфейса в состав агентной платформы входит специальный агент «Вид». Агент «Вид» содержит реализации всех интерфейсных элементов, поддерживаемых текущей версией (текстовое поле, кнопка управления, поле ввода, переключатели – флажки и радио-кнопки, пролистываемый и раскрывающийся списки, комбобокс, ссылки). Он осуществляет прорисовку элементов и реализует их поведение. Расширение множества поддерживаемых интерфейсных элементов возможно через модификацию агента «Вид». Интерфейсные элементы могут порождать различные события, вызванные действиями пользователя или системы (ввод символов с клавиатуры, выбор элемента, срабатывание системного таймера и др.). Поступившие события обрабатываются агентом «Вид», который формирует сообщения агенту решателя, далее это сообщение обрабатывается этим агентом и посылается новое сообщение либо другому агенту решателя, либо агенту «Вид», который формирует очередное состояние пользовательского интерфейса и отображает его через клиентское программное обеспечение (браузер).

Помимо агента «Вид» выделен набор интерфейсных агентов, осуществляющих адаптацию интерфейсной задачи к данным, с которыми она связана, т.е. интерфейсный агент выбирает адекватный интерфейсный элемент либо их группу, соответствующие данным, из множества реализованных в агенте «Вид».

В общем случае каждый интерфейсный агент во входном сообщении получает имя агента решателя, который посылает ему сообщение, и множество входных данных для выбора интерфейсного элемента; выходное сообщение интерфейсного агента, которое он посылает агенту «Вид», включает имя агента решателя, входные данные, интерфейсный элемент (элементы). Имя агента решателя необходимо для того, чтобы агент «Вид» посылал выходную информацию напрямую агенту решателя, минуя интерфейсный агент.

4. Заключение

В работе описана технология разработки и сопровождения интеллектуальных систем как интеллектуальных сервисов. В основе предлагаемой технологии разработки интеллектуальных систем лежит принцип четкого разделения между декларативными знаниями (знаниями предметной области) и процедурными знаниями (о методе решении задачи). Дальнейшее развитие получил двухуровневый подход к созданию баз знаний и сложно-структурированных баз данных, который предполагает, что вся информация представлена в виде неразрывной пары метаинформация-информация; информация формируется экспертом по метаинформации, созданной инженером по знаниям. В отличие от технологии Протеже, которая также предлагает формирование знаний по метаинформации, в предлагаемой технологии жестко разделены терминология инженера по знаниям и эксперта, что делает возможным формирование и сопровождение баз знаний и данных непосредственно носителями этой информации, т.е. экспертами и специалистами предметной области без посредников в лице инженеров знаний.

Поддержка процесса формирования и сопровождения информационных ресурсов обеспечивается универсальным редактором, который по метаинфорации, созданной инженером по знаниям, генерирует пользовательский интерфейс для эксперта. В системе Протеже также имеется универсальный редактор знаний, управляемый метаинформацией, однако, по сути там предлагается общий редактор для инженеров по знаниям и экспертов, а отсутствие четкого разделения между их терминологиями делает практически невозможной самостоятельную работу

экспертов предметной области без участия в ней инженеров по знаниям. Дополнительным различием между редакторами Протеже и ИРУО является предоставление последнего как облачного сервиса.

Все информационные ресурсы имеют единый унифицированный формат представления – семантическую сеть. Это обеспечивает возможность доступа ко всем типам ресурсов (как данным, так и знаниям) через оболочку, предоставляющую набор программных интерфейсов, скрывающий формат их внутреннего представления, что значительно упрощает к ним доступ, поскольку разработчику не надо знать детали внутренней организации информации и программировать доступ к ней.

Решатель задач представляется в виде совокупности агентов, где каждому агенту соответствует подзадача решаемой задачи. Агент является повторно-используемым компонентом, представленным семантической сетью и связанным с ней множеством блоков продукций. В отличие от технологии Протеже, которая фиксирует классы задач, решаемых интеллектуальной системой, в данной технологии нет такого ограничения, каждый агент связан с уникальной подзадачей, которая может быть как проблемно-независимой (что расширяет круг ее использования), так и проблемно-ориентированной на класс задач, ограниченный некоторой метаинформацией (проблемно-ориентированный агент в общем случае связан с фрагментом метаинформации и, соответственно, может быть применен к обработке любого фрагмента информации, которая создается или модифицируется по тому фрагменту метаинформации, с которой связан агент).

Пользовательский интерфейс в отличие от Протеже, не является частью метода решения задачи. Он реализуется через системный агент «Вид», который является компонентом платформы и реализует поведение пользовательского интерфейса на основе входящих в него реализаций интерфейсных элементов, а также набора дополнительных интерфейсных агентов, которые обеспечивают адаптацию интерфейса к пользователю и наборам входных данных. Пользовательский интерфейс реализуется через браузер, что обеспечивает также его кроссплатформенность и возможность использования на всех компьютерах, включая мобильные устройства.

Платформой для разработки интеллектуальных интернет-сервисов является Интернет-комплекс IACPaaS (Intelligent Application, Control and Platform as a Service) [Грибова и др., 2011], предоставляющий контролируемый доступ и единую систему администрирования для создания интеллектуальных сервисов и их компонентов, представленных семантическими сетями и агентами (в том числе интерфейсными), поддержку функционирования сервисов и агентов (через передачу и обработку сообщений, запуск методов, а также распараллеливания вычислений в сервисах).

В основе платформы лежит программно-информационный комплекс, основанный на технологии облачных вычислений и обеспечивающий удаленный доступ конечных пользователей к интеллектуальным системам, а разработчикам и управляющим – к средствам создания интеллектуальных систем и их компонентов, а также управления ими.

Работа выполнена при финансовой поддержке ДВО РАН (инициативный научный проект 12-I-ОНИТ-04) и при финансовой поддержке РФФИ (проект 12-07-00179-а).

Библиографический список

[Bachant et al., 1984] Bachant, J. McDermott, J. "R1 revisited: four years in the trenches", The AI Magazine (Fall 1984; Fall), pp.21-32

[Compton, 1988] Compton, P. Horn, K. Quinlan, R. Lazarus, L. "Maintaining an expert system". Proceedings of the fourth Australian Conference on Applications of Expert Systems, (1988), pp.110-129.

[Gennari et al., 2003] Gennari, J.H.; Musen, M.A.; Fergerson, R.W, etc. The evolution of Protégé: An environment for knowledge-based systems development/International Journal of Human-Computer Studies. 2003. 58(1):89-123

[Грибова и др., 2011] Грибова В.В., Клещев А.С., Крылов и др. Проект IACPaaS – развиваемый комплекс для разработки, управления и использования интеллектуальных систем // Искусственный интеллект и принятие решений. – 2011. – №1.

[Клещев и др., 2006a] Клещев А.С., Орлов В.А. Компьютерные банки знаний. Многоцелевой банк знаний// Информационные технологии, № 2, 2006. С. 2-8.

[Клещев и др, 2006b] Клещев А.С., Орлов В.А. Компьютерные банки знаний. Универсальный подход к решению проблемы редактирования информации. – Информационные технологии. – 2006. – №5.

[Пивоварова и др., 2007] Л. М. Пивоварова, В. Ш. Рубашкин Компоненты онтологических систем и их реализация в современных проектах//Интернет и современное общество: Труды X Всероссийской объединенной конференции. СПб.: Факультет филологии и искусств СПбГУ, 2007. с.277-279.

CLOUD AND SEMANTIC TECHNOLOGIES FOR INTELLIGENT SERVICES DEVELOPMENT

Gribova V, Kleschev A.

Institute of Automation and Control Processes, Far Eastern Branch of Russian Academy of Sciences, Vladivostok
gribova@iacp.dvo.ru, kleschev@iacp.dvo.ru

A technology of development and maintenance of intelligent systems as intelligent services is presented. The proposed technology of intelligent system development based on the principle of clear separation between declarative knowledge (domain knowledge) and procedural (the method of solving the problem). The principles of the formation of each component of an intelligent system (knowledge base, task solver, and user interface) are described.