УДК 004.4`2+004.89

КОМПЛЕКС СРЕДСТВ ПОДДЕРЖКИ ПРОЦЕССОВ РАЗРАБОТКИ И СОПРОВОЖДЕНИЯ РЕШАТЕЛЕЙ ДЛЯ СИСТЕМ С ОНТОЛОГИЧЕСКИМИ БАЗАМИ ЗНАНИЙ

Грибова Валерия Викторовна
д.т.н, с.н.с, зам.директора по научной работе
e-mail: gribova@iacp,
Шалфеева Елена Арефьевна,
к.т.н, доцент, с.н.с.
e-mail: shalf@dvo.ru,
Институт автоматики и процессов управления ДВО РАН, Владивосток,
690041, Приморский край, г. Владивосток, ул. Радио, 5

Аннотация. Использование онтологического подхода является одним из современных подходов к созданию систем с базами знаний. Для построения жизнеспособных программных сервисов, работающих с такими базами знаний, и управления их коллективной разработкой предложена инструментальная среда. Метод инструментальной поддержки нацелен на создание и развитие библиотек онтолого-базированных операций, на использование их при конструировании программных средств, на распределение полномочий по созданию компонентов систем с базами знаний, на контроль и интеграцию их в облачную сопровождаемую систему поддержки принятия решений на основе знаний.

Ключевые слова: онтологическая база знаний, алгоритм решения, решатель задачи, повторное использование, сопровождение, коллективная разработка.

Цитирование: Грибова В. В., Шалфеева Е.А. Комплекс средств поддержки процессов разработки и сопровождения решателей для систем с онтологическими базами знаний // Информационные и математические технологии в науке и управлении. 2020. № 4 (20). С. 34-43. DOI:10.38028/ESI.2020.20.4.003

Введение. Современным подходом к созданию *систем с базами знаний* является использование онтологического подхода. Все преимущества такого способа создания баз знаний активно обсуждаются в литературе и, в настоящее время, как отмечено в [1], являются стандартом «де факто».

Технология разработки баз знаний на основе онтологий достаточно широко обсуждается в литературе, для разработки онтологических баз знаний предложен ряд инструментальных средств и платформ, типичными представителями являются OSTIS, Protégé, OntoEdit, IACPaaS и ряд других [2, 7,].

Вместе с тем проблема разработки решателей задач, основанных на онтологических базах знаний, освещена в литературе гораздо меньше. Традиционно считается, что в качестве решателя системы с базой знаний (СБЗ) выступает машина вывода, обрабатывающая продукционные правила, алгоритмический подход не применим к созданию такого класса систем, именно поэтому часто онтологические базы знаний преобразуют в базу правил и для нее разрабатывают машину вывода [9]. Для создания решателей используются и

специализированные, нетрадиционные модели и технологии, например, технология OSTIS, в которой решатель, по определению авторов, рассматривается в «неклассическом варианте» и представляет собой графодинамическую *SC*-машину [3].

При разработке онтологических баз знаний, представленных семантическими сетями, решатель задач может быть представлен в виде алгоритма, выполняющего обход базы знаний на основе знания ее структуры (онтологии) для сопоставления информации из базы знаний входным данным [6]. Очевидно, что разработка решателя для онтологических СБЗ должна отвечать всем основным требованиям к разработке программных систем: обеспечение снижения трудоемкости разработки и сопровождения, прозрачность, поддержка коллективной разработки. Первые три требования реализуются через применение специализированных оболочек, декларативно-компонентный метод разработки решателей, упрощение интерфейсов и уменьшение сцепления программных компонентов, их повторную используемость.

Применение специализированных оболочек, ориентированных на конкретный класс систем с базами знаний, обеспечило заметное сокращение сроков разработки и усилий на сопровождение систем, но у специализированных оболочек "жесткий" решатель и встроенный в него интерфейс, которые, в случае изменения требований, трудно модифицировать. Альтернатива оболочкам - применение объектно-ориентированного программирования и архитектуры MVC. При их использовании «в коде создаются программные объекты, описывающие свойства моделируемых объектов реального мира и операции над ними», поэтому «в программном коде прикладных автоматизированных систем все еще содержится много логики, связанной непосредственно с предметной областью и конкретными решаемыми задачами». «Очевидным ограничением этого подхода является необходимость модифицировать код при внесении изменений в структуру модели» [4].

Для коллективной разработки необходимо обеспечивать каждый тип разработчиков собственным «рабочим местом», предоставляющим ему необходимую функциональность для исполнения своих задач.

Современные технологии Protege, OSTIS и IACPaaS [2, 6, 7] поддерживают работу по созданию знаний и работе с ними на дистанционно доступных собственных платформах. На базе инструментального комплекса АТ-Технология разработана веб-ориентированная версия для поддержки построения прикладных интегрированных экспертных систем на всех традиционных этапах их создания [5].

Целью данной работы является описание комплекса средств поддержки процессов разработки и сопровождения решателей для систем с онтологическими базами знаний, реализованных в виде алгоритма.

1. Поддержка разработки и сопровождения решателей: онтологические решатели в системах с базами знаний. Онтологический решатель в общем случае обрабатывает несколько информационных ресурсов *(ИнфРес)*, сформированных на основе онтологии.

Онтология предметной области ПрОбл определяет правила представления и обработки информации (анализ, формирование, модификация) и включает

$$Onto = KnOnt \cup SitOnt \cup Agreem \cup Dict,$$

где *KnOnt* определяет структуру профессиональных знаний о решении задачи,

SitOnt – структуру информации об объектах действительности,

Agreem – онтологические соглашения о правилах обработки, т.е. сопоставления фактов и знаний в процессе рассуждения и принятия решений, а также о правилах выдвижения,

подтверждения, опровержения гипотез, содержащихся в знаниях или создаваемых с их помощью;

Dict – словарь названий и области значений понятий ПрОбл.

Онтологический решатель *Solv (KnOnt∪SitOnt∪Agreem*) в составе системы с базой знаний (СБЗ) должен быть способен выдвигать гипотезы по результату сопоставления входной информации (об объекте) – предложениям из базы знаний (Knowledge Base, KB) *(KnOnt, Dict)* на основе онтологических соглашений *Agreem* о связи данных и знаний и с учетом знания структуры предложений (связей терминов) *KnOnt*, и ограничений на интерпретацию смысла терминов. Задача онтологического решателя – предложить (путем «рассуждения») одно или несколько обоснованных решений-гипотез, использующих входную информацию и согласованных с Базой знаний KB *(KnOnt)*. Обоснование каждой гипотезы – явное указание ее связи с теми предложениями Базы знаний, которые ее подтверждают или не отвергают.

При применении соглашений о соответствии фактов знаниям о рассматриваемых явлениях, связываемые понятия используются для промежуточных заключений. Часто в роли *посылок* (причин для принятия промежуточных решений) выступают факты из ситуаций действительности и ожидаемые, согласно знаниям, проявления анализируемых процессов. *Следствиями* являются значения предикатов их соответствия. Обработка – это поиск (в документе, описывающем действительность) фактов для подтверждения условий течения процессов (и «вывод»/получение значений соответствующих предикатов); вывод значений предикатов (на множестве условий, на множестве проявлений) – предикатов существования гипотез из элементов модели знаний по полученным значениям предикатов-«посылок», выдвижение гипотез на основе соответствующих вычисленных предикатов (подтвержденных или (не-) отвергнутых посылок).

Онтологический решатель (в технологии IACPaaS [6]) – программно реализованное «рассуждение», алгоритм, проводящий обход каждой Базы знаний KB (представленной семантической сетью) для сопоставления входным данным (фактам или условиям) знаний и соглашений, выдвижения и объяснения гипотез и построения семантической сети, фиксирующей этапы рассуждения – шаги принятия или отклонения возможных логических заключений.

2. Проектирование онтологического решателя из программных единиц. Решатель (умеющий выдвигать гипотезы и объяснять их) планируется к реализации из программных единиц *(ПрЕд)*, среди которых единицы ($Unit_m$) разных типов, с доступом и без доступа к *ИнфРес* (информационным компонентам СБЗ), для вывода (или промежуточного заключения), для поиска фактов, для вычислений, для связи с внешним окружением.

ПрЕд для вывода – запрашивает и получает информацию о подтверждении или отрицании элементов отношений или их цепочек, прописанных в *Kn* (часто как ответ с областью значений {данные соответствуют; противоречат; отсутствуют}). Например, *ПрЕд* проверяет истинность или подтвержденность варианта развития процесса для подтверждения существования процесса; подтвержденность последовательности периодов развития для подтверждения Варианта. Программная единица, обрабатывая такие виды связей между элементами *ИнфРес*, делает промежуточные заключения процесса логического вывода.

ПрЕд ($Unit_m$), которые делают некоторое заключение о соответствии подмножества фактов знаниям некоторого типа (причинно-следственных или других структурных связей

между понятиями), «имеют дело» с *Agreem*. В совокупности такие *ПрЕд*, как правило «покрывают» все утверждения из *KnOnt*∪*Agreem*. Решатель, создаваемый из *ПрЕд,* программируемых по такому принципу – онтологический.

В алгоритме решателя происходит поочередный вызов *ПрЕд* – процедур вывода следствий из обработанных посылок, которые записывают результат проверки посылок в объяснение. Например, *ПрЕд* «*проверить вариант развития процесса»* активируется после активации *ПрЕд* «*проверить гипотезу о процессе»* и после активации *ПрЕд* «*проверить необхУсловия процесса». Она реализует* процедуру вывода заключения о подтверждении или опровержении наличия *комплекса признаков* из обработанных результатов подтверждения или опровержения всех обязательных *признаков* и всех дизъюнктивных наборов признаков.

ПрЕд для поиска фактов получает список условий на события и прочие возможные факторы, ищет в документе названия таких наблюдений, сравнивает с условиями и дает ответ: данные соответствуют условию или данные не соответствуют или данные отсутствуют; она также получает значения наблюдений, ищет в документе названия таких наблюдений, сравнивает с условиями и дает ответ: {данные соответствуют; противоречат/ не соответствуют; отсутствуют}.

В составе решателя также могут быть и другие *ПрЕд*, не нуждающиеся в доступе к *ИнфРес* – вычислительные, интерфейсные.

3. Онтологические операции доступа к информации как вид программных единиц и метод инструментальной поддержки коллективного развития библиотек операций. При разработке алгоритма для каждого шага вывода требуется либо обращение к элементам описания модели (знаний), либо к фактам действительности, либо к тому и другому.

ПрЕд для поиска фактов и *ПрЕд для вывода* обращаются с запросами к той КВ, где содержатся знания об известных связях, типы которых зафиксированы в онтологии знаний. Например, *ПрЕд* для поиска методов для воздействия на признак для изменения его значений в заданном направлении *getMethodsForTrendSign* реализует отношения «состояние – воздействие – результат» (между методом воздействия, признаком(-ами) до начала воздействия и признаком(-ами) желаемыми\целевыми).

При поиске фактов для подтверждения условий существования искомых явлений или признаков их проявления (т.е. «посылок» для вывода) требуется обращение к фактам действительности (к возрасту, к размеру, к температуре…). Каждое такое обращение за фактом или их комплексом – «кандидат» на самостоятельную операцию доступа Op_i к документу с данными $Unit_m$, которые делают некоторое заключение о соответствии подмножества фактов знаниям некоторого типа, сопоставляют результаты выполнения *операций* над информацией *Sit* и над Базами знаний КВ. *Операции* могут получать на вход список параметров, на выходе обеспечивать вычисленные значения либо модификацию *ИнфРес*, например, объяснения или создаваемого плана\проекта (породить экземпляр указанного понятия, записать значение указанного элемента).

Операции доступа к содержимому онтологических *ИнфРес* наиболее связаны со структурными и причинно-следственными связями используемых понятий (определяемыми *KnOnt* и *SitOnt*), поэтому являются онтологическими. С конкретной онтологией связывается множество таких операций, чтобы применяться затем к любым *ИнфРес* (знаниям, данным), управляемым этой онтологией. Op_i осуществляют запросы к элементам структуры *ИнфРес* в соответствии с правилами интерпретации и соглашениями.

Набор операций отражает востребованные запросы, возникающие в переборных алгоритмах решения классов задач. Формирование и использование операций доступа Op_i для управляемого программного доступа к целевым базам знаний и другим хранимым *ИнфРес* требует библиотек онтолого-базированных операций доступа к *ИнфРес.*

В процессе разработки (и сопровождения) онтологического решателя IACPaaS и его *ПрЕд* (IACPaaS-агентов) предлагается использование готовой операции (которую можно вызывать из агента). Разработка новой операции – это декларативное описание операции, редактирование, сохранение и компиляция исходного кода операции с использованием средства «Редактор библиотеки операций».

При генерации заготовки агента, исходя из его декларативного описания с *указанием онтологий* (из $KnOnt \cup SitOnt$) обрабатываемых *ИнфРес*, предложено оповещать о наличии операций над такими *ИнфРес*, предложить список их имен и описаний с возможностью сгенерировать в коде шаблон-заготовку вызова-обращения к выбранной операции для *ИнфРес*.

Таким образом, эффективное конструирование (снижение трудозатрат) решателей ряда практических задач основано на повторном использовании программных единиц и операций. Одни и те же виды структурных и причинно-следственных связей могут приниматься во внимание при решении разных классов задач (прогноз, диагностика, планирование и др.) по единой модели течения процесса. В частности, построение заключения о соответствии подмножества фактов знаниям некоторого типа причинно-следственных связей, характерного для задач разных классов (например, диагностики и прогноза) является часто выполняемой обработкой. Для Решателей таких задач (или интегрирующих их составных задач) шанс повторно использовать $Unit_m$, которые делают такое заключение, увеличивается. Даже при создании одного Решателя частые (повторяемые) варианты поиска, сравнения, выбора, добавления информации рекомендуется оформлять в самостоятельные *ПрЕд*, обрабатывающие *ИнфРес*, для их повторного использования.

4. Интегрированная среда построения жизнеспособных программных компонентов. Для обеспечения сборки Решателя из заменяемых компонентов и повышения прозрачности его архитектуры актуальны средства декларирования всех составных частей – *ПрЕд* (и порядка их запуска), контроля доступности используемых ими *ИнфРес*, стандартизация их взаимодействий (шаблоны обращений), что особенно важно в условиях коллективной разработки и развития. Следовательно, важна инструментальная поддержка декларирования решателя (произвольной задачи) и декларирования его компонентов.

Декларация онтологической *ПрЕд* содержит среди параметров имена обрабатываемых *ИнфРес* и дополнительно содержит список ссылок на онтологии этих *ИнфРес*. Декларирование включает и описание назначения *ПрЕд*, что способствует их повторному использованию. Декларация *ПрЕд* содержит формат обращения к ней со списком типов параметров. Для интегрируемости онтологических *ПрЕд* (*SemUnit и SemOp*) их внешние связи (форматы знаний и данных, входных и выходных сообщений) тоже специфицируются (в соответствующей декларации этой агента).

Создаваемая IACPaaS-декларация решателя – это документирующий ИнфРес, он содержит список ссылок на онтологии всех обрабатываемых *ИнфРес*, читаемых и формируемых. Такая спецификация Решателя может быть интерпретируема человеком и машиной. Кроме спецификации форматов знаний и данных, могут быть определены все $Unit_m$

и даже порядок их выполнения и связей друг с другом. Декларация Решателя может использоваться (1) автоматически для контроля сборки, для динамической (runtime) загрузки частей по мере потребности в них, (2) человеком – для сборки СБЗ из Решателя и всех обрабатываемых *ИнфРес*, а также для его сборки из *ПрЕд,* если для них создаются свои специфицирующие документы. Проектирование архитектуры решателей задач из декларируемых программных компонентов разных типов – шаг в направлении создания жизнеспособных систем (и еще один ключевой принцип проектирования). В декларации решателя задач, как минимум, указывается агент, начинающий работу. Другие $Unit_m$ могут явно не перечисляться, а быть указаны внутри "первого" агента при передаче им управления.

Кроме средств (инструментальных сервисов) декларирования нужны средства создания заготовок исходных кодов по декларациям, средства кодирования новых *ПрЕд,* средства каталогизации $Unit_m$ как потенциально повторно используемых компонентов и средства интеграции ПИК и новых *ПрЕд* в новые решатели или их новые версии. Поэтому в *среде* декларативно-компонентного метода разработки и развития СБЗ инструментарий для формирования *программных решателей* включает:

- онтологию *ПрОбл* (включая базу терминов),
- инструменты декларирования решателя и *ПрЕд,*
- среда разработки программ (IDE) для кодирования *ПрЕд.*
- библиотеки подготовленных и протестированных *ПрЕд,* обрабатывающих ИнфРес.

Методы и инструменты декларирования решателя нацелены на обеспечение прозрачности его архитектуры, а декларирование *ПрЕд* – на поддержку создания их исходных кодов и динамическое подключение в процессе выполнения.

5. Поддержка коллективной разработки программных компонентов. В соответствии с традициями инструментальной поддержки разработки интеллектуального сервиса (Типовые Программные Процедуры поддержки ресурсно-календарного планирования [5]) в IACPaaS среде на основе декларативного описания процесса разработки создаются инструменты планирования работ, ответственных исполнителей, сроков, передачи результатов производства компонентов СБЗ и фрагментов этих компонентов.

В *онтологии коллективной разработки интеллектуальных сервисов*, по которой может быть построено декларативное описание очередного Проекта, выделяются основные типы участников разработки, всевозможные действия и полномочия участников. Участники разработки реализуют свои полномочия в собственных кабинетах – «рабочих местах».

Для разработки СБЗ строится дерево задач. Данное дерево отображает, что необходимо сделать для создания интеллектуального сервиса, а каждая вершина является задачей для конкретного исполнителя. В вершинах обозначены роли ответственных за исполнение задачи. Корневой вершине соответствует всё дерево, а исполнение её задачи соответствует завершению всего проекта по разработке интеллектуальной системы.

Задача имеет ряд атрибутов, среди которых – стадия. С помощью стадий определяется жизненный цикл задачи. В отношении задачи всегда существуют участники разработки в двух ролях: автор задачи, который внёс задачу в систему и выбрал её исполнителя, и, непосредственно, исполнитель задачи. В зависимости от роли, пользователю доступны разные переходы по стадиям.

Для управления коллективной разработкой предоставляется функциональность для создания *дерева задач процесса разработки* интеллектуального сервиса на основе IACPaaS-технологии разработки сервисов, основанных на знаниях.

Инициатор разработки или *руководитель Проекта* должен назначить исполнителей на все задачи в дереве. В свою очередь, исполнители задачи могут декомпозировать (на несколько подзадач) или делегировать (всю) свою задачу. В этом случае к вершине дерева задач добавляются новые вершины. Исполнитель имеет возможность исполнить свою задачу (перемещая её по стадиям в рамках жизненного цикла) и возможность управлять задачей – например, создавать дочерние задачи, предоставлять результат исполнения. По мере исполнения задач проводится управляемое объединение результатов задач.

Комплекс инструментальных сервисов поддержки процесса коллективной разработки создается на основе онтологического подхода. В этом случае технология разработки рассматривается как *ПрОбл*, а онтология процесса разработки является основой онтологических *программных средств разработки* (инструментальных сервисов). В частности, инструментальные сервисы декларирования решателя и *ПрЕд* – это Редакторы (соответствующей информации), генерируемые по составной части онтологии этой технологии – по документу, описывающему содержание и формат декларации.

Декларативное определение процесса коллективной разработки позволяет наглядно представлять процессы и состояние разработки.

Заключение. На платформе IACPaaS [6], используемой для дистанционного создания и использования онтологических баз знаний, разработан комплекс средств поддержки процессов разработки и сопровождения онтологических решателей.

Онтологический решатель обрабатывает несколько информационных ресурсов, сформированных на основе онтологии. В процессе рассуждения и принятия решений алгоритм опирается на формализованные связи и зафиксированные онтологические соглашения. Особенности, принципиально отличающие онтологический решатель от других решателей:

- алгоритм формируется на основе онтологии и не зависит от самих БЗ, что соответствует современному подходу к разработке интеллектуальных систем;
- в нем «естественно» реализуется подробное и понятное специалисту объяснение предлагаемого решения.
- Для снижения трудоемкости разработки и сопровождения онтологического решателя предложены следующие решения:
- декларативное представление всех программных единиц;
- декларативное представление решателя из составных частей, включая порядок их запуска;
- контроль доступности используемых *ИнфРес*,
- генерация фрагментов кода по этому представлению,
- инструменты поддержки коллективного использования и развития библиотеки онтологических операций доступа к *ИнфРес*.

Инструментальной средой для создания таких решателей задач является облачная платформа IACPaaS – среда для построения проблемно-ориентированных онтологий и решателей соответствующих классов задач. Под управлением проблемно-ориентированных онтологий (с общим доступом) участниками процесса создаются фрагменты баз знаний и

ПрЕд для решателей в рамках коллективной разработки СБЗ, обладающих свойством жизнеспособности.

Работа выполнена при частичной финансовой поддержке РФФИ (проекты №№ 19-07-00244, 18-07-01079).

СПИСОК ЛИТЕРАТУРЫ

1. Гаврилова Т.А., Страхович Э.В., Визуально-аналитическое мышление и интеллект-карты в онтологическом инжиниринге // Онтология проектирования. 2020. Т. 10. № 1. С. 87-99.

2. Голенков В.В., Гулякина Н.А. Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 2: унифицированные модели проектирования // Онтология проектирования. 2014. №4(14). С 34-53;

3. Голенков В.В., Д.В. Шункевич, Давыденко И.Т. Семантическая технология проектирования интеллектуальных решателей задач на основе агентно-ориентированного подхода // Программные системы и вычислительные методы. 2013. №1(2). С.82-94.

4. Горшков С. В. Онтологическое моделирование предприятий: методы и технологии: монография. Екатеринбург: Изд-во Урал. ун-та. 2019. 236 с.

5. Рыбина Г. В., Блохин Ю. М., Иващенко М.Г. Интеллектуальная технология построения интегрированных экспертных систем // Искусственный интеллект и принятие решений. 2011. № 3. С. 48-57.

6. Gribova, V.V., KleschevA.S., MoskalenkoPh.M., Timchenko, V.A., FedorischevL.A., ShalfeevaE.A. 2017a. The IACPaaS Cloud Platform: Features and Perspectives. In Proc. of 2017 Second Russia and Pacific Conference on Computer Technology and Applications (RPC). Vladivostok. Pp. 80-84.

7. Musen M. The Protégé Project: A Look Back and a Look Forward // AI Matters. 2015 Jun; № 1(4). Pp. 4-12.

8. Sure Y., M. Erdmann, J. Angele, S. Staab, R. Studer, Wenke D. OntoEdit: Collaborative ontology development for the Semantic Web // Proc. of the Inter. Semantic Web Conference (ISWC 2002). Sardinia. Italia.

9. Verhodubs O., Grundspeņķis, J. Evolution of Ontology Potential for Generation of Rules," Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, Craiova. 2012. Article № 58. Pp. 1–4. DOI:10.1145/2254129.2254201.

УДК 004.4`2+004.89

A SET OF TOOLS TO SUPPORT DEVELOPMENT AND MAINTENANCE OF SOLVERS FOR SYSTEMS WITH ONTOLOGICAL KNOWLEDGE BASES

Valeria.V. Gribova
Doctor of Technical Sciences, Senior Scientist, Deputy Director for Scientific Work
e-mail: gribova@iacp,
Elena.A. Shalfeeva
Candidate of Technical Sciences, Associate Professor, Senior Scientist
e-mail: shalf@dvo.ru,
Institute of Automation and Control Processes
Far Eastern Branch of the Academy of Sciences,
690041, Vladivostok, Primorsky Territory, Vladivostok, st. Radio, 5

The modern approach to creating systems with knowledge bases is based on an ontological approach. A tool environment is proposed for building viable software services that work with such knowledge bases and managing their collective development. The tool support method is aimed at creating and developing libraries of ontology-based operations, using them in the design of software tools, allocating authority to create components of systems with knowledge bases, and controlling and integrating them into a cloud-based, supported knowledge-based decision support system.

Keyword: ontological knowledge base, solution algorithm, problem solver, reuse, maintainability, collaborative development.

References

1. Gavrilova T.A., Strakhovich E.V., Vizual'no-analiticheskoye myshleniye i intellekt-karty v ontologicheskoy inzhenerii [Visual-analytical thinking and mind maps in ontological engineering] // Ontologiya dizayna = Design Ontology. 2020. Vol. 10. № 1. Pp. 87-99.
2. Golenkov V.V., Gulyakina N.A. Proyekt otkrytoy semanticheskoy tekhnologii komponentnogo proyektirovaniya intellektual'nykh sistem. Chast' 2: unifitsirovannyye modeli proyektirovaniya [Project of open semantic technology for component design of intelligent systems. Part 2: unified design models] // Ontologiya dizayna = Design Ontology. 2014. №. 4 (14). Pp .34-53.
3. Golenkov V.V., Shunkevich D.V., Davydenko I.T. Semanticheskaya tekhnologiya proyektirovaniya intellektual'nykh reshateley zadach na osnove agentno-oriyentirovannogo podkhoda [Semantic technology for designing intelligent problem solvers based on an agent-based approach] // Programmnyye sistemy i vychislitel'nyye metody = Software systems and computational methods. 2013. №1 (2). Pp. 82-94.
4. Gorshkov S. V. Ontologicheskoye modelirovaniye predpriyatiy: metody i tekhnologii [Ontological modeling of enterprises: methods and technologies]. Yekaterinburg: Izd-vo Ural'skogo universiteta = Yekaterinburg: Publishing house of the Ural University. 2019. 236p.
5. Rybina G.V., Blokhin Yu.M., Ivaschenko M.G. Intellektual'naya tekhnologiya postroyeniya integrirovannykh ekspertnykh system [Intelligent technology for building integrated expert

systems] // Iskusstvennyy intellekt i prinyatiye resheniy = Artificial Intelligence and Decision Making. 2011. № 3. Pp . 48-57.

6. Gribova, V.V., KleschevA.S., MoskalenkoPh.M., Timchenko, V.A., FedorischevL.A., ShalfeevaE.A. 2017a. The IACPaaS Cloud Platform: Features and Perspectives. In Proc. of 2017 Second Russia and Pacific Conference on Computer Technology and Applications (RPC). Vladivostok. Pp.80-84.

7. Musen M. The Protégé Project: A Look Back and a Look Forward // AI Matters. 2015 Jun; № 1(4). Pp. 4-12.

8. Sure Y., M. Erdmann, J. Angele, S. Staab, R. Studer, Wenke D. OntoEdit: Collaborative ontology development for the Semantic Web // Proc. of the Inter. Semantic Web Conference (ISWC 2002). Sardinia. Italia.

9. Verhodubs O., Grundspeņķis, J. Evolution of Ontology Potential for Generation of Rules," Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics, Craiova. 2012. Article № 58. Pp. 1–4. DOI:10.1145/2254129.2254201.