Вопросы реализации проблемно-ориентированных агентов интеграции знаний

А. А. Крижановский
Санкт-Петербургский институт информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия В.О., д.39
aka@mail.iias.spb.su

УДК 681.3
А. А. Крижановский. **Вопросы реализации проблемно-ориентированных агентов интеграции знаний.** // Труды СПИИРАН, т. 1. —СПб: СПИИРАН, 2001.
Аннотация. *Настоящая статья рассматривает вопросы реализации многоагентных систем, описывает проблемно-ориентированных агентов в системе «Интеграция».* — Библ. 20 назв.

UDC 681.3
A. Krizhanovsky. **Realization questions of knowledge fusion problem-oriented agents.** // SPIIRAS Proceeding, v. 1. — SPb: SPIIRAS, 2003.
Abstract. *The paper considers realization questions of multi-agent systems; it describes the implementation of problem-oriented agents in the system "KSNet".* — Bibl. 20 items.

1. Введение

В настоящее время под многоагентной системой подразумевается множество интеллектуальных агентов, распределенных по сети, мигрирующих по ней в поисках релевантных данных, знаний и процедур и кооперирующихся в процессе выработки решений [1].

В области многоагентных систем можно выделить следующие основные направления исследований [2]:

- теория агентов, в которой рассматриваются формализмы и математические методы для описания рассуждений об агентах и для выражения желаемых свойств агентов;
- методы кооперации агентов (организации кооперативного поведения) в процессе совместного решения задач или при каких-либо других вариантах взаимодействия;
- архитектура агентов и многоагентных систем - это область исследований, в которой изучается, как построить компьютерную систему, которая удовлетворяет тем или иным свойствам, которые выражены средствами теории агентов;
- языки программирования агентов;
- методы, языки и средства коммуникации агентов.

В данной статье даётся определение агенту, кратко рассматривается классификация архитектур многоагентных систем. Рассматриваются вопросы реализации транслирующего агента и агента интеграции знаний в многоагентной системе «Интеграция» [3].

2. Определение агента и его свойства.

Что такое агент? На Токийской встрече в октябре 1996 года мнение FIPA по этому вопросу сформулировано следующим образом [4]:
«Агент - это сущность, которая находится в некоторой среде, от которой она получает данные и которые отражают события, происходящие в среде, интерпретирует их и исполняет команды, которые воздействуют на среду. Агент может содержать программные и аппаратные компоненты... Отсутствие четкого определения мира агентов и присутствие большого количества атрибутов, с ним связанных, а также существование большого разнообразия примеров агентов говорит о том, агенты это достаточно общая технология, которая аккумулирует в себе несколько различных областей».

Принято различать два определения интеллектуального агента – «слабое» и «сильное» [4]. Под интеллектуальным агентом в «слабом» смысле понимается программно или аппаратно реализованная система, которая обладает такими свойствами:

- автономность - способность ИА функционировать без вмешательства человека и при этом осуществлять самоконтроль над своими действиями и внутренним состоянием;
- общественное поведение (social ability) - способность функционировать в сообществе с другими агентами, обмениваясь с ними сообщениями с помощью некоторого общепонятного языка коммуникаций;
- реактивность (reactivity) - способность воспринимать состояние среды и своевременно отвечать (реагировать) на те изменения, которые в ней происходят;
- про-активность (pro-activity) - способность агента брать на себя инициативу, т.е. способность генерировать цели и действовать рационально для их достижения, а не только реагировать на внешние события.

«Сильное» определение агента требует наличия ряда дополнительных свойств. В частности, главным из них является наличие у агента хотя бы некоторого подмножества так называемых «ментальных свойств», называемых также интенсиональными понятиями, к которым относятся следующие:

- знания (knowledge) - это постоянная часть знаний агента о себе, среде и других агентах, т.е. та часть, которая не изменяется в процессе его функционирования;
- убеждения (beliefs, вера) - знания агента о среде, в частности, о других агентах; это те знания, которые могут изменяться во времени и становиться неверными, однако агент может не иметь об этом информации и продолжать оставаться в убеждении, что на них можно основывать свои выводы;
- желания (desires) - это состояния, ситуации, достижение которых по разным причинам является для агента желательным, однако они могут быть противоречивыми и потому агент не ожидает, что все они будут достигнуты;
- намерения (intentions) - это то, что агент или обязан сделать в силу своих обязательств по отношению к другим агентам (ему «это» поручено и он взял эту задачу на себя), или то, что вытекает из его желаний (т.е. непротиворечивое подмножество желаний, выбранное по тем или иным причинам, и которое совместимо с принятыми на себя обязательствами);
- цели (goals) - конкретное множество конечных и промежуточных состояний, достижение которые агент принял в качестве текущей стратегии поведения;
- обязательства по отношению к другим агентам (commitments) - задачи, которые агент берет на себя по просьбе (поручению) других агентов в рамках кооперативных целей или целей отдельных агентов в рамках сотрудничества.

Первые два из перечисленных понятий называют «позицией агента», его «точкой зрения» (attitudes), остальные характеризуют в англоязычной литературе общим термином «pro-attitude», суть которого в том, что они «направляют» поведение агента таким образом, чтобы сделать отвечающие данному термину содержательные и формальные утверждения истинными.

Некоторые авторы считают, что агент должен обладать также рядом других свойств. К ним относятся [4]:

- мобильность (mobility) - способность агента мигрировать по сети в поисках необходимой информации для решения своих задач, при кооперативном решении задач совместно или с помощью других агентов и т.д.,
- благожелательность (benevolence) - готовность агентов помочь друг другу и готовность агента решать именно те задачи, которые ему поручает пользователь, что предполагает отсутствие у агента конфликтующих целей;
- правдивость (veracity) - свойство агента не манипулировать информацией, про которую ему заведомо известно, что она ложна;
- рациональность (rationality) - свойство агента действовать так, чтобы достигнуть своих целей, а не избегать их достижения, по крайней мере, в рамках своих знаний и убеждений.

3. Архитектура многоагентных систем

При выборе архитектуры многоагентной системы необходимо иметь в виду два ее аспекта:

- архитектуру, поддерживающую методы взаимодействия агентов в процессе функционирования системы в целом, и
- архитектуру отдельного агента.

Основное назначение компоненты архитектуры взаимодействия системы агентов состоит в том, чтобы обеспечить скоординированное поведение агентов при решении общей и/или своих частных задач. Здесь можно выделить два основных варианта архитектур. В одном из них агенты не образуют иерархии и решают общую задачу полностью в распределенном варианте (*одноуровневая архитектура взаимодействия агентов*). В другом варианте координация распределенного функционирования агентов в той или иной мере поддерживается специально выделенным агентом, который при этом относится к мета-уровню по отношению к остальным агентам (*иерархическая архитектура взаимодействия агентов*).

Грубая классификация архитектур агентов основывается на парадигме, лежащей в основе принятой архитектуры. По этому признаку различают два основных класса архитектур [5]:

- архитектура, которая базируется на принципах и методах искусственного интеллекта, т.е. систем, основанных на знаниях («deliberative agent architecture», «архитектура разумного агента»);
- архитектура, основанная на поведении (reactive architecture) или «реактивная архитектура» (основанная на реакции системы на события внешнего мира).

На самом деле к настоящему времени среди разработанных архитектур не существует таких, о которых можно было бы определенно сказать, что она является чисто поведенческой или основана только на знаниях. Любая из разработанных архитектур является по сути гибридной, имея те или иные черты от архитектур обоих типов.

С другой стороны, независимо от лежащей в основе формализации парадигмы, архитектуры агентов классифицируются в зависимости от вида структуры, наложенной на функциональные компоненты агента, и принятых методов организации взаимодействия его компонент в процессе работы.

4. Многоагентная система управления знаниями «Интеграция» [3]

Развитие сети Интернет привело к появлению значительного количества разрозненных источников и хранилищ информации (баз данных и знаний, отдельных информационных ресурсов и т.п.), характеризующихся различными способами представления, форматами и языками описания информации. В результате возникла необходимость извлечения/приобретения, обработки и передачи/транспортировки адекватных знаний из распределенных источников своевременно в нужном контексте указанным работникам для решения актуальных задач. Данная совокупность взаимосвязанных действий была названа логистикой знаний [6]. В рамках исследовательского проекта был разработан подход «Сеть Источников Знаний» («СИЗ») к организации систем логистики знаний, основанный на технологии интеграции знаний [7]. На основе данного подхода были разработаны архитектура и прототип системы логистики знаний «Интеграция», использующие такие технологии, как управление онтологиями, интеллектуальные агенты, удовлетворение и распространение ограничений, поддержка коллективной работы и др.

Использование интеллектуальных агентов [8] было мотивировано распределённым характером задачи и тем, что источники знаний (ИЗ) в большинстве случаев являются гетерогенными и распределёнными. В качестве технологической инфраструктуры для определения свойств и функций агентов системы «Интеграция» выбрана концептуальная модель FIPA [4]. Согласно этой модели в набор агентов были включены следующие технологические агенты (рис. 1): маршрутизатор, обеспечивающий справочную службу «жёлтых страниц» для агентов, посредник, контролирующий и управляющий прохождением заданий внутри системы, интерфейсный агент, осуществляющий взаимодействие с ИЗ и агент пользователя, осуществляющий взаимодействие с пользователями системы. Дополнительно к ним были разработаны следующие проблемно-ориентированные агенты: транслирующий агент, осуществляющий перевод знания из нотации одного словаря в нотацию другого, агент управления онтологиями, осуществляющий работу с библиотекой онтологий [6], агент конфигуратор, осуществляющий конфигурацию сети ИЗ для конкретного запроса пользователя, агент интеграции знаний, осуществляющий операции по получению нового знания, агент мониторинга, осуществляющий проверку состояния сети ИЗ и агент помощник эксперта, обеспечивающий поддержку работы эксперта при добавлении нового знания.

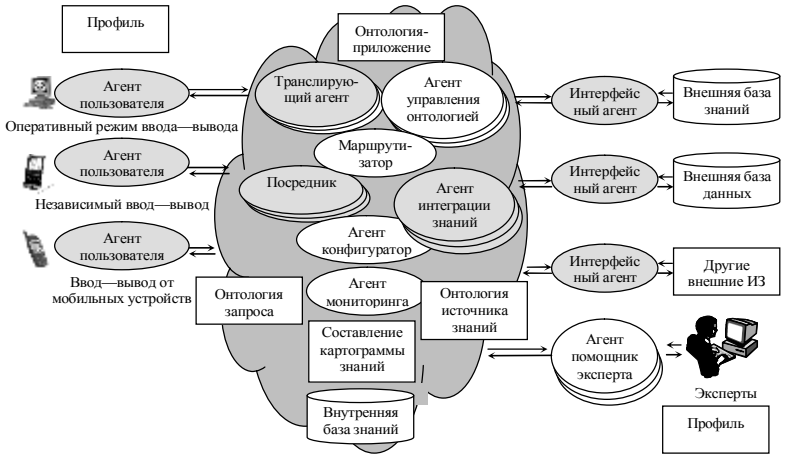

Рис. 1. Многоагентная архитектура сети источников знаний. Серым цветом обозначены агенты, в разработке которых автор работы принимал непосредственное участие

Для реализации взаимодействия агентов системы «Интеграция» был проведён анализ существующих средств разработки многоагентных систем. К ним были предъявлены следующие требования: доступность исходного кода, возможность подключения функций в виде DLL библиотек, наличие встроенных средств визуализации результатов экспериментов и

временных характеристик работы агентов, а также возможность развития данных средств, поддержка обмена сообщениями между агентами как в синхронном, так и в асинхронном режимах, возможность отправления сообщения от одного агента группе агентов, получения и анализа полученные результаты. По результатам анализа была выбрана среда MAS DK [9].

5. Реализация проблемно-ориентированных агентов в системе «Интеграция»

5.1 Транслирующий агент

Одним из необходимых и достаточно сложных в реализации проблемно-ориентированных агентов в контексте задачи управления знаниями является транслирующий агент (ТА). Этот агент переводит знания из нотации одного словаря в нотацию другого. ТА обладает всеми свойствами агента в «слабом» смысле. Про-активность здесь — это определение общей задачи, которую нужно решает пользователь, а также выявление неточностей и неопределённостей в запросе пользователя, требующие последующего диалога с пользователем для их разрешения.

Одна из задач ТА - это перевод понятий пользователя в понятия системы. Агент пользователя передает ТА запрос пользователя, заданный или на естественном языке, или с помощью шаблонов, которые облегчают процесс распознавания запроса. В полученном запросе ТА выделяет *параметрическую* и *структурную* составляющие. В результате работы ТА создаётся XML сообщение [12], которое содержит список ключевых слов и их значений (имена классов и атрибутов, хранящихся в онтологии, - структурная составляющая) и список ограничений на значения атрибутов, где значения — это параметрическая составляющая. Это XML сообщение затем передаётся агенту управления онтологиями для создания предварительной онтологии запроса, которая описывает модель интересов пользователя.

Для каждого ключевого слова указывается список синонимов, например, синоним «*лазарет*» (infirmary) для слова «*больница*» (hospital). Синонимы нужны, чтобы расширить возможности поиска онтологий при работе с библиотекой онтологий. Для поиска синонимов была использована свободно распространяемая система WordNet [13] – [15], которая применялась во многих проектах в области информационного поиска [16] – [19].

Для ключевых слов в XML сообщении (тег <Word> на рис. 2) может быть указан тип слова (тег <Type>). (Тег - это слово в треугольных скобках [12].) После разбора транслирующим агентом запроса пользователя и на его основе выполняется поиск онтологий, соответствующих запросу. Указание на этом этапе типа слова позволяет структурировать запрос пользователя и облегчить последующий поиск в библиотеке онтологий. В системе «Интеграция» онтология описывается с помощью классов объектов, классов задач и методов, атрибутов, ограничений, поэтому ТА поддерживает и распознаёт четыре типа слов: класс (Class), задача (Action), атрибут с заданным ограничением на значение (Attribute) и нераспознанное слово (Unrecognized). Тип задача (Action) указывает на класс из онтологии задач и методов. Если слово из запроса пользователя не отнесено ни к одному из указанных типов, то тип слова не указывается, то есть тег <Type> отсутствует (например, ключевое слово «mobile» на рис. 2).

При определении типа слова ТА использует несколько синтаксических правил (для предложений на английском языке), с помощью которых слово относится к одному из типов:

- **Класс**, если слово стоит после артикля («*a*» или «*an*») и до союза («*with*» или «*where*»), например, словосочетание «mobile hospital» в контексте «a mobile hospital with price».
- **Задача**, если слово стоит после приинфинитивной частицы «*to*» и до артикля «*a*» или «*an*», например, слово «*build*» в контексте «*to build a hospital*».
- **Атрибут** – это конструкция, в которой выделяют три составляющие: (i) имя атрибута (тег <Word>), (ii) тип ограничения (тег <Operator>) и (iii) задаваемое значение (тег <Value>). Тип ограничения в прототипе задаётся следующими математическими знаками: <, >, =, <=, >=.
- **Нераспознанное слово**, если слово не принадлежит ни одному из вышеприведенных типов и отсутствует в словаре WordNet.

Алгоритм обработки запроса пользователя транслирующим агентом (расширение — поиск синонимов и структуризация запроса — типизация слов):

1. Читается файл с запросом пользователя.
2. Разбор сообщения: извлечение ограничений (тип *атрибут*).
3. Инициализация WordNet.
4. Разбор сообщения: извлечение слов, относимых к типу "класс" и "задача" и тех слов, которые не относятся к какому-либо типу (извлечение только уникальных слов, если слово повторяется, то его пропускают).

5. Поиск синонимов для каждого уникального слова с помощью WordNet (перед добавлением очередного слова в результирующий набор слов проверяется отсутствие данного слова в наборе).
6. Сохранение в файл извлечённых из запроса пользователя ключевых слов.

Запрос пользователя эффективно разбирается благодаря использованию регулярных выражений языка Perl [10]. Регулярные выражения – это специальные шаблоны, которые часто используются при обработке текстов. Такие известные утилиты Unix, как: *grep*, *sed* и *awk*, а также язык программирования *perl*, которые активно используют регулярные выражения. Динамическая библиотека свободно распространяемого продукта компании ActiveState [11] обеспечивает возможность работы с регулярными выражениями при программировании на C++. На рис. 2 представлен пример XML сообщения для распознанного запроса на естественном языке: «*To build a mobile hospital with price <= 10000*».

```
<?xml version="1.0" standalone="yes"?>
<!--KSNet 2002 SPIIRAS-->
<Root_Element><!--В таких скобках пишут xml-комментарии                    -->
    <KSWord>  <!--Начало блока информации для слова «build»                -->
        <Word>build</Word><!-- Определяем, что описываем слово «build»     -->
        <Type>Action</Type>          <!--Тип слова «build»                 -->
        <Synonym>build</Synonym>
        <Synonym>construct</Synonym> <!--Синоним слова «build»             -->
        <Synonym>make</Synonym>      <!--Синоним слова «build»             -->
        <Synonym>progress</Synonym>  <!--Синоним слова «build»             -->
        <Synonym>establish</Synonym> <!--Синоним слова «build»             -->
        <Synonym>create</Synonym>    <!--Синоним слова «build»             -->
    </KSWord> <!--Конец блока информации для слова «build»                 -->
    <KSWord>  <!--Для фразы «mobile hospital» не найдено синонимов         -->
        <Word>mobile hospital</Word>
        <Type>Class</Type>           <!--Тип фразы «mobile hospital»       -->
    </KSWord>
    <KSWord>
        <Word>mobile</Word>
    </KSWord>
    <KSWord>
        <Word>hospital</Word>
        <Synonym>hospital</Synonym>
        <Synonym>infirmary</Synonym> <!--Синоним слова «hospital»          -->
    </KSWord>
    <KSWord>
        <Word>cost</Word>            <!--Имя переменной «cost»             -->
        <Type>Attribute</Type>       <!--Тип конструкции «price <= 1000»-->
        <Operator>&lt;=</Operator>   <!--Тип ограничения «<=»              -->
        <Value>10000</Value>         <!--Значение ограничения «1000»       -->
        <Synonym>cost</Synonym>      <!--Синоним слова «cost»              -->
        <Synonym>terms</Synonym>     <!--Синоним слова «cost»              -->
        <Synonym>damage</Synonym>    <!--Синоним слова «cost»              -->
        <Synonym>monetary_value</Synonym><!--Синоним слова «cost»          -->
        <Synonym>price</Synonym>     <!--Синоним слова «cost»              -->
        <Synonym>toll</Synonym>      <!--Синоним слова «cost»              -->
    </KSWord>
</Root_Element>
```

Рис. 2 Пример успешно распознанного запроса в виде XML сообщения, содержащего список распознанных слов с указанием типов и перечислением синонимов

Если запрос не был распознан или распознан частично, то для решения этой проблемы формируется группа экспертов. Эксперты проверяют нераспознанные слова из запроса пользователя, выделяют структурную и параметрическую составляющие запроса. Результаты работы экспертов передаются далее агенту управления онтологиями.

5.2 Агент интеграции знаний

Агент интеграции знаний – это ещё один необходимый для системы «Интеграция» проблемно-ориентированный агент. Этот агент осуществляет операции по получению нового

знания. Ещё одним свойством агента интеграции знаний является наличие желаний. Как было сказано, это свойство характеризует наличие состояний, ситуаций, достижение которых является для агента желательным, однако они могут быть противоречивыми. Это вызвано тем обстоятельством, что с одной стороны необходимо решить задачу, поставленную пользователем, с другой стороны есть объективные ограничения – определённая структура онтологии с заданным наполнением (классы, атрибуты, ограничения на них и отношения между ними).

В основе системы «Интеграция» лежит модель сети источников знаний, которая основывается на нотации объектно-ориентированной сети ограничений. Сеть представляется классами сущностей, относящихся к различным онтологиям; логикой атрибутов и моделью задачи удовлетворения ограничений. Данная модель объединяет основные концепции языков программирования, такие как стандартные объектно-ориентированные языки, использующие классы, и языки программирования ограничений. В качестве языка программирования ограничений был выбран ILOG Configurator [20]. В целом ILOG Configurator — это декларативный язык, доступный через C++ или Java интерфейс. Поэтому его преимущество заключается в том, что при изменении постановки задачи нет необходимости в таком серьёзном переписывании кода, как это было бы для «чистого» C++ или Java. В нашем случае задача определяется на основе той информации, которая получена из библиотеки онтологий, поэтому проблема наименее трудоёмкого и быстрого изменения кода, соответствующего новой задаче, стоит особенно остро.

Для решения указанной проблемы был предложен оригинальный механизм компиляции «на ходу» (рис. 3). Суть этого механизма в том, что (вместо создания универсального интерфейса к пакету ILOG Configurator) постановка задачи для онтологии-приложения (классы, атрибуты, ограничения) записывается непосредственно в C++ файл (рис. 3). Онтологии-приложения являются специализацией онтологий предметных областей (содержат понятия определенной области знаний) и онтологий задач (описывают определенные задачи или деятельность), при этом опираются на определения, характерные для конкретного приложения. Данные об этой онтологии берутся из библиотеки онтологий, которая входит в систему «Интеграция». После создания такого файла он копируется в заранее подготовленный проект Microsoft Visual Studio. Проект компилируется и создаётся выполнимый файл в виде динамически подключаемой библиотеки (DLL). Теперь, зная имя функции, можно вызвать её и решить задачу.

Одна из задач, которая стоит перед агентом интеграции знаний – генерация множества решений (если они есть) по некоторой онтологии-приложению и онтологии, построенной по запросу пользователя, в создании которой принимает участие транслирующий агент. Агент интеграции знаний, используя указанную методику динамической компиляции кода со встроенными командами пакета ILOG Configurator, генерирует множество решений, удовлетворяющих требованиям запроса пользователя и правилам онтологии-приложения.

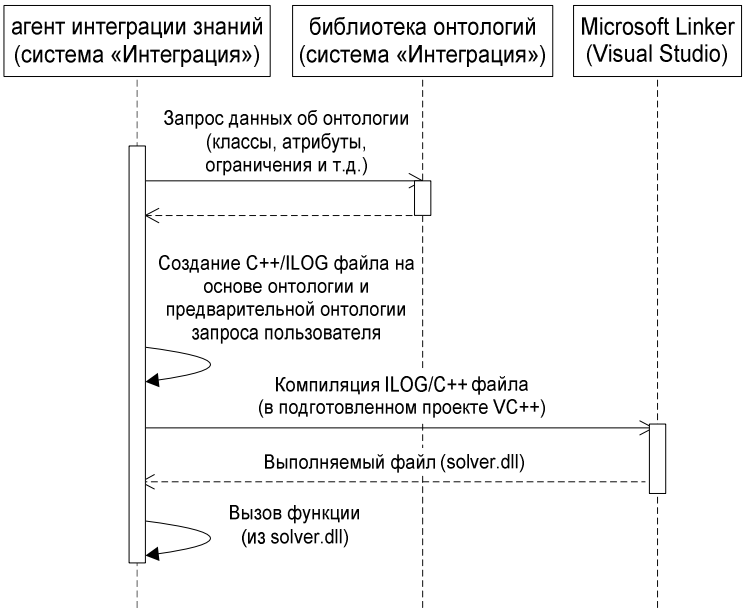

Рис. 3 UML диаграмма последовательности действий агента интеграции знаний на фазе компиляции и выполнения

Заключение

В статье описаны такие проблемно-ориентированные агенты системы «Интеграция», как: транслирующий агент и агент интеграции знаний.

Транслирующий агент разбирает запрос пользователя, выделяя в нём *параметрическую* и *структурную* составляющие. По специальным синтаксическим правилам ТА выделяет несколько типов ключевых слов, что облегчает последующий поиск подходящих онтологий в библиотеке онтологий системы «Интеграция».

На основе оригинального механизма «компиляции на ходу» и технологии удовлетворения ограничений ILOG [20] агент интеграции знаний генерирует множество решений или сообщает об их отсутствии, при этом наличие базы знаний даёт возможность повторного использования найденных раньше решений.

Литература

[1] Agent projects. Europe's Network of Exellence for Agent-based Computing. http://www.agentlink.org/resources/-agentprojects-db.html, 2001.

[2] В.Городецкий, М.Грушинский, А.Хабалов. Многоагентные системы (обзор). "Новости искусственного интеллекта", №1, 1997.

[3] Smirnov A., Pashkin M., Chilov N., Levashova T. Multi-agent Architecture for Knowledge Fusion from Distributed Sources. Lecture Notes in Artificial Intelligence. Springer, 2002. 2296, pp. 293—302.

[4] FIPA 2000 Specifications. Geneva, Switzerland, Foundation for Intelligent Physical Agents, 2002. http://www.fipa.org/repository/fipa2000.html

[5] Agent projects. Europe's Network of Exellence for Agent-based Computing. http://www.agentlink.org/resources/-agentprojects-db.html, 2001.

[6] Левашова Т.В., Пашкин М.П., Смирнов А.В., Шилов Н.Г. Принципы построения систем для быстрой интеграции знаний из распределенных источников // Труды международного конгресса «Искусственный интеллект в XXI веке». — Дивноморское, Россия, 2001. Т. 1. — С. 105–119.

[7] Смирнов А.В., Пашкин М.П., Шилов Н.Г., Левашова Т.В. Онтологии в системах искусственного интеллекта: способы построения и организации. Новости искусственного интеллекта, 2002. № 1. Часть 1. С. 3–13. № 2. Часть 2. С. 3–9.

[8] Wooldridge M.J., Jennings N.R. Agent Theories, Architecture, and Languages: A survey. In: Intelligent Agents: Proceedings of the Workshop on Agents Theories, Architecture, and languages (ECAI-94), Springer-Verlag, 1995. pp. 1–39.

[9] Gorodetski V., Karsayev O., Kotenko I., Khabalov A. Software Development Kit for Multi-agent Systems Design and Implementation. Proceedings of the Second International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'01). (B.Dunin-Keplicz, E.Nawarecki (eds.)). Krakow, Poland, 26-29 September, 2001. pp. 99–108.

[10] ActivePerl. The industry-standard Perl distribution for Linux, Solaris, and Windows, ActiveState, 2002. http://www.activestate.com/Products/ActivePerl

[11] ActiveState Homepage, ActiveState, 2002. http://www.activestate.com

[12] Extensible Markup Language (XML). W3C Architecture Domain, W3C, 2002. http://www.w3.org/XML.

[13] Wordnet http://www.cogsci.princeton.edu/~wn, 2002.

[14] Miller G.A., Beckwith R., Fellbaum C., Gross D., and Miller K.J. Introduction to WordNet: an on-line lexical database. International Journal of Lexicography, 1990, 3(4), pp. 235–244. ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps.

[15] Miller A.G. Wordnet: A lexical database for English. Communications of the ACM, 38 (11), November 1995, pp. 39–41. http://www.cogsci.princeton.edu/~wn.

[16] Gangemi A., Guarino N., Masolo C., Oltramari A. Restructuring WordNet's Top-Level. 2002.

[17] Lytinen, S., Tomuro, N. and Repede, T. (2000). "The Use of WordNet Sense Tagging in FAQFinder". In Proceedings of the workshop on Artificial Intelligence for Web Search at the 17th National Conference on Artificial Intelligence (AAAI-2000), Austin.

[18] Tomuro, N. (1998). "Semi-automatic Induction of Systematic Polysemy from WordNet". In Proceedings of the workshop on Usage of WordNet in Natural Language Processing Systems at the 17th International Conference on Computational Linguistics (COLING-98), Montreal, Canada, pp. 108-114.

[19] WonderWeb - Ontology Infrastructure for the Semantic Web http://wonderweb.semanticweb.org.

[20] ILOG Configurator. Whitepaper. 2001. Интернет адрес http://www.ilog.com/products/configurator/wp_configurator.pdf.