УДК 681.3

Анализ инструментальных средств построения агентно-ориентированных систем

Жабская Т.Е., Федяев О.И.
Донецкий национальный технический университет
fedyaev@r5.dgtu.donetsk.ua

Abstract

Zhabskaya T, Fedyaev O., Analysis of toolkits to create agent-oriented systems. *The submitted article considers well-known toolkits to create agent-oriented systems and deals with a problem of selection effective toolkits for implementing MAS tutorial system, which will provide autonomy, individuality and state of distribution carrying out educational-methodological duties of all the education process participants at the department level.*

Введение

Вступление университетов в эпоху глобализации образовательной среды, динамичности рынка образовательных услуг и ужесточение конкуренции вузов способствовало возникновению разнообразных виртуальных форм знания и учебных подразделений, обеспечивающих гибкость образовательной организации с настраиваемой на среду структурой, состоящей из автономных искусственных агентов, сотрудничающих для достижения поставленной общей цели. Такая виртуальная компьютеризированная система обучения обладает большими возможностями направленной эволюции и самоорганизации в интересах адаптации к быстро меняющейся среде постиндустриального общества [1].

Современный уровень развития сетевых информационных систем, информатики и искусственного интеллекта позволяет создать принципиально новую компьютеризированную среду адекватную как для группового, так и индивидуально-дистанционного обучения студентов университета [2].

Основными недостатками традиционных подходов к автоматизации процесса обучения является слабая интеграция различных форм обучения и учебно-методических ресурсов, ориентация на изучение только отдельных дисциплин, а не на комплексную подготовку специалистов по специальности или циклу дисциплин, недостаточно развитые средства дистанционного образования, стирающие границы между очным и заочным образованием, отсутствие распределённого интеллектуального моделирования функций всех субъектов учебного процесса как единой целостной системы обучения. Даже новые экспертно-обучающие системы не поддерживают в полной мере взаимодействие разнотипных источников знаний, которые должны динамически обмениваться запрашиваемыми знаниями в процессе обучения.

В данной статье решается задача выбора эффективных инструментальных средств для информатизации и интеллектуализации деятельности выпускающей кафедры, которая является базовой структурной единицей технического университета. При этом кафедра рассматривается как территориально распределённый объект, деятельность которого направлена на комплексную подготовку специалистов по конкретной специальности или циклу дисциплин. Наиболее адекватным средством описания функционирования виртуальной организации, моделирующей деятельность реальной кафедры, является агентно-ориентированный подход [3]. Для построения агентно-ориентированных систем предложено достаточно много методологий [4-11] и соответствующих инструментальных средств, предназначенных для автоматизации их программной реализации [12-17]. Поэтому выбор наилучшей методологии и инструментария с учётом специфики объекта автоматизации является актуальной задачей.

Создание агентно-ориентированной системы связано с решением следующих задач:

1. Обеспечение взаимодействия между агентами с целью использования ресурсов агентного сообщества для решения сложной задачи.
2. Наполнение агентов знаниями для проявления необходимых ментальных свойств (разработка онтологии для определения понятий предметной области приложения, которыми оперируют агенты в процессе общения, механизма рассуждений агента о применении своих собственных способностей и способностей других агентов, языка коммуникации для общения агентов друг с другом, механизма поиска информации о существовании других агентов с заданными способностями, их сетевых адресов).

3. Координация совместной деятельности агентов для достижения поставленной перед многоагентной системой цели.
4. Программная реализация агентов на основе знаний.

Для решения вышеперечисленных задач наиболее известными проектами из многочисленного ряда инструментальных систем являются AgentBuilder [12], Bee-gent [13], JACK [14], JADE [15], FIPA-OS [16], Zeus [17].

Инструментарий AgentBuilder

Инструментарий AgentBuilder предоставляет для разработчиков средства разработки и среду выполнения агентного приложения. Технология создания интеллектуального агента в инструментарии AgentBuilder представлен на рис.1.

Средства разработки и среда выполнения написаны на языке программирования Java, что позволяет им работать на всех платформах, где установлена Java среда. Агент, созданный с помощью инструментария AgentBuilder, может выполняться на любой платформе с виртуальной машиной Java (версии 1.1 и выше).

Средства разработки представляют собой удобный графический интерфейс для анализа предметной области разрабатываемой МАС и спецификации желаемого поведения агентов, разрабатываемых при помощи графических редакторов. В данной инструментальной среде предусмотрены следующие этапы построения многоагентного приложения:

– определение состава агентства;
– создание агентов, которое предусматривает построение онтологии, используемой для выполнения делегированных агенту полномочий, и ментальной модели (убеждения, способности, обязательства, правила поведения);
– создание протоколов для спецификации взаимодействия агентов данного агентства;
– генерация специального файла описания агента на языке RADL, который, в конечном итоге, представляет ментальную модель и желаемое поведение агента.

Среда выполнения агентного приложения состоит из агентной программы и процессора выполнения агента. Процессор использует эффективные процедуры логического вывода путём сопоставления правил поведения агента с убеждениями агента, определяемыми текущей ментальной моделью, и входящими сообщениями. На основе проводимых рассуждений процессор выполняет определенные действия, связанные с полномочиями агента. Агентная программа представляет собой определение агента в виде файла на языке RADL вместе с укомплектованной библиотекой классов проекта. Агентная программа совместно с процессором образуют выполняемого агента.

При запуске среды выполнения, инициализируется процессор агента, который использует RADL-модель и онтологию агента, представленную в виде библиотеки классов проекта (Project Accessories Library). Для этого необходимы определение агента (файл RADL, который обеспечивает агента способностью рассуждения и начальной ментальной моделью) и библиотека классов проекта (вспомогательные классы проекта PACs из библиотеки классов проекта) – эти объекты используются для отображения предметной области задачи.

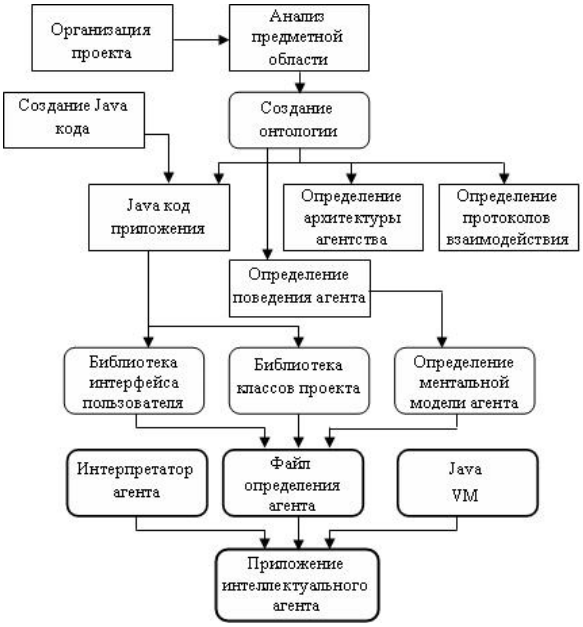

Рисунок 1. Процесс создания интеллектуального агента в инструментарии AgentBuilder

Система Bee-gent

В системе Bee-gent разработка агентно-ориентированных приложений выполняется по методологии спецификации поведения агентов распределенной системы с использованием MAC - библиотеки, реализованной на языке Java.

На основе предлагаемых системой Bee-gent графических средств, возможна чёткая структуризация поведения каждого агента в виде графа состояний и определение протоколов взаимодействий агентов. Графы состояний агентов строятся на основании жизнеспособности ролей, определенных в виде регулярных выражений на этапе агентно-ориентированного анализа (например, по методологии Gaia [4]). Пример фрагмента графа поведения агента Студент обучающей системы показан на рис. 2.

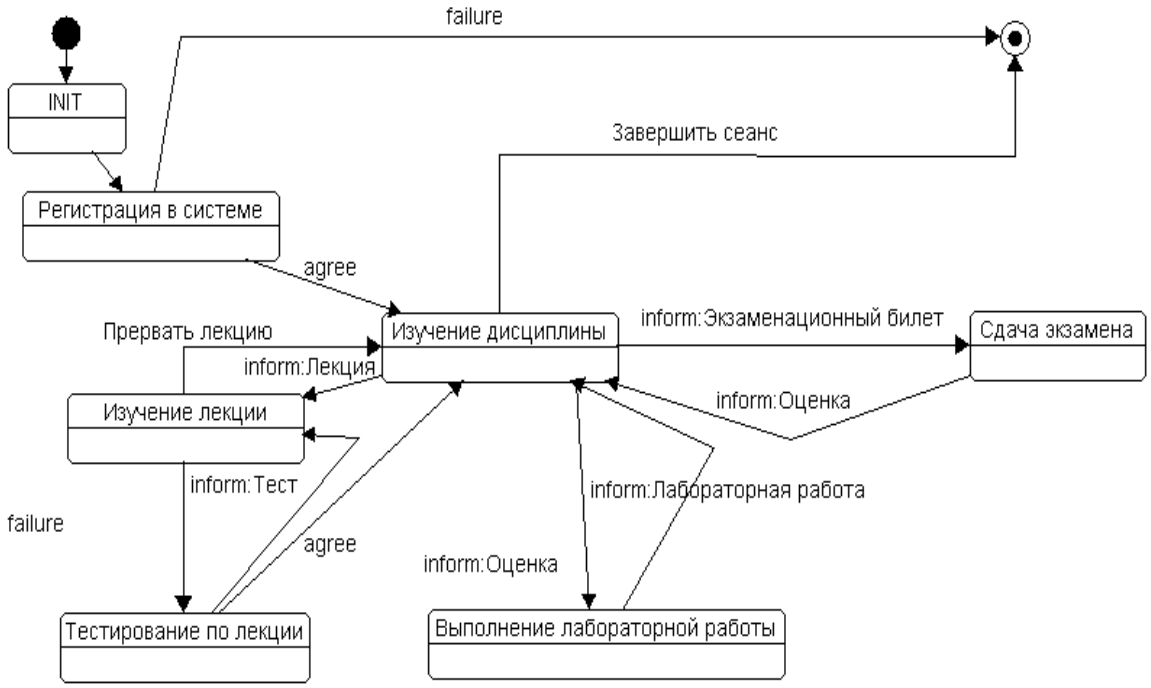

Рисунок 2. Граф состояний поведения агента Студент

Граф состояний регистрирует все имена состояний, в которых агент может находиться. На следующем шаге разработки определяются классы для каждого состояния. Каждое состояние в графе является экземпляром класса AwrIPState из агентной библиотеки фирмы Toshiba, реализованной на языке Java. В конструкторе класса определяются пред и пост условия, т.е. условия, которые должны быть выполнены агентом в текущем состоянии для того, чтобы выполнить действия, определенные классом состояния, и определить переход в следующее состояние. Затем специфицируются действия, которые должны быть выполнены в каждом состоянии (включая собственные процессы агента и взаимодействия с другими агентами). Для начального и конечного состояний также создаются классы "INIT" и "END".

Если агент взаимодействует с другими агентами, то при спецификации отдельных состояний система Bee-gent предусматривает определение протокола взаимодействия. Протокол должен отражать все линии поведения агента в данном состоянии. В каждом состоянии деятельность агента направлена на выполнение протоколов взаимодействия с целью реализации планируемой линии поведения. Деятельность каждого агента в MAC определяется, например, моделью услуг, разработанной на этапе агентно-ориентированного анализа по методологии Gaia.

Каждая линия поведения документируется диаграммой взаимодействия агентов с указанием содержимого сообщений и их очередности. На рис.3 приведен пример диаграммы взаимодействия для состояния “Изучение дисциплины” агента Студент. Формат сообщений определяется языком XML/ACL, который является развитием языка коммуникации KQML.

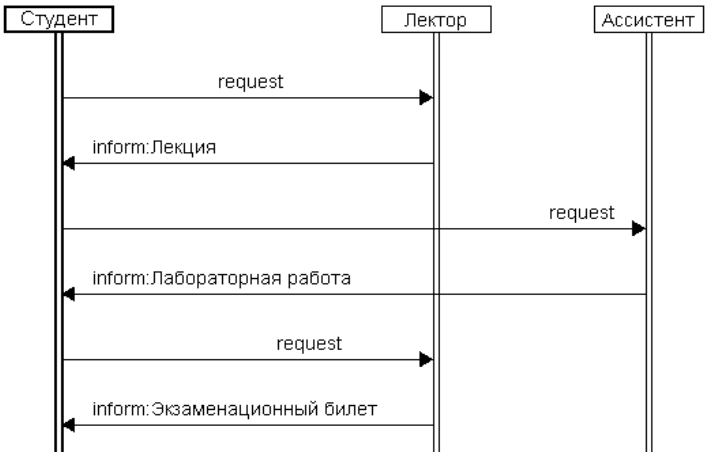

Рисунок 3. Диаграмма взаимодействия агента Студент в состоянии “Изучение дисциплины”

Таким образом, на основе разработанных логических моделей, система Bee-gent автоматически генерирует на языке Java скелет программного кода многоагентной системы, который дополнятся необходимым программным кодом, обеспечивающим заданный “жизненный цикл” агентов.

В системе Bee-gent, в отличие от AgentBuilder, при описании поведения агентов не используются правила, определяющие реакцию агента на внешние события и его внутреннее состояние.

Среда разработки JACK

Агентно-ориентированная среда разработки JACK™ Intelligent Agents (JACK) построена на основе языка программирования Java. JACK является надстройкой Java в виде расширения синтаксиса Java конструкциями для программной реализации свойств, связанных с понятием интеллектуального агента. Язык программирования агентов JACK предлагает следующие возможности:

- определяет новые основные классы, интерфейсы и методы;
- расширяет синтаксис Java для поддержки новых агентно-ориентированных классов, определений и операторов;
- предоставляет расширения семантики (особенности при выполнении) для поддержки модели выполнения, требуемой агентно-ориентированной программной системой.

Все расширения языка реализованы как plug-in, что делает язык максимально расширяемым и гибким в агентно-ориентированном программировании.

На уровне классов введены 5 главных конструкций:

- агент, который в JACK моделирует интеллектуальные сущности;
- способность, которая собирает в одно целое функциональные компоненты (события, планы, множество убеждений и др. способности), для использования их агентами;
- событие, для моделирования ситуаций и сообщений, на которые агент должен быть способен ответить;
- план, который предназначен для моделирования процедурного описания того, как агент управляет данным событием (все предпринимаемые агентом действия заранее предусмотрены и описаны в его планах);
- множество убеждений, для моделирования знаний агента в виде убеждений, которые придерживаются семантики закрытого или открытого мира. Данная конструкция представляет убеждения агента в виде реляционных кортежей первого порядка и обеспечивает их логическую непротиворечивость.

Следует отметить, что желаемое поведение агента инкапсулируется в модульных единицах, определяемых этими классами, а классы содержат все требуемые для независимого выполнения структуры и методы, которые программисты на языке JACK могут использовать.

Для установления отношений между упомянутыми выше классами предоставлен набор деклараций. Ниже приведен фрагмент кода для реализации конструкции плана, написанного на JACK (элементы синтаксиса, которые принадлежат JACK, выделены жирным шрифтом):

```
plan MovementResponse extends Plan {
#handles event RobotMoveEvent moveresponse;
#uses agent implementing RobotInterface
robot;
static boolean relevant (RobotMoveEvent ev)
{ … }
context()
{ … }
#reasoning method
body(){ … }}.
```

В этом примере определяемый план действий программного агента наследует свои основные выполняемые функции от класса JACK-Plan. Кроме того, с помощью нескольких деклараций для планов языка JACK указывается, каким образом план будет использоваться. Каждая декларации предваряется символом # для того, чтобы отличить их от элементов синтаксиса Java.

Декларация `#handles event` определяет цель или событие, на которое этот план отвечает. Декларация `#uses agent implementing` закрепляет агента(ов), которые могут использовать этот план. План в примере могут выполнять только те агенты, которые реализуют указанный интерфейс (`RobotInterface`). В фигурных скобках содержится обычный код Java.

Помимо деклараций язык JACK для описания рассуждений и поведения, предпринимаемых агентом при выполнении плана, предоставляет свои операторы методов рассуждения, которые выделяются предшествующим символом @.

Для поддержки выполнения агентно-ориентированной программной системы JACK предоставляет следующие дополнительные языковые расширения, обеспечивающие следующую семантику:

- Многопоточность встроена в ядро и выведена из-под контроля программиста.
- Работа агентов осуществляется таким образом, что агенты обрабатывают множество планов и имеют доступ к описаниям убеждений. Агенты выполняют планы в задачах управления событиями, когда они возникают, сравнивая свои убеждения, когда необходимо. Эти планы могут инициировать подзадачи, которые в свою очередь могут инициировать свои подзадачи, если агент требует трудоемкий и сложный ответ.
- Введена новая структура данных, названная логический элемент (logical member), значение которого зависит от результата запроса к множеству убеждений агента.
- Возможность выполнение запросов к множеству убеждений агента, используя для этого логические элементы, посредством их объединения для получения желаемого результата. Если запрос имеет успех, то логический элемент содержит желаемое значение.

Компонента среды разработки JACK (JACK Development Environment) дает возможность рисования обзорных диаграмм, по которым среда генерирует скелет программного кода и следит за тем, чтобы изменения, произведенные в коде, отображались и на диаграммах.

Агенты, создаваемые в JACK, имеют архитектуру, присущую интеллектуальным агентам. Таким образом, возможно моделирование разумного поведения, в соответствии с теоретической моделью BDI-архитектуры агента [9], основанной на убеждениях, желаниях и намерениях.

Согласно BDI-архитектуре, интеллектуальные агенты JACK – это автономные программные компоненты, которые могут проявлять разумное поведение на основе проактивности (целенаправленность) и реактивности (направляемое событиями) на входные сигналы. Каждый такой агент имеет:

- убеждения (это его набор данных о мире);
- желания (набор событий на которые он будет реагировать и набор целей, достижения которых он может желать);
- намерения (набор планов, которые описывают как он может управлять возникающими целями и планами).

Если агента рассматривать как аналог личности, то набор планов описывает шаги, которые агент должен выполнить при возникновении определенного события или желании достичь определенного результата. На первый взгляд, поведение агента может показаться похожим на действия экспертных систем, со всеми присущими им ограничениями. Однако, принципиальное отличие агентно-ориентированных систем в том, что агенты можно программировать для выполнения планов точно так же, как действовала бы разумная личность. В частности, с помощью агентов можно реализовать следующие свойства, ассоциирующиеся с разумным поведением:

- *устойчивую целенаправленность* – агенты сосредоточены на целях, а не на выбранных методах для их достижения;
- *контекстную зависимость в реальном времени* – агенты будут следить за вариантами, которые применимы в каждый момент времени и принимать решения относительно последующих действий, на основе имеющихся условий;
- *утверждение правильности подхода в реальном времени* – агент будет гарантировать, что он следует выбранному курсу действий до тех пор, пока определенные условия продолжают быть истинными;
- *одновременность* – агентная система является многопоточной. Если возникают новые цели и события, то агент способен определить приоритеты по требованию многозадачности.

JACK приложение представляет собой исходный код, реализующий характерные для агентно-ориентированного подхода понятия: агентов, способностей, события, планы, убеждения, view (запросы), а также Java класс с функцией `main()`, которая является точкой входа

для виртуальной машины Java, и любые другие Java необходимые файлы.

Файлы, которые создаются для этих понятий, должны иметь такое же имя, как и у объекта, определяемого в файле. Они имеют расширение, определяющее тип JACK понятия.

Компилятор агентов JACK конвертирует исходные файлы на языке агентов JACK в код на языке Java, который затем компилируется в код виртуальной машины Java для выполнения на целевой системе

Программная среда JADE

Программная среда JADE (Java Agent Development Framework) получила широкое применение для разработки многоагентных систем. Она полностью реализованная на языке Java и поддерживает FIPA – стандарты для создания интеллектуальных агентов. Цель создания среды JADE – упростить процесс разработки посредством стандартизации способов взаимодействия агентов во всеоохватывающей среде системных сервисов. Для достижения этой цели JADE предлагает програмисту-разработчику агентных систем следующие возможности:

- агентную платформу *FIPA-compliant Agent Platform*, основанную на FIPA и включающую обязательные типы системных агентов для управления, во-первых, агентной платформой (AMS), во- вторых, каналом коммуникации (ACC) и службы каталогов (DF) (эти типы агентов автоматически активируются при запуске платформы);
- распределенную агентную платформу *Distributed Agent Platform*, которая может использовать несколько хостов, при чем на каждом узле запускается только одна Java Virtual Machine. Агенты выполняются как Java-потоки. В зависимости от местонахождения агента, посылающего сообщение, и того, кто его получает, для доставки сообщений используется соответствующий транспортный механизм.
- *Multiple Domains support* – ряд основанных на FIPA DF-агентов могут объединится в федерацию, таким образом реализуя мультидоменную агентную среду.
- *Multithreaded execution environment with two-level scheduling*. Каждый JADE-агент имеет собственный поток управления, но он также способен работать в многопотоковом режиме. Java Virtual Machine проводит планирование задач, исполняемых агентами или одним из них.
- *Object-oriented programming environment*. Большинство концепций, свойственных FIPA-спецификации, представляются Java-классами, формирующими интерфейс пользователя.
- *Library of interaction protocols*. Используются стандартные интерактивные протоколы fipa-request и fipa-contract-net. Для того, чтобы создать агента, который мог бы действовать согласно таким протоколам, разработчикам прикладных программ нужно только имплементировать специфические доменные действия, в то время как вся независимая от прикладной программы протокольная логика будет осуществляться системой JADE.
- *Administration GUI*. Простые операции управления платформой могут исполняться через графический интерфейс, отображающий активных агентов и контейнеры агентов. Используя GUI, администраторы платформы могут создавать, уничтожать, прерывать и возобновлять действия агентов, создавать иерархии доменов и мультиагентные федерации DF (фасилитаторов).

JADE базируется на технологиях Java RMI, Java CORBA IDL, Java Serialization и Java Reflection API. Разработка МАС в этой среде упрощается благодаря использованию FIPA-спецификаций и ряда инструментов поддержки фазы отладки и развертывания системы. Эта агентная платформа может устанавливаться на компьютерах с разными операционными системами, и ее можно конфигурировать через удаленный GUI-интерфейс. Процесс конфигурирования этой платформы достаточно гибкий: ее можно изменить даже во время исполнения программ, посредством перемещения агентов с одной машины на другую. Единственным требованием для работы системы является установка на машине Java Run Time 1.2.

Каждый запущенный экземпляр среды JADE является контейнером, т.к. может содержать несколько агентов. Группа активных контейнеров образуют платформу. Главный контейнер всегда должен быть активен, а все другие контейнеры должны быть зарегистрированы им при их создании. Поэтому, первый контейнер, запущенный на платформе является основным контейнером, а все остальные - обычными контейнерами и должны получить указания о том, где находится их основной контейнер, на котором они должны быть зарегистрированы.

Если в сети запускается еще один основной контейнер, то он представляет собой другую платформу, на которой новые обычные контейнеры имеют возможность зарегистрироваться. На рис. 4 показаны приведенные выше концепции платформы и контейнера и показывает сценарий с двумя JADE платформами, состоящими из трёх и одного контейнера соответственно. JADE агенты должны иметь уникальные имена, знать имена друг друга и, благодаря этому, они могут общаться напрямую, независимо от их фактического местонахождения, т.е. внутри одного контейнера (например, агенты A2 и A3), в различных

контейнерах внутри одной платформы (например, A1 и A2) или в различных платформах (например, A4 и A5).

Основной контейнер отличается от обычных тем, что содержит систему управления агентами и маршрутизатор, которые автоматически запускаются при запуске основного контейнера.

Система управления агентами AMS (Agent Management System), представляет собой «власть» в платформе (создание / удаление агентов в удаленных контейнерах, запрашиваемых через AMS) и обеспечивает службу именования агентов.

Маршрутизатор каталогов DF (Directory facilitator), который обеспечивает сервис «Жёлтых страниц», помогает найти агенту других агентов, для получения от них необходимых услуг, необходимых ему для достижения своих целей.

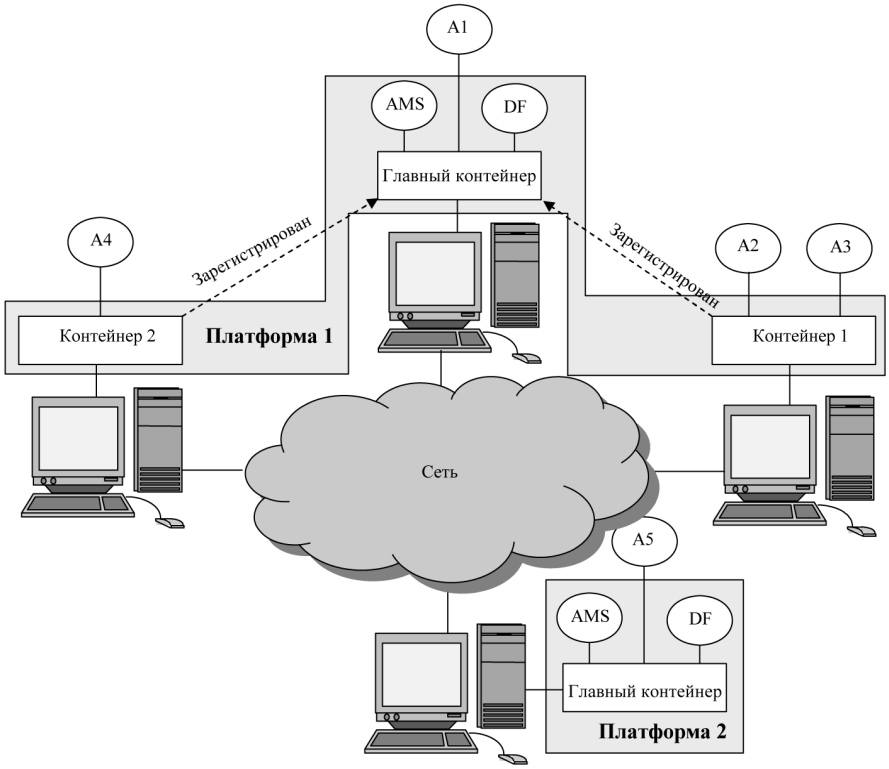

Рисунок 4. Среда «существования» агентов JADE.

Для осуществления коммуникации архитектура среды предоставляет гибкий и эффективный процесс обмена сообщениями, в котором JADE создает очередь и управляет потоком ACL-сообщений, являющихся приватными для каждого агента. Агенты способны обращаться к очереди с помощью комбинации нескольких режимов своей работы: блокирование, голосование, перерыв в работе и сопоставление с эталоном (если это касается методов поиска). В последних версиях системы используется Java RMI, event-notification и IIOP. Однако, можно легко добавить и другие протоколы. Также предусмотрена возможность интеграции SMTP, HTTP и WAP. Большинство коммуникационных протоколов, которые уже определены международным сообществом разработчиков агентных сред, доступны и могут иллюстрироваться на конкретных примерах после определения поведения системы и ее основных состояний. Вместе с поддержкой определенных пользователем контентных языков, реализованы онтологии управления агентами, а также онтологии, которые могут быть реализованы и зарегистрированы агентами и использованы системой. С целью существенного расширения работоспособности JADE, предусмотрена возможность интеграции с JESS и Java-оболочкой CLIPS. JADE используется рядом компаний и академических групп. Среди которых можно выделить такие известные как: BT, CNET, NHK, Imperial College, IRST, KPN, University of Helsinky, INRIA, ATOS.

Заключение

Сравнительный анализ возможностей, предоставляемых различными инструментальными средствами для разработки агентно-ориентированных систем, приводится в табл.1. На основании сравнения характеристик рассмотренных инструментальных систем можно сделать вывод о том, что наиболее мощными и гибкими технологиями реализации понятия

"агент" в разрабатываемой многоагентной обучающей системе по дисциплинам кафедры, являются подходы, предложенные инструментарием AgentBuilder и агентно-ориентированной средой JACK.

Достоинство данных инструментариев заключается в использовании последних достижений для создания архитектуры интеллектуальных агентов [9,18]. Очень важно, что оба инструментария обеспечивают моделирование ментальных свойств агентов, необходимых для решения задачи обучения, поскольку построение системы, обладающей заданными качествами, напрямую зависит от того, насколько удачно реализовано понятие "агент" и связанные с ним свойства.

Следует отметить наличие в этих системах механизма рассуждений, необходимых для воспроизведения программными агентами делегированной им интеллектуальной деятельности.

Заслуживает внимания простота изменения характера деятельности агентов. Это обеспечивается тем, что направление деятельности интеллектуальных агентов определяется на основе логического вывода. Для изменения характера функционирования МАС в целом достаточно изменить знания каждого агента о своей деятельности. В инструментарии AgentBuilder это достигается путем изменения правил поведения, а в среде JACK изменением логики рассуждений, каждого из автономных агентов в составе МАС.

От успешной реализации искусственного агента зависит возможность построения системы, обладающей новыми качествами, которые нельзя получить иначе как с помощью агентно-ориентированного подхода.

Проведенный анализ наиболее известных инструментальных систем позволил выбрать эффективный и доступный язык JACK для программной реализации обучающей системы

На основе созданной компьютерной системы обучения агентно-ориентированного типа нового поколения будет проведен анализ адекватности ментальных моделей участникам учебного процесса и получена оценка эффективности агентной архитектуры в целом.

Таблица 1. Сравнительный анализ инструментариев для разработки агентно-ориентированных приложений

Возможности инструментальных систем	*Agent Builder*	*Bee-gent*	*JACK*	*JADE*
Средства построения агентств	да	нет	нет	нет
Средства управления проектом	да	нет	да	нет
Графическая среда для определения спецификаций агентов	да	нет	да	нет
Механизм контроля целостности	да	нет	да	нет
Средства построения онтологии	да	нет	нет	нет
Библиотека для разработки МАС	да	да	да	да
Механизм рассуждений агента о своих способностях и способностях других агентов	да	нет	да	нет
Формальный язык коммуникации	да	да	да	да
Средства отладки взаимодействия агентов	да	нет	да	да
Механизм поиска агентов с заданными способностями	нет	нет	нет	да
Экспертная оценка возможностей инструментальных средств, %	90%	20%	70%	40%

Литература:

1. Емельянов В. В., Тарасов В. Б. Виртуальная кафедра в техническом университете// Дистанционное образование. – 2000. – № 6. – С.39–45.
2. Федяев О.И., Жабская Т.Е., Грач Е.Г. Многоагентная модель процесса обучения студентов на кафедральном уровне //Наукові праці Донецького національного технічного університету. Серія "Проблеми моделювання та автоматизації проектування динамічних систем". Випуск 5(116). – Донецк: ДонНТУ, 2006.- с. 105 - 116.
3. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика.- М.: "Едиториал УРСС", 2002. -352 с.
4. F.Zambonelli, N.R. Jennings, and M.Wooldridge. Developing Multiagent Systems: The Gaia Methodology. In ACM Transactions on Software Engineering Methodology, 12(3):317-370, July 2003.
5. Giunchiglia F., Mylopoulos J., Perini A. The Tropos software development methodology:

Processes, Models and Diagrams // Third International Workshop on Agent-Oriented Software Engineering, 2002.
6. L. Padgham and M. Winikoff. Prometheus: A Methodology for Developing Intelligent Agents. In F. Giunchiglia, J. Odell, and G. Weiss, editors, Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002.
7. A. Collinot, A. Drogoul, P. Benhamou, Agent oriented design of a soccer robot team, Proceedings ICMAS-96, (1996) 41-47.
8. M. Elammari and W. Lalonde. An agent-oriented methodology: High-level and intermediate models. In G. Wagner and E. Yu, editors, Proc. of the 1st Int. Workshop. on Agent-Oriented Information Systems, 1999.
9. David Kinny, Michael Georgeff, Anand Rao A Methodology and Modelling Technique for Systems of BDI Agents. In W. van der Velde and J. Perram, editors, *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent MAAMAW'96, (LNAI Volume 1038)*. Springer-Verlag: Heidelberg, Germany, 1996.
10. Carlos A. Iglesias, Mercedes Garijo, Jos´e C. Gonz´alez, and Juan R. Velasco. Analysis and design of multiagent systems using MAS-CommonKADS. In AAAI'97Workshop on Agent Theories, Architectures and Languages, Providence, RI, July 1997. ATAL. (An extended version of this paper has been published in *INTELLIGENT AGENTS IV: Agent Theories, Architectures, and Languages*, Springer Verlag, 1998.
11. Amund Tveit A survey of Agent-Oriented Software Engineering. NTNU Computer Science Graduate Student Conference. Norwegian University of Science and Technology, May (2001)
12. AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents [Electronic Resourse] / Reticular Systems, 1999. – Mode of access: http://www.agentbuilder.com.
13. Bee-gent, 1999. Bee-gent Home Page. – Mode of access: http://www2.toshiba.co.jp/beegent.
14. JACK Intelligent Agents. – Mode of access: http://www.agent-software.com/products/jack/index.html
15. JADE. – Mode of access: http://jade.tilab.com/
16. FIPA-OS. – Mode of access: http://sourceforge.net/projects/fipa-os/
17. The ZEUS Agent Building Toolkit. – Mode of access: http://sourceforge.net/projects/zeusagent
18. Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence, 60(1), 51 - 92.

Поступила в редколлегию 04.03.2009