

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. Координація спільної діяльності агентів для досягнення поставленої перед багатоагентною системою цілі.
4. Програмна реалізація агентів на основі знань.

Для рішення вищеперечислених задач найбільш відомими проектами з численного ряду інструментальних систем є AgentBuilder [12], Bee-gent [13], JACK [14], JADE [15], FIPA-OS [16], Zeus [17].

Інструментарій AgentBuilder

Інструментарій AgentBuilder надає для розробників засоби розробки і середовище виконання агентного застосування. Технологія створення інтелектуального агента в інструментарії AgentBuilder зображено на рис. 1.

Засоби розробки і середовище виконання написані на мові програмування Java, що дозволяє їм працювати на всіх платформах, де встановлено Java середовище. Агент, створений за допомогою інструментарія AgentBuilder, може виконуватися на будь-якій платформі з віртуальною машиною Java (версії 1.1 і вище).

Засоби розробки представляють собою зручний графічний інтерфейс для аналізу предметної області розроблюваної МАС і специфікації бажаного поведінки агентів, розроблюваних за допомогою графічних редакторів. В даній інструментальній середі передбачені наступні етапи побудови багатоагентного застосування:

- визначення складу агентства;
- створення агентів, яке передбачає побудову онтології, використовуваної для виконання делегованих агенту

повноважень, і ментальної моделі (убезпечення, здатності, обов'язки, правила поведінки);

- створення протоколів для специфікації взаємодії агентів даного агентства;
- генерація спеціального файлу описання агента на мові RADL, який, в кінцевому підсумку, представляє ментальну модель і бажане поведінку агента.

Середовище виконання агентного застосування складається з агентної програми і процесора виконання агента. Процесор використовує ефективні процедури логічного висновку шляхом співставлення правил поведінки агента з убезпеченнями агента, визначуваними поточною ментальною моделлю, і входять повідомленнями. На основі виконуваних висновків процесор виконує певні дії, пов'язані з повноваженнями агента. Агентна програма представляє собою визначення агента в вигляді файлу на мові RADL разом з упакованою бібліотекою класів проекту. Агентна програма разом з процесором утворюють виконуваний агент.

При запуску середовища виконання, ініціалізується процесор агента, який використовує RADL-модель і онтологію агента, представлену в вигляді бібліотеки класів проекту (Project Accessories Library). Для цього необхідно визначення агента (файл RADL, який забезпечує агента здатністю висновку і початковою ментальною моделлю) і бібліотеку класів проекту (вспомогателі класи проекту PACs з бібліотеки класів проекту) – ці об'єкти використовуються для зображення предметної області задачі.

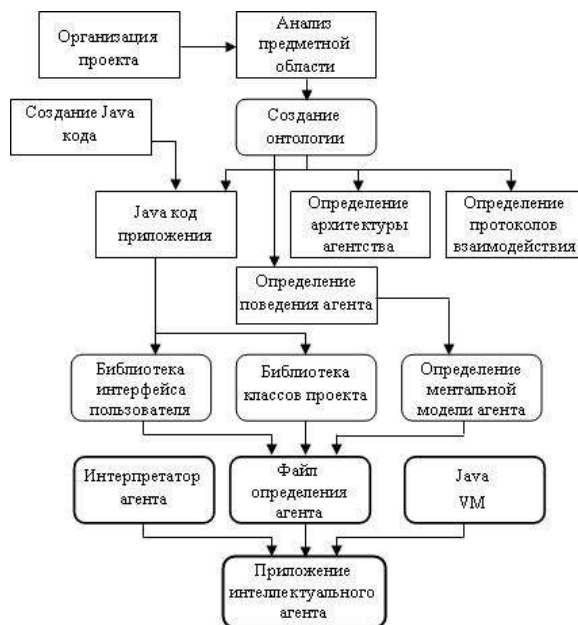


Рисунок 1. Процесс создания интеллектуального агента в инструментарии AgentBuilder

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.

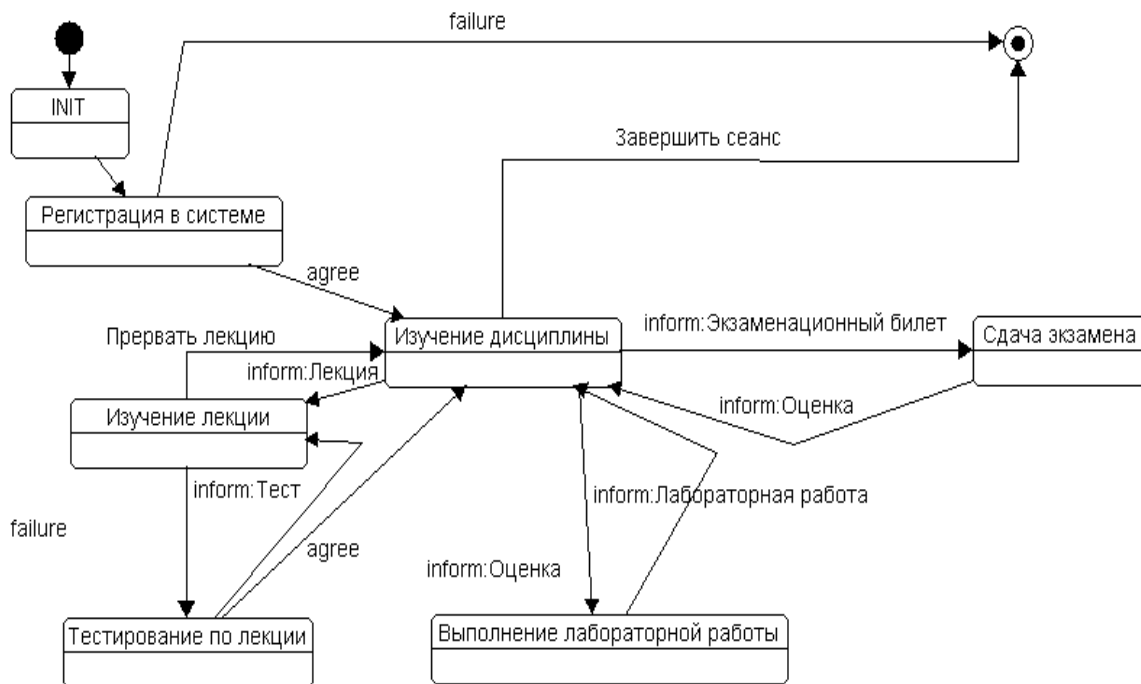


Рисунок 2. Граф состояний поведения агента Студент

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.

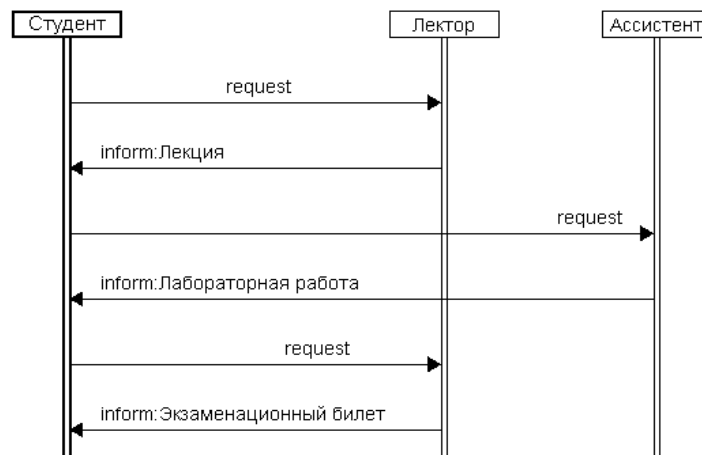


Рисунок 3. Диаграмма взаимодействия агента Студент в состоянии “Изучение дисциплины”

Таким образом, на основе разработанных логических моделей, система Bee-gent автоматически генерирует на языке Java скелет программного кода многоагентной системы, который дополняется 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 декларация 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 виртуальної машини Java, і будь-які інші файли Java необхідні файли.

Файли, які створюються для цих понять, повинні мати таке ж ім'я, як і у об'єкта, визначається в файлі. Вони мають розширення, визначає тип JASK поняття.

Компілятор агентів JASK конвертує вихідні файли на мові агентів JASK в код на мові Java, який потім компілюється в код виртуальної машини Java для виконання на цільовій системі

Програмна середовище JADE

Програмна середовище JADE (Java Agent Development Framework) отримала широке застосування для розробки багатоагентних систем. Вона повністю реалізована на мові Java і підтримує FIPA – стандарти для створення інтелектуальних агентів. Мета створення середовища JADE – спростити процес розробки за допомогою стандартизації способів взаємодії агентів в універсальному середовищі системних сервісів. Для досягнення цієї мети JADE пропонує програмісту-розробнику агентних систем наступні можливості:

- агентну платформу *FIPA-compliant Agent Platform*, засновану на FIPA і включає обов'язкові типи системних агентів для управління, в-перших, агентної платформою (AMS), в-других, каналом комунікації (ACC) і служби каталогів (DF) (ці типи агентів автоматично активуються при запуску платформи);
- розподілену агентну платформу *Distributed Agent Platform*, яка може використовувати декілька хостів, при чому на кожному вузлі запускається тільки одна Java Virtual Machine. Агенти виконуються як Java-потіки. В залежності від місцезнаходження агента, відправлює повідомлення, і того, хто його отримує, для доставки повідомлень використовується відповідний транспортний механізм.
- *Multiple Domains support* – ряд заснованих на FIPA DF-агентів можуть об'єднатися в федерацію, таким чином реалізуючи мультидоменну агентну середовище.
- *Multithreaded execution environment with two-level scheduling*. Кожен JADE-агент має власний потік управління, але він також здатний працювати в багатопотоковому режимі. Java Virtual Machine проводить планування завдань, виконуваних агентами або одним з них.
- *Object-oriented programming environment*. Більшість концепцій, властивих FIPA-специфікації, представляються Java-класами, формуючими інтерфейс користувача.
- *Library of interaction protocols*. Використовуються стандартні інтерактивні протоколи fipa-

request і fipa-contract-net. Для того, щоб створити агента, який міг би діяти згідно таким протоколам, розробникам прикладних програм потрібно тільки імплементувати специфічні доменні дії, в той час як вся незалежна від прикладної програми протокольна логіка буде виконуватися системою JADE.

- *Administration GUI*. Прості операції управління платформою можуть виконуватися через графічний інтерфейс, що відображає активних агентів і контейнерів агентів. Використовуючи GUI, адміністратори платформи можуть створювати, знищувати, переривати і оновлювати дії агентів, створювати ієрархії доменів і мультиагентні федерації DF (фасилітаторів).

JADE базується на технологіях Java RMI, Java CORBA IDL, Java Serialization і Java Reflection API. Розробка MAC в цій середовищі спрощується завдяки використанню FIPA-специфікацій і ряду інструментів підтримки фази налагодки і розгортання системи. Ця агентна платформа може встановлюватися на комп'ютерах з різними операційними системами, і її можна налаштувати через віддалений GUI-інтерфейс. Процес налаштування цієї платформи достатньо гнучкий: її можна змінити навіть в час виконання програм, за допомогою переміщення агентів з однієї машини на іншу. Єдиним вимогою для роботи системи є встановлення на машині Java Run Time 1.2.

Кожний запущений екземпляр середовища JADE є контейнером, т.к. може містити декілька агентів. Група активних контейнерів утворює платформу. Головний контейнер завжди повинен бути активним, а всі інші контейнери повинні бути зареєстровані при їх створенні. Тому, перший контейнер, запущений на платформі є основним контейнером, а всі інші – звичайними контейнерами і повинні отримати вказівку про те, де знаходиться їх основний контейнер, на якому вони повинні бути зареєстровані.

Якщо в мережі запускається ще один основний контейнер, то він представляє собою іншу платформу, на якій нові звичайні контейнери мають можливість зареєструватися. На рис. 4 показані приведені вище концепції платформи і контейнера і показує сценарій з двома JADE платформами, що складаються з трьох і одного контейнера відповідно. JADE агенти повинні мати унікальні імена, знати імена одне одного, і, завдяки цьому, вони можуть спілкуватися безпосередньо, незалежно від їх фактичного місцезнаходження, т.е. всередині одного контейнера (наприклад, агенти A2 і A3), в різних

контейнерах всередині однієї платформи (наприклад, A1 і A2) або в різних платформах (наприклад, A4 і A5).

Основний контейнер відрізняється від звичайних тим, що містить систему управління агентами і маршрутизатор, які автоматично запускаються при запуску основного контейнера.

Система управління агентами AMS (Agent Management System), представляє собою

«влада» в платформі (створення / видалення агентів в віддалених контейнерах, запитаних через AMS) і забезпечує службу іменування агентів.

Маршрутизатор каталогів DF (Directory facilitator), який забезпечує сервіс «Жовтих сторінок», допомагає знайти агенту інших агентів, для отримання від них необхідних послуг, необхідних йому для досягнення своїх цілей.

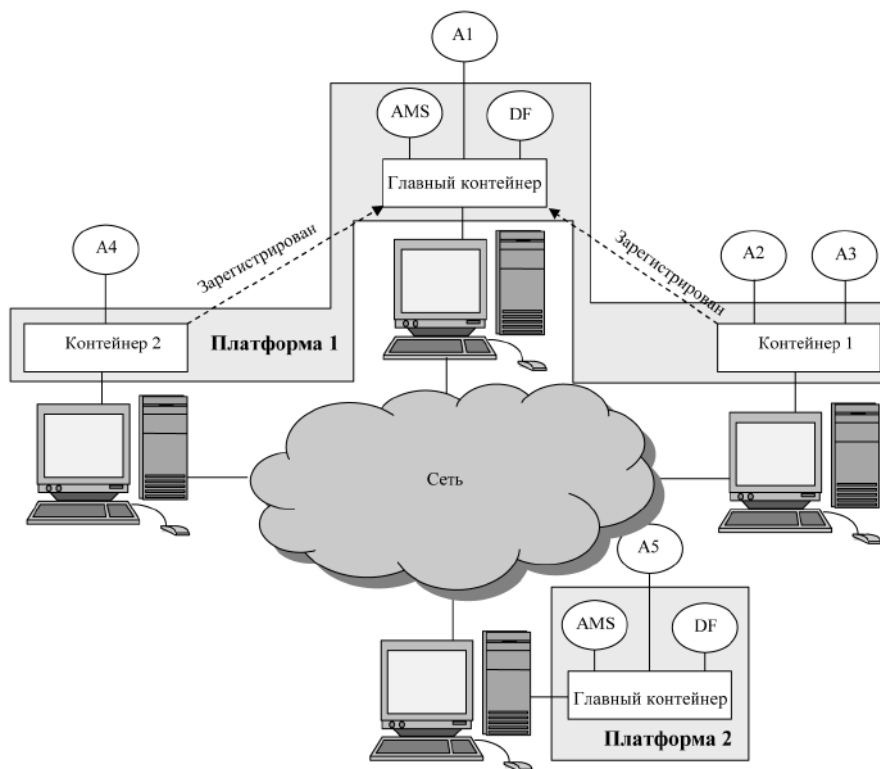


Рисунок 4. Среда «существования» агентов JADE.

Для здійснення комунікації архітектура середовища надає гнучкий і ефективний процес обміну повідомленнями, в якому JADE створює чергу і керує потоком ACL-повідомлень, являючись приватними для кожного агента. Агенти здатні звертатися до черги з допомогою комбінації декількох режимів своєї роботи: блокування, голосування, перерив в роботі і сопоставление с эталоном (якщо це стосується методів пошуку). В останніх версіях системи використовується Java RMI, event-notification і ПОО. Однак, можна легко додати і інші протоколи. Також передбачена можливість інтеграції SMTP, HTTP і WAP. Більшість комунікаційних протоколів, які вже визначені міжнародним співтовариством розробників агентних серед, доступні і можуть ілюструватися на конкретних прикладах після визначення поведінки системи і її основних станів. Разом з підтримкою визначених користувачем контентних мов, реалізовані

онтології управління агентами, а також онтології, які можуть бути реалізовані і зареєстровані агентами і використані системою. З метою суттєвого розширення спроможності JADE, передбачена можливість інтеграції з JESS і Java-оболочкою CLIPS. JADE використовується рядом компаній і академічних груп. Серед яких можна виділити такі відомі як: BT, CNET, NHK, Imperial College, IRST, KPN, University of Helsinki, INRIA, ATOS.

Заключення

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 розробляемій багатоагентній навчаючій системі по дисциплінам кафедри, являються підходи, пропонувані інструментарієм AgentBuilder і агентно-орієнтованою середою JACK.

Достоїнство даних інструментарієв заключається в використанні останніх досягнень для створення архітектури інтелектуальних агентів [9,18]. Дуже важливо, що обидва інструментарії забезпечують моделювання ментальних властивостей агентів, необхідних для розв’язання задачі навчання, оскільки побудова системи, яка володіє заданими властивостями, напряму залежить від того, наскільки успішно реалізовано поняття “агент” і пов’язані з ним властивості.

Слід згадати наявність у цих системах механізму висновків, необхідних для створення програмними агентами делегованої їм інтелектуальної діяльності.

Заслужує уваги простота зміни характеру діяльності агентів. Це забезпечується тим, що напрямки діяльності інтелектуальних агентів

визначаються на основі логічного висновку. Для зміни характеру функціонування МАС в цілому достатньо змінити знання кожного агента про свою діяльність. В інструментарії AgentBuilder це досягається шляхом зміни правил поведінки, а в середою JACK зміною логіки висновків, кожного з автономних агентів в складі МАС.

Від успішної реалізації штучного агента залежить можливість побудови системи, яка володіє новими властивостями, які неможливо отримати інакше як з допомогою агентно-орієнтованого підходу.

Проведений аналіз найбільш відомих інструментальних систем дозволив вибрати ефективний і доступний мову JACK для програмної реалізації навчаючої системи

На основі створеної комп’ютерної системи навчання агентно-орієнтованого типу нового покоління буде проведено аналіз адекватності ментальних моделей учасникам навчального процесу і отримано оцінку ефективності агентної архітектури в цілому.

Таблиця 1. Порівняльний аналіз інструментарієв для розробки агентно-орієнтованих програм

<i>Возможности инструментальных систем</i>	<i>Agent Builder</i>	<i>Bee-gent</i>	<i>JACK</i>	<i>JADE</i>
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
Експертна оцінка можливостей інструментальних засобів, %	90%	20%	70%	40%

Литература:

- Емельянов В. В., Тарасов В. Б. Виртуальная кафедра в техническом университете// Дистанционное образование. – 2000. – № 6. – С.39–45.
- 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). – Донецьк: ДонНТУ, 2006.- с. 105 - 116.
- Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика.- М.: “Едиториал УРСС”, 2002. -352 с.
- 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.
- 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 AAI'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 Resource] / 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