КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.75: 681.3

А.Я. ГЛАДУН, М.В. НЕСЕН, В.Н. ШТОНДА

ИНТЕЛЛЕКТУАЛЬНЫЕ АГЕНТНО-ОРИЕНТИРОВАННЫЕ УСЛУГИ, БАЗИРУЮЩИЕСЯ НА ПЛАТФОРМАХ ИНТЕЛЛЕКТУАЛЬНЫХ СЕТЕЙ

Главным недостатком интеллектуальных сетей и систем (Intelligence Network, IN) в настоящее время является то, что интеллектуальные услуги, спецификацию которых нам представляет организация по стандартизации ITU-T, еще слабо поддерживаются. Набор услуг (Capability Set) продолжает пополняться, однако он еще недостаточно широк для внедрения их во все сферы человеческой деятельности. Особенно недостаток интеллектуализации услуг ощущается в задачах автоматизации бизнес-приложений. В данной работе предлагается подход по реализации интеллектуальных услуг корпоративных систем на базе мультиагентных систем, позволяющий повысить функциональность существующих корпоративных систем в сфере автоматизации деловых процессов.

© А.Я. Гладун, М.В. Несен, В.Н. Штонда, 2004

Введение. Одной из главных проблем при решении многоаспектной проблемы автоматизации современных бизнес–приложений является недостаток реальной автоматизации многих задач. Недостатком корпоративных систем и интеллектуальных сетей в настоящее время является то, что интеллектуальные услуги, спецификацию которых нам представляет организация по стандартизации ITU-T, еще слабо поддерживаются. Набор услуг (Capability Set) продолжает пополняться, однако он еще недостаточно широк для внедрения их во все сферы человеческой деятельности. Причиной этого явилось то, что исторически первые IN-системы были сфокусированы для реализации телефонных услуг и в период формирования архитектуры IN, услуги по поиску, передаче и обработке данных, автоматизации деловых процессов не имели первостепенного значения. В настоящее время телекоммуникационные и компьютерные приложения и услуги развиваются в направлении сплошных телематических услуг (мультилингвистические системы, интеллектуальные системы поиска информации, автоматизация бизнес-процессов, телемедицина, дистанционное обучение, системы обработки графической информации и др.). Кроме того, сегодня мы наблюдаем процесс интеграции различных сетей (мобильных и наземных, специализированных сетей Ad-hoc, VPN с сетями общего (публич-

ного использования (Internet, Frame Relay и т.д.), что позволяет в свою очередь сделать прогресс в архитектуре IN на основе использования всех существующих на сегодня перспективных сетевых технологий и тем самым еще более расширить сферу применения интеллектуальных услуг. В данной работе предлагается подход, основанный на агентно-ориентированных технологиях (АОТ), позволяющий повысить функциональность существующих IN-систем в сфере внедрения интеллектуальных услуг.

Главное преимущество при внедрении систем е-бизнеса – сокращение стоимости и скорость реализации деловых процессов, а также предоставление более удобных услуг клиентам. Однако при использовании е-бизнес систем сегодня ощущается недостаток реальной автоматизации многих задач.

На современном этапе развития информационных технологий для решения этих задач с успехом применяется АОТ, которая базируется на использовании интеллектуальных программных агентов и позволяет увеличить функциональные возможности современных распределенных систем [1–3]. АОТ – это интегрированная технология, использующая различные источники и концепции: теория решений; распределенные системы; объектно-ориентированная технология; теория организаций; системы баз знаний [4, 5].

Исследованиями в этой области занимаются ученые разных стран, в частности, – профессор В.Г. Городецкий (Россия, Санкт-Петербург) [5], ученые США, Германии, Франции, Японии, а также ведущие мировые корпорации – производители программного обеспечения (Microsoft, Motorolla, ForeSystems, Siemens, Fujitsu и др.) [1–7]. Каждый такой продукт содержит в себе определенное “ноу-хау”, определяющее эффективность работы агентов в распределенной системе.

Мобильные агенты, перемещаясь по сети, исполняются на различных ее узлах независимо от платформы. По словам Джорджа Люггера, если считать появление промежуточного программного обеспечения первым этапом эволюции архитектуры информационных систем, первоосновой которой была клиент-серверная модель, то технология мобильных агентов – ее следующий этап [1].

Мобильные агенты не заменяют собой программного обеспечения (ПО) middleware, но существенно расширяют его возможности.

Мобильным агентом является агент, который имеет способности к перемещению в распределенной информационной среде [1, 2] и который имеет следующие свойства: автономность; взаимодействие; реактивность (реагируют на изменения среды в реальном времени, обычно их деятельность описывается следующим образом: *WHEN event IF condition THEN action*); проактивность; способность существовать как постоянно выполняющийся процесс; интеллектуальность (обучаемость).

Основная задача работы – исследование вопросов разработки базовой мультиагентной системы (МАС) на базе Java для решения задач е-бизнеса. Для реализации модуля принятия решений, который входит в состав МАС использовался аппарат нечетких множеств (fuzzy aproach) – сочетание аддитивного и мультипликативного критериев принятия решений [5, 8]. Кроме того, в данной работе мы придерживались ограничений, согласно которым первые три свойства

агента являются обязательными, а остальные – необязательными. Обобщенная архитектура интеллектуального агента [6], которая включает в себя главные компоненты: обеспечение по самоуправлению; комплекс целей; хранилище для данных; компонента по безопасности и компонента связи для взаимодействий с другими агентами, ресурсами системы и пользователями, показана на рис.1.

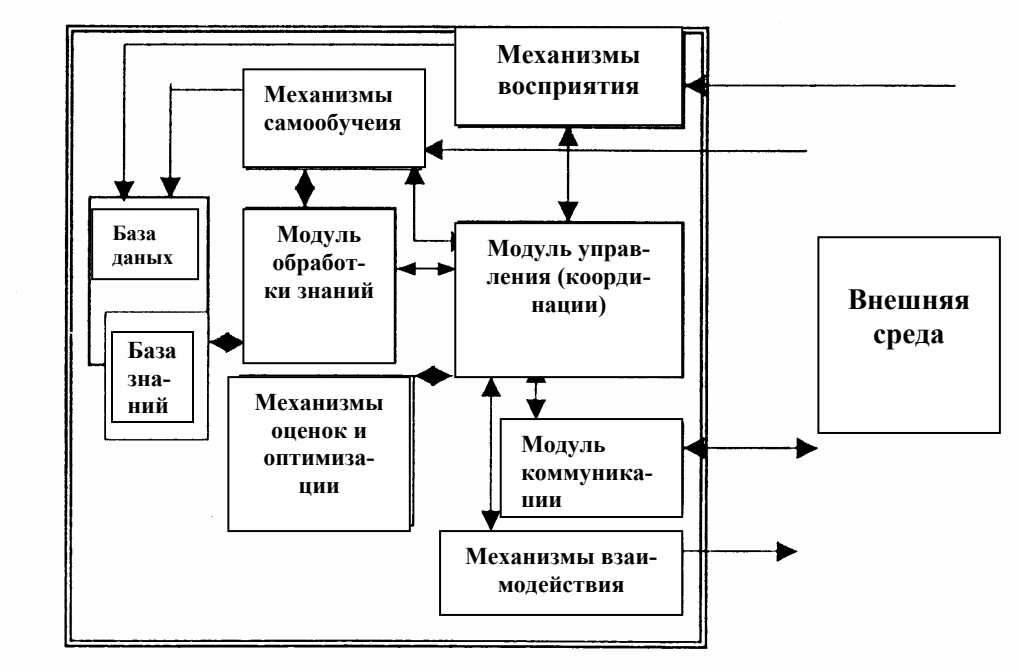

РИС.1 . Обобщенная архитектура мобильного интеллектуального агента

Разработка МАС для задач электронной коммерции. Суть проблемы в том, что, несмотря на значительный прогресс в области теоретических исследований по МАС, новых возможностей оказалось недостаточно для создания таких систем. Для создания действительно сложных открытых МАС (ОМАС), такие системы должны постоянно «жить» на сервере предприятия и непрерывно участвовать в решении задач, а не запускаться от случая к случаю. Известные в настоящее время МАС пока ориентированы в основном на применения только в области е–коммерции и поиска в Интернет, не имеют возможностей представления и использования корпоративных знаний, сложны в разработке, не располагают необходимыми инструментальными системами, не обеспечивают большого числа агентов и высокой скорости работы и т. д. В связи с этим основной целью представленной работы была попытка реализации простой МАС, формирование требований к МАС и предоставление разработчикам практической методики реализации МАС на Java. На базе этой методики показаны этапы практической реализации МАС и разработан обобщенный алгоритм построения МАС.

Распределенная платформа системы мобильных агентов. Рассмотрим основные технические предпосылки создания МАС**.** При использовании МАС в каждом вычислительном узле должен быть сервер (далее – агентская система), то есть платформа, предоставляющая среду для создания /уничтожения, приема /передачи и среду для выполнения агентов.

Агентская система может иметь своего владельца. Агентская система имеет определенный тип, иначе агенты, управляемые системой, имеют некоторый профиль (например, в понятие профиля агента входят язык реализации агента, производитель, алгоритм сериализации агента). Профили необходимы для обеспечения взаимодействия различных систем – зная профиль агента, система в состоянии определить, может ли она обеспечить среду для выполнения этого агента [7]. Агентская система должна поддерживать понятие Места (place) [11]. В этом случае сеть компьютеров представляется как набор мест, предоставляющих сервисы. С другой стороны, место – контейнер и фабрика агентов. Агент порождается в месте и умирает в месте. В течение жизненного цикла агент перемещается между местами. Плэйсы могут существовать как на стороне клиента, так и на стороне сервера. Плэйс может хранить "следы" посещавших его агентов.

Требования к реализации агентских систем. На основе анализа существующих МАС [3–7] выделим две МАС (MADAE: Multi-Agent Engine for E-bussiness Applications и WAE: Multi-Agent Engine for Web-Based Applications), предназначенные для построения ОМАС в сети Интернет. Рассмотрев аналоги, сформулируем следующие требования к проектируемой МАС:

1) обеспечение переносимости кода на различные платформы. Понятие мобильности неразрывно связано с понятием переносимости. Переносимость кода можно обеспечить двумя различными способами: посредством использования интерпретируемых языков (Perl, Tcl и др.); посредством использования одного из языков, поддерживающих раздельную компиляцию (Java, Oberon и др.);

2) доступность на множестве платформ. Это требование является продолжением предыдущего. Мобильные агенты должны осуществлять свою работу в гетерогенной компьютерной среде;

3) поддержка сетевого взаимодействия. Помимо операций непосредственно связанных с перемещением между агентскими серверами, агент должен обладать средствами для коммуникации с другими агентами и доступа к удаленным ресурсам;

4) многопоточная обработка. Для реализации одновременного выполнения нескольких действий агентская система должна включать в себя поддержку параллельного выполнения функций агента и поддержку средств синхронизации;

5) безопасность. Мобильные агенты, приходящие из сети, могут содержать потенциально опасный, вредоносный код. Поэтому система должна поддерживать средства безопасности, достаточные для ее нормальной работы.

Обоснование использования Java при реализации МАС. Технология Java предоставляет открытую, стандартную, универсальную платформу для сетевых вычислений. При разработке технологии особый акцент был сделан на незави-

симость приложений Java от конкретной аппаратно-программной платформы (что и позволяет успешно обмениваться в гетерогенной вычислительной среде приложениями и даже их фрагментами). Эта цель достигается с помощью языка *Java* и виртуальной *Java-машины,* в коды которой (так называемые, *байт–коды*) транслируются *Java–приложения.*

Java предусматривает создание приложений, переносимых на различные платформы. Программа, написанная на Java, компилируется в специальный машинно-независимый байт-код. Затем этот код может быть исполнен с помощью интерпретатора Java на любом компьютере, где реализована *JVM* (платформонезависимость Java-приложений на уровне байт-кода). Мы выбрали язык Java, благодаря его интуитивной понятности – т. е. классы и методы имеют такие понятные названия, что исходный текст часто не требует комментариев, а сам проект программного агента очень удобно можно вложить в один архив **.jar**, независимо от количества файлов в этом проекте и запускать программу агента с командной строки **java –jar agent.jar**, не имея проблем впоследствии с типом ОС – UNIX, Windows, Solaris и т. д.

Последовательный алгоритм реализации МАС. Практическую реализацию МАС можно выполнить при условии выполнения следующих шагов:

1) необходимо четко определить функциональные задачи, которые должна выполнять реализуемая МАС (перечень задач приведен в [11]). МАС обеспечивают комплексную и наглядную среду для оценки разумного поведения агента и для оптимального использования МАС нужны специализированные знания этой предметной области;

2) выбор языка программирования на котором будет реализован агент. Язык программирования в современной информационно-телекоммуникационной среде должен поддерживать функции многопоточности для возможности одновременной параллельной обработки многих задач. Многопоточная обработка (“multithreading”) обозначает, что агент может выполнять некоторые действия одновременно. Тем самым, язык программирования агентов должен включать поддержку параллельного выполнения различных функций агента (типа “threads”) и различных примитивов синхронизации (семафоры, мониторы, критические секции и т. д.). Этот язык должен быть «интуитивно-понятным» и Интернет-ориентированным, а также позволять запускать потоки-демоны, работающие даже после завершения основной (материнской) части программы. Как известно, всеми этими качествами владеют два языка – .NET расширение для C++ и Java;

3) следующий этап реализации МАС – создание модели МАС и проектирование функциональных логических модулей из которых будет состоять МАС;

4) программная реализация. Каждый агент реализован отдельным файлом. Пользователь будет запускать агент из модуля интерфейса и уже потом будет осуществляться инициализация главного модуля;

5) тестирование программной реализации МАС. Тестирование должно включать возможные неожиданные действия со стороны пользователя и соответствующую реакцию программы на эти действия;

6) фаза отладки. В этой фазе реализации МАС в текст программы вносятся изменения в соответствии с обнаруженными недостатками;

7) фаза эксплуатации МАС. Эта фаза может выполняться совместно с фазой тестирования при возникновении непредвиденных ситуаций.

Описание модели МАС. Для разработки логической модели МАС мы использовали язык UML. При этом очень важно учесть особенности структуры МАС. Каркас МАС должен включать следующие модули: модуль интерфейса; модуль функций для взаимодействия с пользователем (обработчик событий); главный модуль МАС – модуль координации и управления (в соответствии с поставленной задачей) и модуль дополнительных функций для работы с данными (сортировка, фильтрация, поиск и т. д.); модуль возврата результатов пользователю (в виде log-файла – сообщений на интерфейс пользователя). С учетом классов и методов языка Java, мы получили следующую модель МАС (рис.2).

<table>
<tr><td>AboutDialog</td><td>Better
BayerAgent</td><td>Marketplace
App</td></tr>
<tr><td>Basic
Negotiation</td><td>Better
SellerAgent</td><td>Marketplace
Frame</td></tr>
<tr><td rowspan="2">Best
BayerAgent</td><td>BayerAgent</td><td rowspan="2"></td></tr>
<tr><td>BaySel
Message</td><td>Offer</td></tr>
<tr><td>Best
SellerAgent</td><td>Facilitator
Agent</td><td>SellerAgent</td></tr>
</table>

РИС. 2. UML-модель мультиагентной системы для задач электронной коммерции

Модель МАС содержит следующие модули: Offer – определения цены товара на торгах; BasicNegotiation – ведения базовых переговоров (правила); AboutDialog – ведение диалога (интерфейс с пользователем); Marketplace Frame– блок координации и управления МАС; BaySelMessage – интерфейс взаимодействия с пользователем; FacilitatorAgent – агент-посредник; BetterBuyerAgents – наилучший агент-покупатель; BestBayerAgent – лучший агент-покупатель; BayerAgent – агент-покупатель; BetterSellerAgents – наилучший агент-продавец; BestSellerAgents – лучший агент-продавец; SellerAgents – агент-продавец.

Пример программной реализации МАС на Java. На основе Java, разработан FacilitatorAgent (агент–посредник), который управляет рынком, агенты BuyerAgents (агент–покупатель) и SellerAgents (агент–продавец), используемые для взаимодействия внутри этого рынка. Эти интеллектуальные агенты получены из базового класса CIAgent, который детально описан в [4]. Агенты Клиента и Продавца (Торговца) различаются прежде всего сложностью ихних стратегий переговоров, которые начинаются с простой логики (в терминах if-then-else), а затем переходят к методам формирования правил, которые базируются на приобретенных фактах.

FacilitatorAgent – посредник между Клиентами и Торговцами. Все агенты обязаны зарегистрироваться у Посредника перед тем, как они начнут взаимодействовать с какими–либо другими агентами на рынке (marketplace).

Торговцы рекламируют желание продать товары или услуги с помощью Посредника, в то время как, Клиенты также просят Посредника рекомендовать им вероятного продавца. Как только агенты Клиента и Торговца будут представлены Посредником, они будут продолжать общаться только через него. Мы реализовали программу – MARKETPLACE на Java, на основе UML-модели МАС. В рабочем окне программы используются две текстовые зоны, которые показывают сообщения от Посредника, Покупателя и Продавца на рынке. Пункты меню «Файл» включают Начало и Останов. Меню Вид обеспечивает альтернативный вывод детализированной информации или сообщения суммарного содержания. Меню Клиента и Торговца обеспечивают три типа Клиентов и Торговцев, для размещения на рынке. Эти агенты могут быть выбраны в любой комбинации Основных, Средних и Продвинутых Клиентов (термины «Основной», «Средний» и «Продвинутый» обозначают степень разумности, интеллектуальности соответствующих методов).

Все агенты запускают свои собственные потоки и просыпаются через конкретные фиксированные промежутки. Все связи между Клиентами, Торговцами и Посредником используют интерфейс CIAgent-Events и CIAgentEventListener. Объект аргумента, который передается с CIAgentEvents – новый объект под названием BuySellMeswisard, который формируется по образцу стандартного сообщения KQML.

Алгоритм процедуры ведения переговоров между агентами в МАС. KQML конкретизирует формат и некоторое содержание взаимодействий между Торговцем и Клиентом. Ниже представлена процедура ведения переговоров сбыта, которая используется в программе MARKETPLACE:

1) Клиент просит Посредника отрекомендовать ему одного из Торговцев для покупки товара P;

2) Посредник говорит Клиенту имя Торговца;

3) Клиент спрашивает Торговца (через Посредника), имеет ли Торговец товар P для продажи;

4) Торговец же или откликается на предложение сотрудничества с Клиентом (сообщая о товаре P, уникальном идентификаторе товара (ID) и начальной желаемой цене) или же дает отрицательный ответ, указывая, *что* товара P для продажи нет;

5) Клиент может принять предложение путем пересылки соответствующего предложения обратно Торговцу или сделать альтернативное предложение Торговцу (указывая другую желаемую цену);

6) Торговец может: а) принять предложение, б) сделать еще одно альтернативное предложение, в) отменить предложение.

7) в случае, если Торговец принимает предложение, тогда он отсылает tell-сообщение Клиенту. Таким образом, договор о сбыте будет завершен;

8) в случае, если Торговец отбрасывает предложение, то переговоры продолжаются дальше.

Клиент и Торговец никогда не общаются непосредственно, а используют для этого FacilitatorAgent, как посредника в переговорах о купле-продаже. Менеджер коммуникаций (BaySelMessage) содержит в себе сообщения, представленные на языке коммуникаций с примитивами типа примитивов языка KQML: обратиться с просьбой, принять, отвергнуть, изменить, предложить, проинформировать, запросить данные, отказаться и подтвердить.

Выводы. На основании сформированных требований к МАС была разработана агентная система для электронной коммерции. Модуль принятия решения в МАС построен с использованием теории нечетких множеств (сочетание числового и лингвистического подходов). Алгоритм принятия решения позволил выделить три группы агентов в системе по уровню их “интеллектуальности”. В работе рассмотрена программа электронного рынка с использованием семерки CIAgent-базовых “умных” агентов. Программа содержит следующие модули:

1) Marketplace-программа, модуль состоящий из единого FacilitatorAgent, одного или более BuyerAgents, и одного или более SellerAgents. Использованы три типа агентов Клиента и Торговца, которые использовали (в нарастающем порядке) лучшие стратегии ведения переговоров;

2) все связи между Клиентами и Торговцами проходят через Посредника на основе использования KQML-подобных объектов под названием BuySellMessages. Эти сообщения включают такие KQML-performatives, как например, “запись”, “рекламируют”, “не рекламируют”, “не рекомендуют один”, “не спрашивают”, и “не говорят”;

3) описан протокол переговоров сбыта, который предоставляет больше контроля над процессом продажи. Класс BasicNegotiation был представлен для того, чтобы инкапсулировать детали каждого договора. Предложение состоит из имени агента, имени товара, уникального ID товара, который генерируется вначале переговоров, и цены предложения;

4) BestBuyerAgent и BestSellerAgent используют базу правил типа булеан (BooleanRuleBase).

Представленная МАС может использоваться как для моделирования ситуаций связанных с рынком, так и для разработки готового программного продукта не только для электронной коммерции, но и для других бизнес–приложений.

Примерами использования агентов может быть поиск информации (data mining, агенты взаимодействуют с серверами баз данных и хранилищами данных), электронная коммерция [12].

Перспективы дальнейших работ. Реализованная МАС требует в дальнейшем проработки вопросов коммуникационной инфраструктуры для движения агентов в сети, а также расширения функциональных задач, решаемых МАС:

1) необходимо стандартизировать действия агентских систем при пересечении агентом нескольких доменов безопасности, а также формат представления кода и состояния агента при его перемещении между агентскими системами разных типов;

2) технология мобильных агентов достаточно новая, поэтому системы программирования мобильных агентов существенно различаются по архитектуре и реализации. Нужно проработать вопросы обеспечения интероперабельности между различными агентскими системами;

3) интероперабельность достигается при стандартизации таких аспектов, как передача агентов и служебных (используемых агентом) классов между агентскими системами, а также управление агентами;

4) необходимо решить задачу безопасности в МАС, в частности, наличие системы защиты от несанкционированного доступа и "плохих кодов";

5) кроме вышеперечисленных аспектов, необходима также стандартизация синтаксиса и семантики различных параметров, например, имен агентов и агентских систем, типов агентских систем.

Полученные результаты дают основание надеяться на успешное решение в будущем этой сложнейшей проблемы.

1. *Люгер Джордж Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание / Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 864 с.
2. *Russel Stuart J.* Artifical Intelligence: A modern approach. Prentice Hall, Upper saddle River, New Jersey, 1998. – 715 p.
3. *Rzevski G.* "On Behaviour and Architectures of Autonomous Intelligent Agents: An Engine-ering Perspective" // Proceedings First International Round – Table on Abstract Intelligent Agents, ENEA, Rome, 1993. – P. 178 – 188.
4. *Joseph P. Bigus, Jennifer Bigus* Constructing Intelligent Agents Using Java, Second Edition, Addison-Wesly, New York, 2002. – 520 p.
5. *Городецкий В.И., Грушинский М.С., Хабалов А.В.* Многоагентные системы (обзор). Санкт-Петербургский Ин –ут информатики и автоматизации РАН. – 1999. – 74 с.
6. *Тодорка Атанасова* Агентная технология: концепции, модели, приложения. – Варна, 2000. – 155 с.
7. *Гладун А.Я., Перевозчикова О.Л., Плескач В.Л.* Разработка OSI- профилей открытых систем // УСиМ. – 1999. – № 6. – С.40 – 56.
8. *Гриценко В.И., Гладун А.Я.* Агентно–ориентированные WEB-технологии в интеллекту-альных сетях: новые области реализации интеллектуальных услуг // Тр. Межд. научно-практической конф. "Современные и будущие информационные технологии Украины", Киев, 15 – 17 марта, 2000. – С. 63 – 68.
9. *Гладун А.Я., Плескач В.Л.* Использование агентно–ориентированных технологий в телекоммуникационных сетях // Проблемы программирования. – 2000. – № 1. – С. 43 – 59.
10. *Гладун А.Я., Журавльов Ю.Д., Штонда В.М.* Аналіз вимог та особливості реалізації архітектури корпоративної системи керування електронним документообігом // Технічні вісті. – Львів. – 2003, № 1 (16), № 2 (17). – С.28 – 33.
11. *Плескач В.Л., Гладун А.Я.* Архитектурная концепция интеллектуальных сетей // Искусственный интеллект. – 1999. – № 2. – С. 413 – 421.
12. *Витих В.А.* Управление открытыми системами на основе интеграции знаний // Автометрия. – 1999. – № 3. – С. 38 – 49.

Получено 15.01.2004