УДК 004.75: 681.3

МОДЕЛЬ МУЛЬТИАГЕНТНОЙ СИСТЕМЫ ДЛЯ Е-БИЗНЕСА И ТЕХНОЛОГИЯ ЕЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

В.И. Гриценко, А.Я. Гладун, Ю.Д.Журавлев, М.В. Несен

Международный научно-учебный центр информационных технологий и систем НАН и МО Украины
03680, МСП Киев-187, проспект Академика Глушкова,40
тел.: (044)-266-34-96, email: glad@irtc.kiev.ua

Улучшение эффективности выполнения задач е-бизнеса требуют дальнейшего разви-тия методов автоматизации деловых процессов. Автоматизация может использо-ваться на различных стадиях и различных прикладных областях е-бизнеса. Мультиа-гентные системы - это новаторские технологии, способствующие автоматизации ряда бизнес-процессов: ведение автоматизированных переговоров; доверие между бизнес-партнерами; выполнение задач от имени некоторого владельца; анонимность переговоров и т.п. Авторами предложена модель мультиагентной системы для задач е-коммерции и ее программная реализация на Java. В заключении статьи поставлены за-дачи для построения коммуникационной инфраструктуры интеллектуальных агентов в распределенной среде.

Further development of business processes automatization methods is required for improvement of e-business goals productivity. Automatization could be used on different stages and in various applied areas of e-business. Multi-agent systems are innovative technologies that help on a number of business processes automatization: automatic negotiating; achieving confidence between partners; performing tasks in an owner's own name; anonymity of negotiation etc. Authors offer multi-agent system model for e-commerce goals and its Java implementation. Tasks of intelligent agent communicational infrastructure building in distributed environment are set in the article's conclusion.

Введение.

В условиях поиска оптимальных путей информатизации общества и вхождения Украины в мировое информационное пространство первоочередное значение имеет решение многоаспектной проблемы автоматизации современных бизнес–приложений: электронной коммерции (e–commerce); корпоративного электронного документооборота (EDI); электронной биржи (e–exchange); электронного рынка (e–market); электронного магазина (e–shop); электронного аукциона (e–auction) и др. Программные модули этих электронных технологий входят в состав интегрированных корпоративных систем и решают сложные задачи по автоматизации и оптимизации бизнес–процессов. Главное преимущество при внедрении систем е-бизнеса это сокращение стоимости и скорость реализации деловых процессов, а также предоставление более удобных услуг клиентам. Однако, при использовании систем е-бизнеса, сегодня ощущается недостаток реальной автоматизации многих задач.

Для решения этих задач с успехом применяется агентно–ориентированная технология (АОТ), которая базируется на использовании интеллектуальных программных агентов и позволяет увеличить функциональные возможности современных распределенных систем [1-4]. На рис.1 представлена схема взаимодействия современных технологий: АОТ, WWW и приложений е-бизнеса. АОТ – это интегрированная технология, использующая различные источники и концепции: теория решений; распределенные системы; объектно-ориентированная технология; теория организаций; системы баз знаний [5-7].

По своему характеру АОТ является новой парадигмой программирования, расширяющая возможности объектно–ориентированного программирования. Исследованиями в этой области занимаются ученые разных стран, в частности - проф. 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.

Слово агент восходит к латинскому agere - вести, действовать. Главное качество агентов - способность выполнять какую-то делегированную ему работу в чьих-то интересах. В этом их сходство с роботами. Термины «програмный агент» или более общий «интеллектуальный (разумный) агент» становятся все более близкими как для разработчика программного обеспечения, так и для пользователя. Поэтому во многих бизнес–приложениях актуальной задачей становится использование программных агентов, способных воспринимать ситуацию, двигаться в распределенных системах, сетях (быть мобильными) и принимать решения. Классификация агентов в соответствии с их архитектурой приведена в работе [8].

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].

Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:
Удалено:

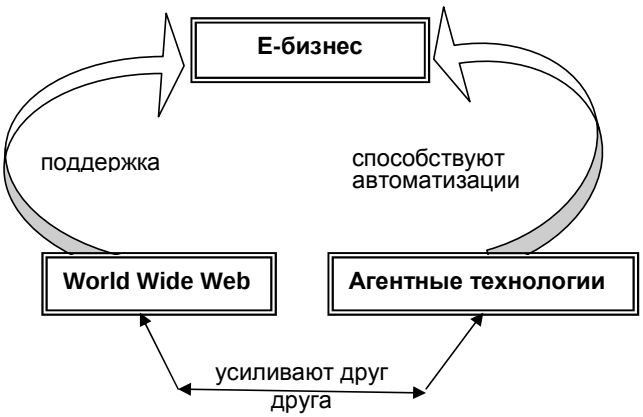

Рис. 1 Агентно-ориентированные технологии для интеллектуализации задач е-бизнеса

Мобильные агенты не заменяют собой программного обеспечения (ПО) middleware, но существенно расширяют его возможности. Прежде всего, сама природа агентов предполагает асинхронную связь. Например, агент может «собрать» несколько запросов пользователя и уже самостоятельно передать их для обработки на сервер базы данных. При этом устраняется необходимость в установке связи с сервером для обработки каждого запроса. Кроме того, при работе с мобильными агентами клиенту не нужно знать, какая операционная система или какая база данных установлена на том или ином сервере, все функции организации связи и преобразования агент берет на себя.

Мобильным агентом является агент, который имеет способности к перемещению в распределенной информационной среде [1-2] и который имеет следующие свойства:.

- автономность – агенты могут выполнять свои задачи без непосредственного вмешательства клиентов или других агентов;
- взаимодействие – при возникновении потребности агенты взаимодействуют с другими агентами или людьми с целью получения или оказания помощи в решении задачи;
- реактивность – агенты реагируют на изменения среды в реальном времени, обычно их деятельность описывается следующим образом:

  WHEN event IF condition THEN action;
- проактивность – способность решать задачи (достигать цели); в отличие от реактивных агентов, они не просто реагируют на изменения среды, но и сами ее опрашивают;
- способность существовать как постоянно выполняющийся процесс, точнее, иметь собственный поток управления;
- способность выполняться от имени некоторого владельца;
- гибкость – действия агентов не фиксированы жестко;
- интеллектуальность (обучаемость) – способность находить новые решения; такие агенты могут изменять свое поведение, используя как свой опыт, так и опыт других агентов.

Основной задачей данной статьи является исследование вопросов разра-ботки 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 [5,10]. Кроме того, в данной работе мы придерживались ограничений, согласно которым первые три свойства агента являются обязательными, а остальные – необязательными.

Исследования, проводимые в мире по МАС, имеют уже почти десятилетнюю историю. Этот интерес обусловлен достижениями в области информационных технологий, искусственного интеллекта, распределенных информационных систем и компьютерных сетей. МАС интегрируют в себе самые передовые достижения перечисленных областей, демонстрируя принципиально новые качества. Коммерческие разработки по МАС поддерживаются такими известными фирмами, как Apple, Hewlett Packard, Digital, японскими фирмами [7,8,12].

Разработка МАС для электронной коммерции.

Ключевым элементом МАС является программный агент, способный воспринимать ситуацию, принимать решения и быть коммуникабельным с другими агентами МАС. Эти новые возможности значительно отличают МАС от существующих «жестко» организованных систем. При этом отдельные модули программы получают возможность договариваться о том, как должна решаться задача. Эти модули приоб-

ретают собственную активность и инициируют диалог с пользователем, они должны работать в условиях неопределенности и предлагать уточнения и переформулировку задач.

Суть проблемы в том, что несмотря на значительный прогресс в области теоретических исследований 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. Кроме того, 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. Поэтому **основной целью** представленной работы была попытка реализации простой МАС и формирование требований к МАС, предоставление разработчикам практической методики реализации МАС на Java. На базе этой методики показаны этапы практической реализации МАС для электронной коммерции и разработан обобщенный алгоритм построения МАС.

Анализ работ в области МАС позволил выделить следующие основные направления исследований в этой области [4-7]:

- 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 МАС - область исследований, в которой изучается, как построить информационную систему, которая удовлетворяет тем или иным свойствам, которые выражены средствами теории агентов;
- языки программирования агентов;
- методы, языки и средства коммуникации агентов;
- методы и программные средства поддержки мобильности агентов (миграции агентов по сети).

Особое место занимают исследования, связанные с разработкой приложений МАС и инструментальных средств поддержки технологии их разработки.

При выборе архитектуры МАС следует учитывать два ее аспекта:

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

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, приведена на рис.2. Чтобы быть применимыми (полезными), мобильные агенты должны связываться с различными компьютерами, агентами и двигаться внутри разнородных сетей. Это требует реализации стандартизированного каркаса и методологии для действий агента через телекоммуникационные сети.

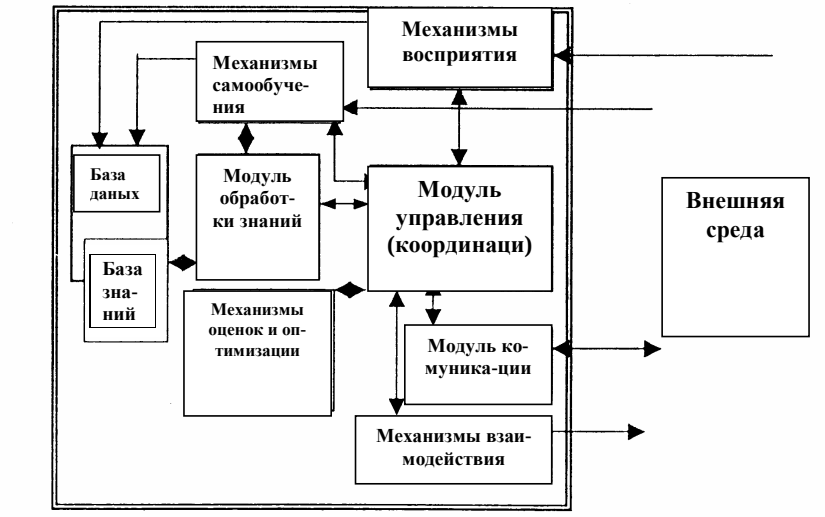

Рис.2 Обобщенная архитектура мобильного интеллектуального агента.

Распределенная платформа системы мобильных агентов.

Рассмотрим основные технические предпосылки создания МАС. При использовании МАС на каждом вычислительном узле должен быть сервер (далее – агентская система), то есть платформа, предоставляющая функциональность для создания/уничтожения, приема/передачи и среду для выполнения агентов. Для организации взаимодействия агентов должен существовать сервис именования, т. е. сервис, предоставляющий возможность работы с сущностью, зная только ее имя. Функции этого сервиса выполняет агентская система.

Агентская система может иметь своего владельца. Агентская система имеет определенный тип, то есть 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 ("плэйса", place). В этом случае сеть компьютеров представляется как набор мест, предоставляющих сервисы. С другой стороны, место является контейнером и фабрикой агентов. Агент порождается в месте и умирает в месте. В течение жизненного цикла агент перемещается между местами. Сервисы, предоставляемые местом, используются агентами, находящимися в нем. Плэйс может накладывать ограничения на ресурсы, которые используют агенты, находящиеся в нем. Плэйсы могут существовать как на стороне клиента, так и на стороне сервера. Плэйс может хранить "следы" посещавших его агентов. Агент может содержать "историю" своей жизнедеятельности (может использоваться для проверки поведения агента (например, для проверки ограничений безопасности) и для обучения агента на собственном опыте). В случае если понятие места ("плэйса") не поддерживается, его функции берет на себя сама агентская система.

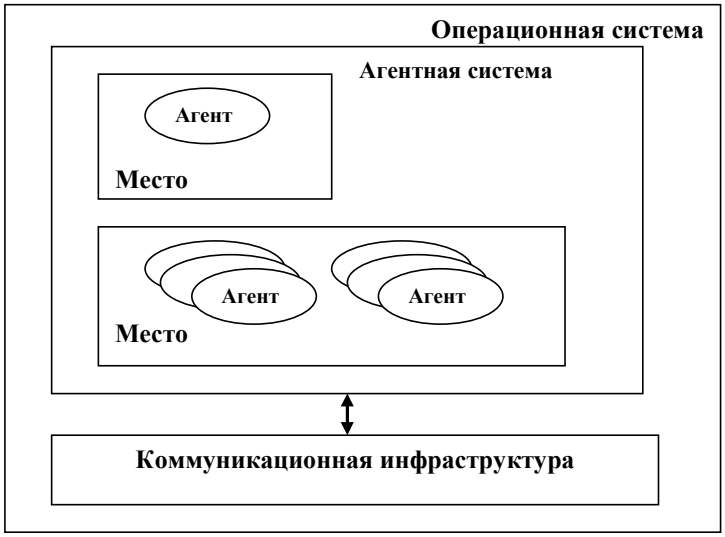

Рис.3 Платформа мультиагентной системы.

Функциональные задачи электронной коммерции.

В системе е-коммерции можно выделить две взаимосвязанные подсистемы: торговли и управления диалогом. В торговой подсистеме могут быть агенты товаров и заказов, а также агенты продавцов и покупателей, склада, поставщиков и т.д. Агенты товаров и заказов ведут переговоры со стратегиями скидок постоянным покупателям, скидок за оптовую покупку, скидок по состоянию конкурентов, скидок по затовариванию склада и др. При этом несколько агентов покупателей (потенциальных конкурентов) могут объединить свои заказы для получения большей скидки, т.е. перейти от конкуренции к кооперации. В подсистеме же управления диалога нужно создать систему выдачи результатов переговоров агентов.

Требования к реализации агентских систем.

На основе анализа сущест-вующих МАС [3-7] мы выделили 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, мы сформулировали 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. Поэтому поддержка сетевых услуг должна включать в себя широкий спектр возможностей (служба имен, RPC, OLE, CORBA, RMI и т.д.).

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

Система программирования на Java включает в себя виртуальную машину Java и транслятор с Java в bytecode. 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 *Java Virtual Machine* (платформо-независимость Java-приложе-ний на уровне байт-кода). Java - объектно-ориентированный язык программирования. Здесь каждый тип данных понимается как класс объектов, любая функция является методом класса. Ее вызов рассматривается с объектно-ориентированных позиций как посылка сообщения объекту. В Java имеется встроенная расширяемая библиотека классов, включающая модули управления окнами (AWT-Abstract Window Toolkit) для создания пользовательских интерфейсов, классы поддержки основных типов данных, многопоточности, сетевых возможностей, графики, мультимедиа и т. д.

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. Интернет–ориентированость языка означает его платформонезависимость (доступность на многих платформах). Это требование непосредственно вытекает из предыдущего, так как интеллектуальные агенты должны работать в гетерогенной компьютерной среде.

3. Следующим этапом реализации МАС является 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. При реализации МАС надо учитывать то, чтобы код был правильно структурированным и была возможность быстрого изменения его при необходимости.

5. Тестирование программной реализации МАС. Эта фаза может длиться в течении всего периода эксплуатации и модернизации системы. 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.

<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">Offer</td></tr>
<tr><td>BaySel Message</td></tr>
<tr><td>Best SellerAgent</td><td>Facilitator Agent</td><td>SellerAgent</td></tr>
</table>

Рис.3 UML-модель мультиагентной системы для задач електронной коммерции

Модель МАС содержит следующие модули: Offer – определения цены товара на торгах; Basic Negotiation – ведения базовых переговоров (правила); AboutDialog – ведение диалога (интерфейс с пользователем); Marketplace Frame- блок координации и управления МАС; BaySel Message- интерфейс взаимодействия с пользователем; FacilitatorAgent – агент посредник; Better BuyerAgents – наилучший агент–покупатель; Best BayerAgent - лучший агент–покупатель; BayerAgent - агент–покупатель; Better SellerAgents - наилучший агент–продавец; Best SellerAgents -лучший агент–продавец; SellerAgents - агент–продавец.

Пример программной реализации МАС на Java.

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

FacilitatorAgent является посредником 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. Рис.4 показывает главную панель программы Marketplace, на основе модели МАС с использованием языка Java. Здесь 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 работы на электронном 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). Одновременно на рынке может размещаться до шести независимых и автономных агентов.

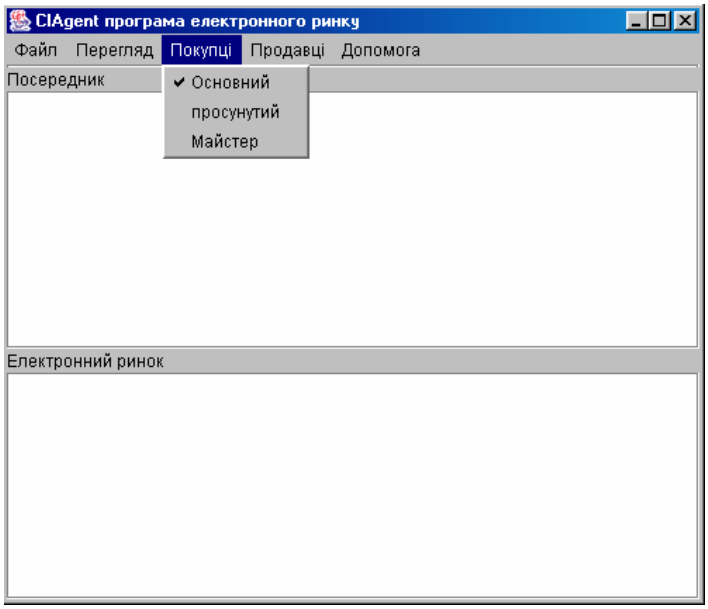

Рис.4 Главная панель программы Marketplace

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 является новым объектом 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. После того, как Посредник зарегистрирует все агенты, которые участвуют в электронном рынке, Торговцы начинают рекламировать свои товары. Следующая процедура BuySellMessages помогает Торговцу и Покупателю обмениваться сообщениями в процессе переговоров на рынке:

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 или *принять* предложение путем пересылки соответствующего предложения обратно к Торговцу или сделать альтернативное предложение Торговцу (указывая другую желаемую цену).
6. Торговец может либо: а) *принять* предложение, б)сделать еще одно альтернативное предложение, в) отбросить 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 (которого иногда называют агентом брокера), как посредника в переговорах о купле-продаже.

Менеджер коммуникаций (BaySelMessage) содержит в себе сообщения, которые должны быть посланы другим агентам, представленные на языке коммуникаций с примитивами типа примитивов языка

KQML: *обратиться с просьбой, принять, отвергнуть, изменить, предложить, проинформировать, запросить данные, отказаться и подтвердить.*

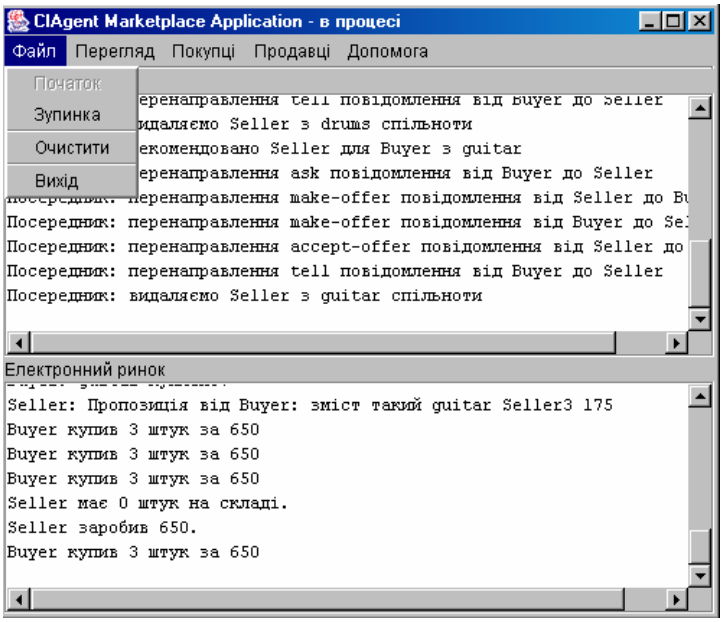

Рис. 5 Окно ведения переговоров между Торговцем и Покупателем через Посредника

Фрагмент кода на Java, описывающего обмен сообщениями между Посредником и Клиентом представлен ниже:

```
startMenuItem_actionPerformed(ActionEvent e) { topTextArea.setText("");
traceTextArea.setText("");
int traceLevel = SUMMARY;
if(detailsCheckBoxMenuItem.isSelected()) { traceLevel = DETAILS;
}facilitator = FacilitatorAgent.getlnstance();
facilitator.reset 0;
facilitator.setTraceLevel(traceLevel);
facilitator.addCIAgentEventListener(це);
facilitator.initialized;
facilitator.startAgentProcessingO;
if(basicSellerCheckBoxMenuItem.isSelectedO) (
   новий SellerAgent0 basicSellerAgent =;
   basicSellerAgent.setTraceLevel();
   basicSellerAgent.addCIAgentEventListener(це);
   basicSellerAgent.initialize();
   basicSellerAgent.startAgentProcessing();
}
if(intermediateSellerCheckBoxMenuItem.isSelect) {
    new BetterSellerAgent() intermedSellerAgent =;
    intermedSellerAgent.setTraceLevel();
    intermedSellerAgent.addCIAgentEventListener(це);
    intermedSellerAgent.initialize();
    intermedSellerAgent.startAgentProcessing();
}
if(advancedSellerCheckBoxMenuItem.isSelectedO) { new BestSellerAgent() advancedSellerAgent =;
advancedSellerAgent.setTraceLevel();
advancedSellerAgent. addCIAgentEventListener ();
advancedSellerAgent.initialize();
advancedSellerAgent.startAgentProcessing();
}if(basicBuyerCheckBoxMenuItem.isSelected()) {
```

```
  new BuyerAgent() basicBuyerAgent =;
  basicBuyerAgent.setTraceLevel(traceLevel);
  basicBuyerAgent.addCIAgentEventListener();
  basicBuyerAgent.initialize();
  basicBuyerAgent.startAgentProcessing();
}
if(intermediateBuyerCheckBoxMenuItem.isSelecte) {
  new BetterBuyerAgent() intermedBuyerAgent =;
  intermedBuyerAgent.setTraceLevel(traceLevel);
  intermedBuyerAgent.addCIAgentEventListener();
```

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 статье рассмотрена программа электронного рынка с использованием семерки CIAgent-базовых “умных” агентов. Главные модули программы следующие:

• *Marketplace программа*, модуль состоящий из единого **FacilitatorAgent,** одного или более **BuyerAgents,** и одного или более **SellerAgents.** Были использованы три типа агентов Клиента и Торговца, которые спользовали (в нарастающем порядке) лучшие стратегии ведения переговоров.

• Все связи между Клиентами и Торговцами проходили 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*”.

• 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.

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, например, для электронного документооборота в корпоративных системах.

Основные применения МАС. Проведенный анализ современных программных продуктов в области МАС, позволил выделить следующие бурно развивающиеся направления их использования:

1. Мобильные вычисления (миграция агентов может поддерживаться не только между постоянно подсоединенными к сети узлами, но и между мобильными платформами, подключаемыми к постоянной сети на некоторые промежутки времени и возможно по низкоскоростным каналам). Клиент подсоединяется к постоянной сети на короткий промежуток времени с мобильной платформы, отправляет агента для выполнения задачи и отсоединяется; затем клиент подсоединяется к другой точке сети и забирает результаты работы агента. Второй вариант - сервер, на который должен переместиться агент, подсоединяется к сети, а затем отсоединяется. В этом случае агент должен уметь переместиться на такой временно подсоединяемый сервер и вернуться в постоянную сеть.
2. Задачи управления информацией:

- Поиск информации (море информации -- один человек не в состоянии найти необходимую ему информацию и проанализировать ее -- использование агента, который странствует по сети в поисках информации, лучше всего удовлетворяющей потребности человека); поисковые агенты содержат сведения о различных информационных источниках (включая тип информации, способ доступа к ней, а также такие характеристики информационного источника, как надежность и точность данных);
- Отбор (обработка) информации. Из всех данных, приходящих к клиенту, выбирают только те данные, которые могут быть интересны клиенту. Используются в комбинации с поисковыми агентами (сначала поиск, затем - отбор);
- Мониторинг данных. Извещение пользователя об изменениях в различных источниках данных в реальном времени (например, мобильный агент перемещается на вычислительный узел, на котором расположен источник данных; это эффективнее, чем использовать статического агента, посылающего запросы источнику данных);
- Универсальный доступ к данным. Агенты - посредники для работы с различными источниками данных, имеющие механизмы для взаимодействия друг с другом (например, агент создает несколько агентов, каждый из которых работает со своим источником данных).

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

Перспективы дальнейших работ.

Реализованная МАС на Java для электронной коммерции требует в дальнейшем проработки вопросов коммуникационной инфраструктуры для движения агентов в сети. Кроме того, необходимо расширить набор функциональных задач электронной коммерции, решаемых МАС и некоторых аспектов архитектуры МАС:

1. На следующем этапе следует 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. Эти различия препятствуют интероперабельности и быстрому внедрению систем мобильных агентов. Необходимо проработать вопросы обеспечения интероперабельности между различными существующими и разрабатываемыми агентскими системами.
3. Интероперабельность достигается при стандартизации таких аспектов, как передача агентов и служебных (используемых агентом) классов (или кода для необъектных систем) между агентскими системами, а также управление агентами. Под управлением агентами подразумеваются функции администратора агентской системы, такие как создание агента заданного класса, его уничтожение, прерывание и возобновление деятельности агента (т. е. соответствующем потоке управления).
4. Необходимо решить задачу безопасности в МАС, в частности, наличие системы защиты от несанкционированного доступа и “плохих кодов”. Это важно по многим причинам. Мобильные агенты, которые приходят из сети, могут таить в себе множество опасностей для принимающей машины, так как они выполняются в ее адресном пространстве. Для обеспечения безопасности перед передачей управления каждому агенту необходимо выполнить процесс авторизации агента, т. е. проверить, зарегистрирован ли он и имеет ли соответствующие полномочия (привилегии), чтобы выполнить то или иное действие или обратиться к некоторым ресурсам. Система безопасности должна предотвращать любые несанкционированные действия агента.
5. Кроме перечисленных выше 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.: Перев. 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.- 715p.
3. Rzevski G. “On Behaviour and Architectures of Autonomous Intelligent Agents: An Engineering Perspective”. // Proceedings First International Round - Table on Abstract Intelligent Agents, ENEA, Rome, 1993.
4. Joseph P. Bigus, Jennifer Bigus Constructing Intelligent Agents Using Java, Second Edition, Addison-Wesly, New York, 2002.-520p.
5. Городецкий В.И., Грушинский М.С., Хабалов А.В. Многоагентные системы (обзор). Санкт-Петербургский Институт Информатики и автоматизации РАН,1999.-74с.
6. Тодорка Атанасова Агентная технология: концепции, модели, приложения , Варна 2000, с.-155.
7. Гладун А.Я., Перевозчикова О.Л., Плескач В.Л. Разработка OSI- профилей открытых систем // “УСИМ”, №6, 1999. С.40-56.
8. Гриценко В.И., Гладун А.Я. Агентно–ориентированные WEB-технологии в интеллектуальных сетях: новые области реализации интеллектуальных услуг// Труды Межд. научно-практической конф. “Современные и будущие информационные технологии Украины”, УкрНИИсвязи, Киев, 15-17 марта 2000 г., с.63-68.
9. Гладун А.Я., Плескач В.Л. Использование агентно–ориентированных технологий в телекоммуникационных сетях// Проблемы программирования, №1, 2000 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 №1(16), 2(17), 2003, Львів, с.28-33.
11. Гладун А.Я., Журавльов Ю.Д., Несен М.В. Реалізація мультиагентної системи для створення інтелектуальних сервісів в розподілених інформаційних системах // Праці Другого Міжнародного Конгресу TRU-2003 „Інформатизація рекреаційної та туристичної діяльності”, 6-9 жовтня 2003 р., Львів-Трускавець, с.56-62.
12. Виттих В.А. Управление открытыми системами на основе интеграции знаний. // Автометрия, №3, 1999, с. 38-49.

В.И.Гриценко, Гладун А.Я., Журавльов Ю.Д., Несен М.В.

МОДЕЛЬ МУЛЬТИАГЕНТНОЇ СИСТЕМИ ДЛЯ Е-БІЗНЕСУ ТА ТЕХНОЛОГІЯ ЇЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Анотація

Покращення ефективності виконання задач в сфері е-бізнесу вимагає подальшого розвитку методів автоматизації ділових процесів. Автоматизація може використовуватись на різних стадіях та в різних областях застосування е-бізнесу. Мультиагентні системи - це новаторські технології, що сприяють автоматизації низки бізнес-процесів: ведення автоматизованих переговорів; довірчість між бізнес-партнерами; виконання задач від імені певного власника; анонімність переговорів і т.і. Авторами запропонована модель мультиагентної системи для вирішення задач е-комерції та її програмна реалізація на Java. У висновках статті поставлені задачі для наступної роботи - побудови комунікаційної інфраструктури інтелектуальних агентів в розподіленому мережному середовищі.

V.I.Gritsenko, A.Y. Gladun, J.D.Shuravlev, M.V. Nesen

MULTI-AGENT SYSTEM MODEL FOR E-BUSINESS AND ITS COMPUTER SOFWARE IMPLEMENTATION TECHNOLOGY

Annotation

Further development of business processes automatization methods is required for improvement of e-business goals productivity. Automatization could be used on different stages and in various applied areas of e-business. Multi-agent systems are innovative technologies that help on a number of business processes automatization: automatic negotiating; achieving confidence between partners; performing tasks in an owner's own name; anonymity of negotiation etc. Authors offer multi-agent system model for e-commerce goals and its Java implementation. Tasks of intelligent agent communicational infrastructure building in distributed environment are set in the article's conclusion.