Информационные системы и технологии

УДК 621.377

Я.В. Бєлецький, В.П. Корнєв, канд. техн. наук

Програмна реалізація мультиагентної концепції – засіб удосконалення мережі терміналів для ведення електронної комерції

Сформированы требования и предложена модель мультиагентной системы предназначенной для задач электронной коммерции. Разработана агентная система, а также её программная реализация на языке JAVA. Предложена программная реализация электронного рынка с использованием базовых интеллектуальных агентов. Описан алгоритм ведения переговоров сбыта, который предоставляет полный контроль над процессом продаж. Выделены основные направления использования мультиагентных систем в электронной коммерции.

Formed requirements and proposed a model of multi-agent system for electronic commerce problems. Agent system is developed, as well as its software implementation in the language JAVA. Propose a software implementation of an electronic market with the base of intelligent agents. We describe the algorithm for negotiations of sales, which provide full control over the sales process. Allocate the basic directions of use of multi-agent systems in electronic commerce.

Ключевые слова: *мультиагентная система, электронная коммерция, агентно-ориентированная технология, распределённая система, электронный документооборот.*

Вступ

Розвиток Інтернет призвів до появи великої кількості проектів, з надання кінцевому споживачеві різних послуг, внаслідок чого має місце питання способу оплати. Як найзручніший спосіб були запропоновані електронні платіжні системи, які на сьогодні, по типу доступу до електронного рахунку можна розділити на дві групи: системи, що потребують установки клієнтського програмного забезпечення на персональний комп'ютер користувача, і системи з WEB-інтерфейсом. Активно розвивається напрямок програмно-апаратних засобів самообслуговування (термінали на базі інформаційних кіосків, мобільні платіжні термінали, прайс-чекери і т. д.). При цьому мобільні термінали мають ряд переваг:

- ідеальне рішення для прийому платежів у роздрібній торгівлі (мережі кіосків, магазинів, кафе, АЗС і т. д.);
- порівняно невеликий обсяг необхідних первинних інвестицій для організації мережі;
- компактність, прийнятна швидкість розгортання;
- як наслідок – скорочення вартості обслуговування;
- відсутність орендної плати.

В умовах пошуку оптимальних шляхів інформатизації суспільства й входження України у світовий інформаційний простір першочергове значення має рішення багатоаспектної проблеми автоматизації сучасних бізнес-процесів: корпоративного електронного документообігу, електронної біржі, електронного ринку, електронного магазину, електронного аукціону та ін. Програмні модулі цих електронних технологій входять до складу інтегральних корпоративних систем і вирішують складні завдання по автоматизації і оптимізації бізнес-процесів. Головна перевага при впровадженні систем електронного бізнесу – це скорочення вартості і швидкість реалізації ділових процесів, а також надання більше зручних послуг клієнтам. Однак, під час використовування систем електронного бізнесу, нині відчувається недостатність реальної автоматизації багатьох завдань.

Для рішення цих завдань можливе застосування агентно-орієнтованої технології (АОТ), що базується на використанні інтелектуальних програмних агентів і дозволяє збільшити функціональні можливості сучасних розподілених систем [1–4].

На рис.1 наведена схема взаємодії сучасних технологій: АОТ, WWW і додатків е-бізнесу, та узагальнена архітектура інтелектуального агента [1–3], що містить у собі головні компоненти: забезпечення по самоврядуванню; комплекс цілей; сховище для даних; компонента по безпеці й компонента зв'язку для взаємодії з іншими агентами, ресурсами системи й користувачами. Щоб бути застосованими мобільні агенти повинні зв'язуватися з різними комп'ютерами, агентами й рухатися усередині різнорідних мереж. Це вимагає реалізації стандартизованого каркаса й методології для дій агента через телекомунікаційні мережі.

За своїм характером АОТ є новою парадигмою програмування, що розширює можливості об'єктно-орієнтованого програмування. Дослідженнями в цій галузі займаються вчені різних країн, зокрема – проф. В.Г. Городецький (Росія,

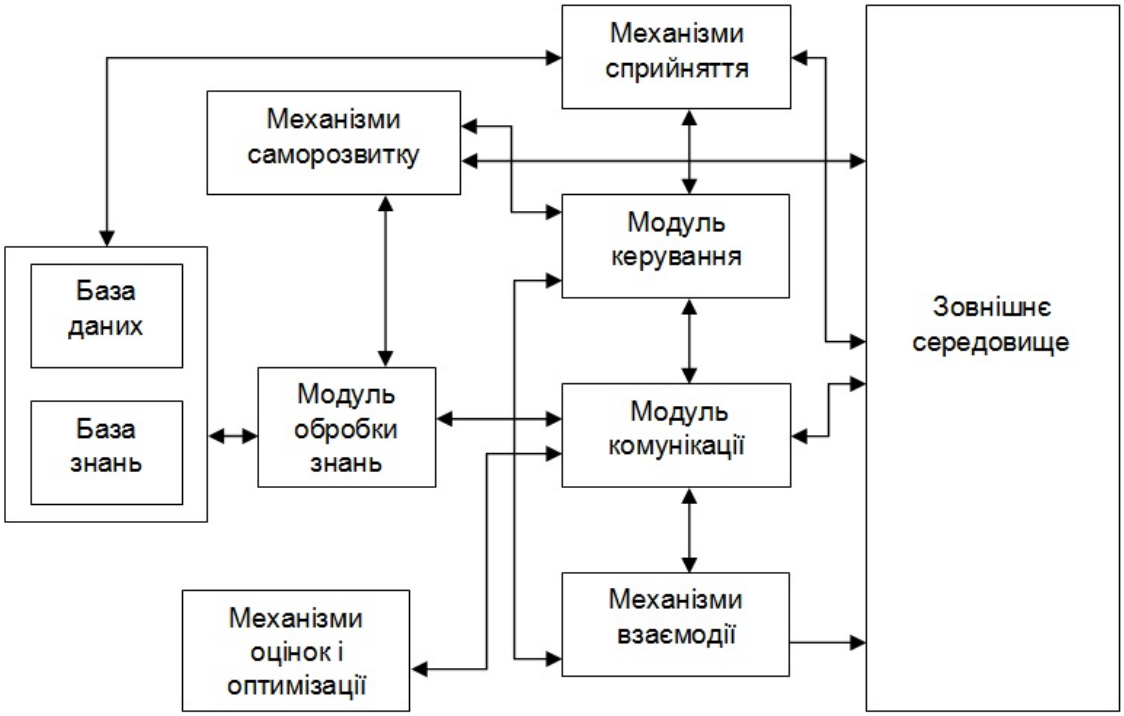

Рис. 1. Узагальнена архітектура мобільного інтелектуального агента

Санкт-Петербург) [2–3], учені США, Німеччини, Франції, Японії, а також провідні світові корпорації виробники програмного забезпечення (Microsoft, Motorolla, ForeSystems, Siemens, Fujitsu й ін.) [1–3]. Кожен такий продукт містить певне «ноу-хау», що визначає ефективність роботи агентів у розподіленій системі.

Слово агент походить від латинського agere – вести, діяти. Головна особливість агентів – здатність виконувати якусь делеговану йому роботу. У цьому вони схожі з роботами. Терміни «програмний агент» або більше загальний «інтелектуальний (розумний) агент» стають близькішими як для розробника програмного забезпечення, так і для користувача. Тому в багатьох бізнес-додатках актуальним завданням стає використання програмних агентів, здатних сприймати ситуацію, рухатися в розподілених системах, мережах (бути мобільними) і приймати рішення. Класифікація агентів відповідно до їх архітектури наведена в роботі [1].

Мобільні агенти не замінюють собою програмного забезпечення (ПЗ) *middleware*, але істотно розширюють його можливості. Насамперед, сама суть агентів передбачає асинхронний зв'язок. Наприклад, агент може зібрати кілька запитів користувача і вже самостійно передати їх для обробки на сервер бази даних. При цьому усувається необхідність у встановленні зв'язку з сервером для обробки кожного запиту. Крім того, під час роботи з мобільними агентами клієнтові не потрібно знати, що за операційна система або яка база даних установлена на тому чи іншому сервері, всі функції організації зв'язку агент бере на себе.

Мобільним агентом є агент, що має здібності до переміщення в розподіленому інформаційному середовищі [1–2] і який має наступні властивості:

- автономність – агенти можуть виконувати свої завдання без безпосереднього втручання клієнтів або інших агентів;
- взаємодія – у разі виникненні потреби агенти взаємодіють із іншими агентами або людьми з метою одержання або надання допомоги в рішенні завдання;
- реактивність – агенти реагують на зміни середовища в реальному часі, зазвичай їхня діяльність описується в такий спосіб:

  WHEN event IF condition THEN action;
- проактивність – здатність вирішувати завдання (досягати мети); на відміну від реактивних агентів, вони не просто реагують на зміни середовища, але й самі опитують;
- здатність до існування, якщо постійно виконується процес, мати власний потік керування;
- гнучкість дії агентів не фіксована жорстко;
- інтелектуальність – здатність знаходити нові рішення (можуть змінювати свою поведінку, використовуючи як свій досвід, так і досвід інших агентів).

В роботі досліджені питання щодо розробки базової мультиагентної системи на мові JAVA для завдань електронної комерції.

Для реалізації модуля прийняття рішень, що входить до складу МАС, використано апарат нечітких множин *(fuzzy aproach)* – сполучення адитивного й мультиплікативного критеріїв прийняття рішень [3, 5]. Крім того, дотримано обмежень, згідно з якими перші три властивості агента є обов'язковими, а інші – необов'язковими.

1. Мультиагентна система для електронної комерції

Ключовим елементом МАС є програмний агент, здатний сприймати ситуацію, приймати рішення й бути комунікабельним з іншими агентами цієї системи. Ці нові можливості значно відрізняють МАС від існуючих жорстко організованих систем. При цьому окремі модулі системи одержують можливість домовлятися про те, як повинно вирішуватися завдання. Ці модулі ініціюють діалог з користувачем і повинні працювати в умовах невизначеності й пропонувати уточнення і переформулювання завдань.

Суть проблеми полягає в тому, що, незважаючи на значний прогрес в галузі теоретичних досліджень МАС, нових можливостей виявилося недостатньо для їх створення. Для розробки дійсно складних відкритих МАС (ВМАС), такі системи повинні постійно існувати на сервері підприємства й безупинно брати участь у рішенні завдань, а не запускатися час від часу. Крім того, відомі на цей час МАС поки в основному орієнтовані на застосування тільки галузі електронної комерції і пошуку в Інтернет, не мають можливостей подання й використання корпоративних знань, складні в розробці, не мають у своєму розпорядженні необхідних інструментальних систем, не забезпечують великої кількості агентів і високої швидкості роботи тощо.

Тому основним завданням була спроба реалізації простої МАС і формулювання вимог до неї та надання розробнику практичної методики реалізації МАС на JAVA. На базі цієї методики показані етапи практичної реалізації МАС для електронної комерції та розроблено узагальнений алгоритм побудови такої системи. Аналіз робіт стосовно МАС дозволив виділити такі основні напрямки досліджень у цій галузі [1–3]:

- теорія агентів, у якій розглядаються формалізми й математичні методи для описання бажаних властивостей агентів;
- методи кооперації агентів (організації кооперативного поводження) у процесі спільного рішення задач, або при інших варіантах взаємодії;
- архітектура агентів і галузь досліджень, у якій вивчається, як побудувати інформаційну систему, що задовольняла б тим чи іншим властивостям, вираженим засобами теорії агентів;
- мови програмування агентів;
- методи, мови й засоби комунікації агентів;
- методи й програмні засоби підтримки мобільності агентів (міграції агентів по мережі).

Особливе місце займають дослідження, пов'язані з розробкою додатків МАС і інструментальних засобів підтримки технології їхньої розробки.

Вибираючи архітектури МАС, варто враховувати два її аспекти:

- архітектуру, що підтримує методи взаємодії агентів у процесі функціонування системи в цілому;
- архітектуру окремого агента.

2. Розподілена платформа системи мобільних агентів

Розглянемо основні технічні передумови створення МАС**.** При використанні такої системи на кожному обчислювальному вузлі повинен бути сервер (далі – агентська система), тобто платформа, що надає функціональність для створення (знищення), прийому (передачі) а також середовище для виконання агентів. Для організації взаємодії агентів повинен існувати сервіс іменування, тобто сервіс, що надає можливість працювати із сутністю, знаючи тільки її ім'я. Функції цього сервісу виконує агентська система.

Агентська система може мати свого власника, та має певний тип, тобто агенти, керовані системою, мають деякий профіль (наприклад, у поняття профілю агента входять: мова реалізації агента, виробник, алгоритм серіалізації агента).

Агентська система повинна підтримувати поняття місця. У цьому разі мережа являє собою набір місць, що надають сервіси. З іншого боку, місце є контейнером і фабрикою агентів. Агент породжується в місці й гине в місці. Протягом життєвого циклу агент переміщується між місцями. Сервіси, що надаються місцем, використовуються агентами, що перебувають у ньому. Місце може накладати обмеження на ресурси, які використовують агенти, що перебувають у ньому. Місце може існувати як на боці клієнта, так і на боці сервера та може зберігати сліди тих агентів, що його відвідували. Агент може містити історію власної життєдіяльності, може використовуватися для перевірки поведінки іншого агента (наприклад, для перевірки обмежень безпе-

ки) і для навчання його на власному досвіді. У випадку, якщо поняття місця не підтримується, його функції переймає власне агентська система.

3. Функціональні завдання електронної комерції

У системі е-комерції можна виділити дві взаємозалежні підсистеми: торгівлі й керування діалогом. Торгівельна підсистема може бути представлена агентами товарів і замовлень, агентами продавців і покупців, агентами складу, постачальників та ін. Агенти товарів і замовлень ведуть переговори щодо знижок постійним покупцям, знижок за оптову покупку, знижок по затовареню складу та ін. При цьому декілька агентів покупців (потенційних конкурентів) можуть об'єднати свої замовлення для одержання більшої знижки, тобто перейти від конкуренції до кооперації.

4. Вимоги до реалізації агентських систем

На основі аналізу відомих МАС [1] виділено дві системи: MADAE – Multi-Agent Engine for E-business Applications, та AE – Multi-Agent Engine for Web-Based Applications; призначені для побудови ВМАС у мережі Інтернет. Аналіз аналогів дозволив сформулювати наступні вимоги до проектованої МАС:

1. Забезпечення переносу коду на різні платформи. Поняття мобільності нерозривно пов'язане з поняттям переносу. Перенос коду можна забезпечити двома способами: за допомогою використання інтерпретуємих мов (Perl, Tcl та ін.); за допомогою використання одної з мов, що підтримують розподільну компіляцію (Java, Oberon та ін.).

2. Доступність на безлічі платформ. Ця вимога є продовженням попередньої. Мобільні агенти повинні здійснювати свою роботу в гетерогенному комп'ютерному середовищі.

3. Підтримка мережної взаємодії. Крім операцій безпосередньо пов'язаних з переміщенням між агентськими серверами, агент повинен мати засоби для комунікації з іншими агентами й доступу до вилучених ресурсів.

4. Багатопоточна обробка. Для реалізації одночасного виконання декількох дій агентська система повинна містити в собі підтримку паралельного виконання функцій агента й підтримку засобів синхронізації.

5. Безпека. Мобільні агенти, що надходять із мережі, можуть містити потенційно небезпечний, шкідливий код. Тому система повинна підтримувати засоби безпеки, достатні для її нормальної роботи.

5. Обґрунтування використання JAVA при реалізації мультиагентної системи

Технологія JAVA надає відкриту, стандартну, універсальну платформу для мережних обчислень. Під час розроблення особливий акцент було направлено на незалежність додатків JAVA від конкретної апаратно-програмної платформи, що і дозволяє успішно обмінюватися в гетерогенному обчислювальному середовищі додатками й навіть їхніми фрагментами. Це було досягнено за допомогою віртуальної JAVA-машини, у коди якої транслюються JAVA-додатки.

Програма, написана на JAVA, компілюється в спеціальний машинно-незалежний байт-код. Потім цей код може бути виконаний за допомогою інтерпретатора JAVA на будь-якому комп'ютері, де реалізована JAVA Virtual Machine. JAVA – об’єктно-орієнтована мова програмування. Кожен тип даних це клас об'єктів, будь-яка функція є методом класу. Її виклик розглядають з об’єктно-орієнтованих позицій як посилку повідомлення об'єкту. В JAVA є вбудована розширювана бібліотека класів, що включає модулі керування вікнами (AWT – Abstract Window Toolkit) для створення користувальницьких інтерфейсів, класів підтримки основних типів даних, багатопотоковості мережних можливостей, графіків, мультимедіа тощо.

Мова JAVA обрана також завдяки її інтуїтивній зрозумілості, тобто класи й методи мають такі назви, що вихідний текст часто не вимагає коментарів, а сам проект програмного агента можна досить зручно вкласти в один архів jar-архів і незалежно від кількості файлів у цьому проекті запускати програму агента з командного рядка java – jar agent.jar, не маючи проблем з типом операційної системи – UNIX, Windows, Solaris та ін. [5].

6. Послідовний алгоритм реалізації мультиагентної системи

Практичну реалізацію МАС пропонується виконувати за такими кроками:

1. Визначення функціональних задач, які повинна виконувати реалізована МАС (перелік задач наведено в роботах [3, 5]).

2. Вибір мови програмування, на якій буде реалізований агент. Мова програмування в сучасному інформаційно-телекомунікаційному середовищі повинна підтримувати функції багато потоковості, для можливості одночасної паралельної обробки багатьох задач. Багатопотокова

обробка означає, що агент може виконувати деякі дії одночасно. Тим самим мова програмування агентів повинна включати підтримку паралельного виконання різних функцій агента (типу *threads*) і різних примітивів синхронізації (семафори, монітори, критичні секції тощо). Мова повинна бути інтуїтивно зрозумілою та одночасно інтернет-орієнтованою, а також передбачати запуск потоків-демонів, що працюють навіть після завершення основної частини програми. Як відомо, всі ці якості мають дві мови – .NET розширення для C++ та JAVA. Інтернет-орієнтованість мови означає її платформонезалежність (доступність на багатьох платформах). Ця вимога безпосередньо випливає з попередньої, тому що інтелектуальні агенти повинні працювати в гетерогенному комп'ютерному середовищі [6, 7].

3. Створення моделі МАС і проектування функціональних логічних модулів з яких буде складатися система.

4. Програмна реалізація. Кожного агента реалізовано окремим файлом. Користувач буде запускати агентну підпрограму з модуля інтерфейсу, а вже після цього здійснюється ініціалізація головного модуля. Під час реалізації МАС необхідно врахувати те, щоб код був правильно структурований і була можливість швидкої зміни його якщо є потреба.

5. Тестування програмної реалізації МАС. Ця фаза може тривати протягом всього періоду експлуатації й модернізації системи. Тестування повинне включати можливі несподівані дії з боку користувача й відповідну реакцію програми на ці дії.

6. Налагодження. У цій фазі реалізації МАС у текст програми вносять зміни відповідно до виявлених недоліків.

7. Експлуатація. Цю фазу можна виконувати разом з фазою тестування у разі виникнення непередбачених ситуацій.

7. Запропонована модель мультиагентної системи

Для розробки логічної моделі МАС зручно скористатися засобами мови UML. При цьому дуже важливо врахувати особливості структури системи. Каркас МАС повинен включати такі модулі: інтерфейсу; функцій для взаємодії з користувачем (обробник подій); головний модуль координації й керування (відповідно до поставленого завдання) і модуль додаткових функцій для роботи з даними (сортування, фільтрація, пошук тощо); повернення результатів користувачеві (у вигляді log-файлу – повідомлень на інтерфейс користувача). З урахуванням класів і методів мови JAVA, отримано модель МАС, що проілюстрована на рис. 2.

Модель МАС містить такі модулі: Offer – визначення ціни товару на торгах; Basic Negotiation – ведення базових переговорів (правила); AboutDialog – ведення діалогу (інтерфейс із користувачем); Marketplace Frame – блок координації й керування МАС; BaySel Message – інтерфейс взаємодії з користувачем; FacilitatorAgent – агент посередник; Better BayerAgent – найкращий агент-покупець; Best BayerAgent – кращий агент-покупець; BayerAgent – агент-покупець; Better SellerAgent – найкращий агент-продавець; Best SellerAgent – кращий агент-продавець; SellerAgent – агент-продавець.

Рис. 2. UML-модель мультиагентної системи для задач е-комерції

8. Програмна реалізація мультиагентної системи на JAVA

На основі мови програмування JAVA, розроблено FacilitatorAgent (Агент-посередник), що управляє електронним ринком, а також агенти BayerAgent (агент-покупець) і SellerAgent (агент-продавець), які використані для взаємодії усередині ринку.

Всі ці інтелектуальні агенти отримано з базового класу CIAgent, що описано в роботах [2–4].

Агенти Клієнта і Продавця (Торговця) розрізняються, насамперед, складністю їхніх стратегій переговорів. Переговори починаються з простої логіки (у термінах if-then-else), а потім переходять до методів формування правил, які базуються на отриманих фактах.

FacilitatorAgent є посередником між Клієнтами і Торговцями. Всі агенти зобов'язані зареєструватися в Посередника перед тим, як вони почнуть взаємодіяти з якими-небудь іншими агентами на ринку (marketplace). Торговці рекламують бажання продати товари або послуги за допомогою Посередника, у той час як, Клієнти також просять Посередника рекомендувати їм імовірного продавця. Як тільки агенти Клієнта і Торговця будуть представлені Посередником, вони продовжуватимуть спілкуватися тільки через нього.

Меню Клієнта й Торговця забезпечують три типи Клієнтів і Торговців, для роботи на електронному ринку. Ці агенти можуть бути обрані в будь-якій комбінації Основних, Середніх і Просунутих Клієнтів.

Одночасно на ринку може розміщатися до шести незалежних й автономних агентів.

Всі агенти запускають власні потоки й активуються через конкретні фіксовані проміжки. Всі переговори між Клієнтами, Торговцями й Посередником відбуваються через інтерфейс CIAgent-Events і CIAgentEventListener. Об'єкт аргументу, що передається з CIAgentEvents, є новим об'єктом за назвою Bay SellMeswisard, що формується за зразком стандартного повідомлення KQML.

9. Алгоритм ведення переговорів між агентами в мультиагентній системі

У KQML конкретизується формат і деякий зміст взаємодій між Торговцем і Клієнтом. Нижче розглянуто алгоритм ведення переговорів збуту, що використовується в програмі.

Після того, як Посередник зареєструє всіх агентів, які беруть участь в електронному ринку, Торговці починають рекламувати свої товари. Наступна процедура Bay SellMessage допомагає Торговцеві і Покупцеві обмінюватися повідомленнями в процесі переговорів на ринку:

- Клієнт просить Посередника відрекомендувати йому одного з Торговців для покупки товару.
- Посередник говорить Клієнтові ім'я Торговця.
- Клієнт запитує Торговця (через Посередника), чи має Торговець товар для продажу.
- Торговець або відгукується на пропозицію співробітництва з Клієнтом (повідомляючи про товар, унікальним ідентифікатором товару (ID) або ж дає негативну відповідь, указуючи, що товару для продажу немає.
- Клієнт може або прийняти пропозицію шляхом пересилання відповідної пропозиції до Торговця або зробити альтернативну пропозицію Торговцеві (указуючи іншу бажану ціну).
- Торговець може: прийняти пропозицію, зробити ще одну альтернативну пропозицію, відкинути пропозицію.
- На випадок, коли Торговець приймає пропозицію, він відсилає tell-повідомлення Клієнтові. Таким чином договір про збут буде завершений.
- На випадок, коли Торговець відкидає пропозицію, то переговори тривають далі.

Клієнт і Торговець ніколи не спілкуються безпосередньо, а використовують для цього FacilitatorAgent (агент брокера), як посередника в переговорах про купівлю-продаж.

Менеджер комунікацій (Bay SelMessage) містить у собі повідомлення, які повинні бути надіслані іншим агентам, представлені мовою комунікацій із примітивами типу KQML: звернутися з проханням, прийняти, відкинути, змінити, запропонувати, проінформувати, запросити дані, відмовитися й підтвердити.

Фрагмент коду на JAVA, що описує обмін повідомленнями між Посередником і Клієнтом наведено в додатку А.

Висновки

На підставі сформованих вимог до МАС розроблена агентна система для електронної комерції. Модуль ухвалення рішення в МАС побудовано з використанням теорії нечітких множин (сполучення числового й лінгвістичного підходів). Алгоритм ухвалення рішення дозволив виділити три групи агентів: FacilitatorAgent (Агент-посередник), а також агенти BayerAgent (агент-покупець) і SellerAgent (агент-продавець), що використані для взаємодії усередині ринку.

Розглянуто програму електронного ринку з використанням CiAgent-базових інтелектуальних агентів. Модуль Marketplace складається з одного агента FacilitatorAgent, одного або більше BuyerAgent, і одного або більше SellerAgent. Всі взаємини між Клієнтами й Торговцями мають зв'язки з Посередником на основі використання KQML-подібних об'єктів з назвою Bay SellMessage. Ці повідомлення включають такі KQML-performatives, як наприклад: запис, рекламують, не рекламують, не рекомендують один, не запитують, не говорять.

Описано алгоритм ведення переговорів збуту, що надає більше контролю над процесом продажу. Клас BasicNegotiation представлено для того, щоб інкапсулювати деталі кожного договору. Пропозиція складається з імені агента, імені товару, унікального ID товару, що генерується спочатку переговорів, і ціни пропозиції.

Представлена МАС може використатися як для моделювання ситуацій, пов'язаних з ринком, так і для розробки готового програмного продукту не тільки для електронної комерції, але й для інших бізнес-задач, наприклад, для електронного документообігу в корпоративних системах.

Прикладами використання агентів може бути пошук інформації (data mining-агенти взаємодіють із серверами баз даних і сховищами даних), електронна комерція.

Реалізована МАС на JAVA для електронної комерції вимагає подальшої проробки питань комунікаційної інфраструктури для руху агентів у мережі. Крім того, необхідно розширити набір функціональних завдань електронної комерції, розв'язуваних МАС і деяких аспектів архітектури МАС:

1. На наступному етапі варто стандартизувати дії агентських систем при перетинанні агентом декількох доменів безпеки, а також формат подання коду й стану агента при його переміщенні між агентськими системами різних типів.

2. Технологія мобільних агентів досить нова, тому системи програмування мобільних агентів істотно розрізняються по архітектурі й реалізації. Ці розходження перешкоджають швидкому впровадженню систем мобільних агентів.

3. Інтероперабельність досягається при стандартизації таких аспектів як передача агентів і службових класів між агентськими системами, а також керування агентами.

4. Необхідно вирішити завдання безпеки в МАС, зокрема, наявність системи захисту від несанкціонованого доступу. Система безпеки повинна запобігати будь-які несанкціоновані дії агента.

5. Крім перерахованих вище аспектів, необхідна також стандартизація синтаксису й семантики різних параметрів, наприклад імен агентів й агентських систем, типів агентських систем.

Гнучкість агентного підходу дозволяє реалізовувати й впроваджувати нову функціональність системи мережі мобільних терміналів для електронної комерції, а також через модульну й розподілену природу удосконалювати та модернізувати її з набагато меншими витратами [7].

Як приклади розширення функціональності можна привести такі напрямки, як он лайн торгівля і прийом платежів.

Отримані результати дають підставу сподіватися на успішне вирішення в майбутньому цих важливих задач.

Література

1. *Атанасова Т.А.* Агентная технология: концепции, модели, приложения. – М.: Варна, 2000. – 155 с.
2. *Гладун А.Я.,* Перевозчикова О.Л., Плескач В.Л. Разработка OSI-профилей открытых систем. – К.: УСИМ, 1999. – С. 40–56.
3. *Городецкий В.И.,* Грушинский М.С., Хабалов А.В. Многоагентные системы. – СПб.: Санкт-Петербургский ИИА РАН, 1999. – 74 с.
4. *Рогоза В.С.,* Іщенко Г.В. Інтелектуальні платформи розподілених інформаційних середовищ – К.: АВЕРС, 2009. – 350 с.
5. *Юрасов А.В.* Основы электронной коммерции: Учебник для вузов. – М.: Телеком, 2008. – 480 с.
6. *Рассел С,* Норвиг П. Искусственный интеллект: современный подход: Пер. с англ. – 2-е изд. – М.: Издательский дом "Вильямс, 2006. – 1408 с.
7. *Люгер Джордж Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем: Пер. с англ. – 4-е изд. – М.: Издательский дом «Вильямс», 2003. – 864 с.

Национальный технический университет Украины «Киевский политехнический институт»

Поступила в редакцию 31 мая 2011 г.

Додаток А

Фрагмент коду, що описує обмін повідомленнями між Посередником і Клієнтом

```
startMenuItem_actionPerformed(ActionEvent e)
{
    topTextArea.setText("");
    traceTextArea.setText("");
    int traceLevel = SUMMARY;
    if(detailsCheckBoxMenuItem.isSelected())
    {
        traceLevel = DETAILS;
    }
    facilitator = FacilitatorAgent.getInstance();
    facilitator.reset 0;
    facilitator.setTraceLevel(traceLevel);
    facilitator.addCIAgentEventListener(це);
    facilitator.initialized;
    facilitator.startAgentProcessingO;
    if(basicSellerCheckBoxMenuItem.isSelectedO) (
    new 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(basicBayerCheckBoxMenuItem.isSelected())
    {
         new BayerAgent() basicBuyerAgent =;
         basicBayerAgent.setTraceLevel(traceLevel);
         basicBayerAgent.addCIAgentEventListener();
         basicBayerAgent.initialize();
         basicBayerAgent.startAgentProcessing();
    }
    if(intermediateBuyerCheckBoxMenuItem.isSelecte)
    {
         new BetterBayerAgent() intermedBuyerAgent =;
         intermedBayerAgent.setTraceLevel(traceLevel);
         intermedBayerAgent.addCIAgentEventListener();
     }
```