УДК 004.4'2

Глибовець А. М.

ПРОТОТИП ПРОГРАМНОЇ СИСТЕМИ ПІДТРИМКИ ДИСТАНЦІЙНОГО НАВЧАННЯ НА БАЗІ АГЕНТНИХ ТЕХНОЛОГІЙ

Описано прототип програмної системи підтримки електронної освіти, яка є базою для впровадження підсистеми підтримки мобільного навчання, з використанням агентного підходу. Проектування велося за допомогою Visual Paradigm, UML (Unified Modeling language). Модель було створено на основі спеціалізованого інструментарію JADE (Java Agent Development Environment).

Вступ

Загальносвітові процеси глобалізації мають сприяти вільному доступу до інформаційних ресурсів. Зростаючий ступінь інтерактивності приводить до того, що на перше місце в навчальному процесі виходять інформативність, комунікативність та мобільність. Адміністративна інформаційна система не може залишатися додатком до баз даних, для неї теж необхідні персоніфіковані інтелектуальні інтерфейси, електронний обмін даними, сценарії виконання рутинних справ, інтелектуальна підтримка прийняття рішень, а також нові ефективні процедури пошуку, створення звітів і аналізу тенденцій розвитку навчального закладу. На перший план виходять також можливість доступу до навчальних матеріалів у будь-якій точці земної кулі. Це стає можливим завдяки стрімкому поширенню технологій мобільного зв'язку.

Сучасні інформаційні мережні технології дають можливість не просто перевести навчальний процес у цифровий режим або замінити навчальну аудиторію, викладача та підручник комп'ютером - вони дозволяють змінити філософію навчального процесу, створити нову навчальну культуру. Дистанційна освіта перейшла від традиційної системи передачі знань, побудованої навколо викладача, до віртуального навчального середовища й навчальної громади, орієнтованих на студента. В останні роки до вже звичної дистанційної освіти почали залучати сучасні мережі та стандарти стільникового зв'язку - GSM, GPRS, EDGE, UMTS. Така освіта називається мобільною (m-learning) [1-3].

На наш погляд, найбільш придатним визначенням мобільної освіти буде таке. Мобільною освітою можна вважати електронну освіту, в якій комунікативна взаємодія між клієнтом і комп'ютерною системою підтримки електронної освіти забезпечується мобільними пристроями та засобами бездротового зв'язку. Основними вимогами до мобільних пристроїв є здатність з'єднуватися з комп'ютерними пристроями, відображувати освітню інформацію й мати можливість реалізовувати двосторонній інформаційний обмін між викладачем і студентом [2].

Опишемо розроблений нами прототип програмної системи підтримки електронної освіти, яка є базою для впровадження підсистеми підтримки мобільного навчання, з використанням агентного підходу. Проектування велося за допомогою Visual Paradigm, UML (Unified Modeling language). Модель створена на основі спеціалізованого інструментарію JADE (Java Agent Development Environment) [4].

Можливості JADE

JADE пропонує програмісту-розробнику агентних систем такі можливості.

FIPA-compliant Agent Platform - агентну платформу, що базується на FIPA і включає обов'язкові типи системних агентів: AMS, ACC та DF. Ці три типи агентів автоматично активуються при запуску платформи.

Distributed Agent Platform - агентну платформу, яка є розподіленою та може використовувати декілька хостів, і тільки одна Java Virtual Machine запускається на кожному вузлі. Агенти імплементуються як Java-треди.

Multiple Domains support - ряд базованих на FIPA DF-агентів можуть об'єднатись у федерацію, таким чином імплементуючи мультидоменне агентне середовище;

Multithreaded execution environment with two-level scheduling. Кожний JADE-агент має власний потік керування, але він також здатен працювати в багатопотоковому режимі. Java Virtual Machin здійснює планування завдань, що виконуються агентами або одним із них.

Object Oriented programming environment. Більшість концепцій, притаманних FIPA-специфікаціям репрезентуються Java-класами, що формують інтерфейс користувача.

Library of interaction protocols. Використовуються стандартні інтерактивні протоколи fipa-request та fipa-contract-net. Для того щоб створити агента, який би міг діяти відповідно до таких протоколів, розробникам застосувань треба тільки імплементувати специфічні доменні дії, у той час як уся незалежна від застосувань протокольна логіка буде здійснюватися системою JADE.

Administration GUI. Прості операції з управління платформою можуть виконуватися через графічний інтерфейс, що відображає активних агентів та контейнери агентів. Використовуючи GUI, адміністратори платформи можуть створювати, знищувати, переривати та відновлювати дії агентів, створювати ієрархії доменів та мультиагентні федерації DF (фасилітаторів).

Модель прототипу

Щоб було зрозуміліше, зафіксуємо такі складові системи (рис. 1): *Student* - комп'ютер студента, що використовує мобільного агента для навчання; *Task Journal* - журнал викладача, що використовує сервер (мобільного агента) для перевірки та надання нових завдань; *Internet* - група пристроїв, що надає доступ до глобальної мережі Інтернет; *Local Network* - пристрої локальної мережі, що забезпечують зв'язок серверів; *File Server* - файловий сервер, що зберігає всі навчальні матеріали та контрольні роботи; *Informatorium* - довідковий агент, в якому зберігається інформація про курси, дати проведення екзаменів та інша довідкова інформація.

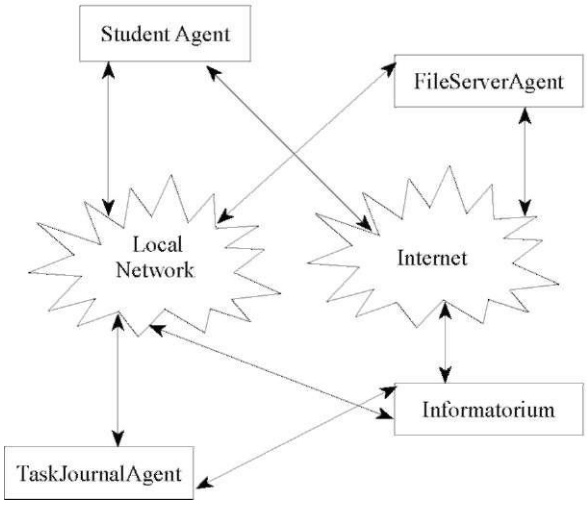

Рис. 1. Структура прототипу

Спочатку опишемо схему (рис. 2) нашої системи з допомогою UML.

Де:

FileServerAgent моделює сховище типу content adressed storage. За такого методу зберігання немає файлової системи, а в кожного елемента зберігання є свій унікальний ID;

StudentAgent посилає запити, чи перевірена його робота, та чекає, поки йому не надійдуть відповідь і результати;

FetchResultBehaviour дозволяє агенту отримати результати перевірки певного завдання від агента FileServerAgent. Метод-конструктор, що приймає на вхід ідентифікатор задачі, результат перевірки якої необхідно отримати, та батьківський агент;

CheckTaskBehaviour дозволяє агенту почати моніторинг статусу перевірки певного завдання та, дочекавшись позитивної відповіді, отримати результат перевірки від FileServerAgent. Метод-конструктор, що приймає на вхід ідентифікатор завдання, результат якого необхідно дочекатися та отримати, батьківський агент та інтервал, через який мають повторюватися перевірки;

TaskRequestServer - успадкована від CyclicBehaviour функція, що кожен раз викликається для даної поведінки;

TaskJournalAgent відзначає перевірку роботи. Викладач не знає, кому належить робота, яку він перевіряє. Студент знає шифр своєї роботи й за ним запитує результати перевірки;

Agent базовий клас для всіх агентів, належить до каркаса JADE;

Informatorium - довідковий агент.

На рис. 3 представлено схему, що описує взаємодію в системі.

Пакет Fileserver складається з Behaviour ContentRequestServer та агента FileServerAgent. Моделює сховище типу content adress storage. За такого методу зберігання немає файлової системи, а в кожного елемента зберігання є свій унікальний ID.

FileServerAgent стартує поведінку ContentRequestServer, яка приймає повідомлення та відповідає на них.

Пакет Student складається з Behaviour CheckTaskBehaviour, Behaviour FetchResultBehaviour та агента StudentAgent. Студент (StudentAgent) посилає запити на перевірку його роботи й чекає, поки йому не надійде відповідь і результати. FetchResultBehaviour висилає запит до Fileserver, у якому вказує, що нам потрібен результат для роботи з унікальним ID (шифр). CheckTaskBehaviour проводить пошук по журналах, чи не з'явилася відповідь на задачу (контрольну).

Функція onTick робить перевірку кожні 5 секунд. Працює у 3 фази:

1) розсилає повідомлення всім відомим журналам і оновлює список журналів агентів (поси-

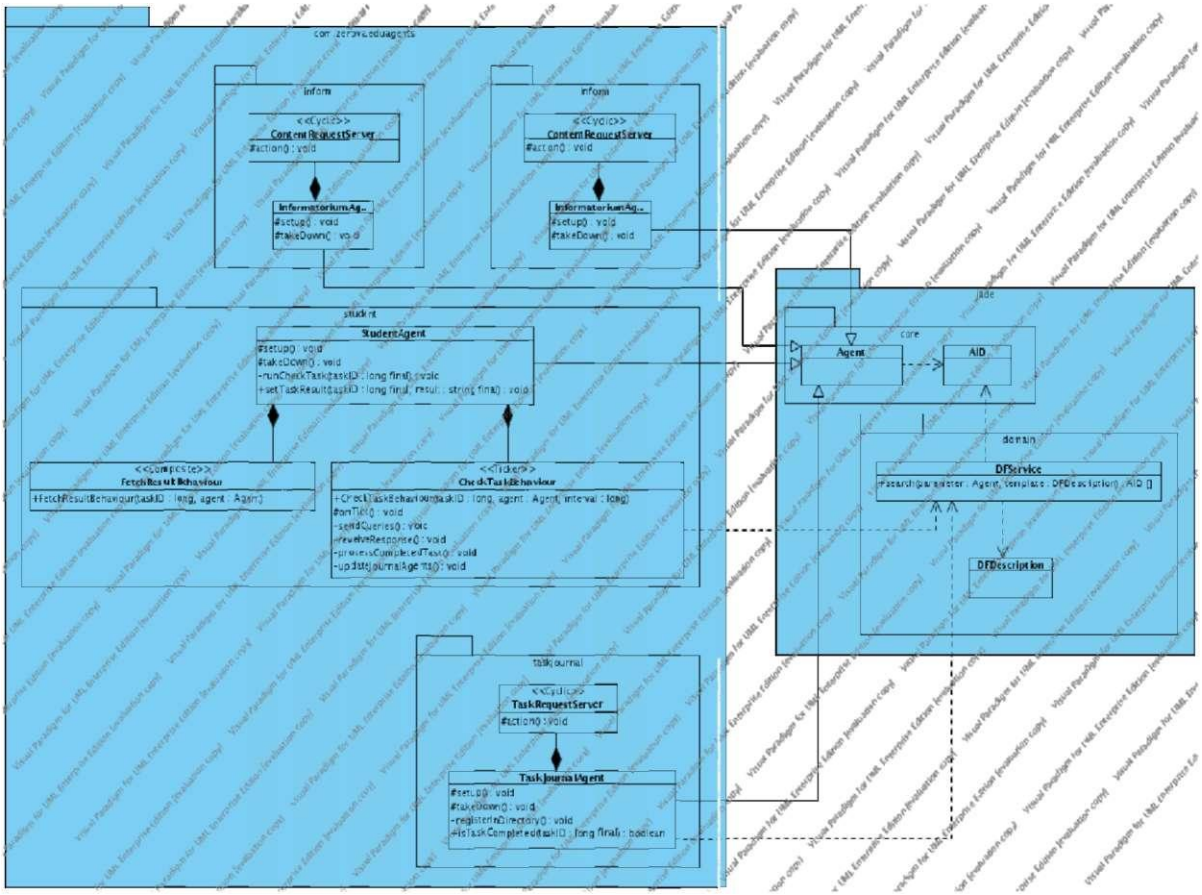

Рис. 2. Схема системи

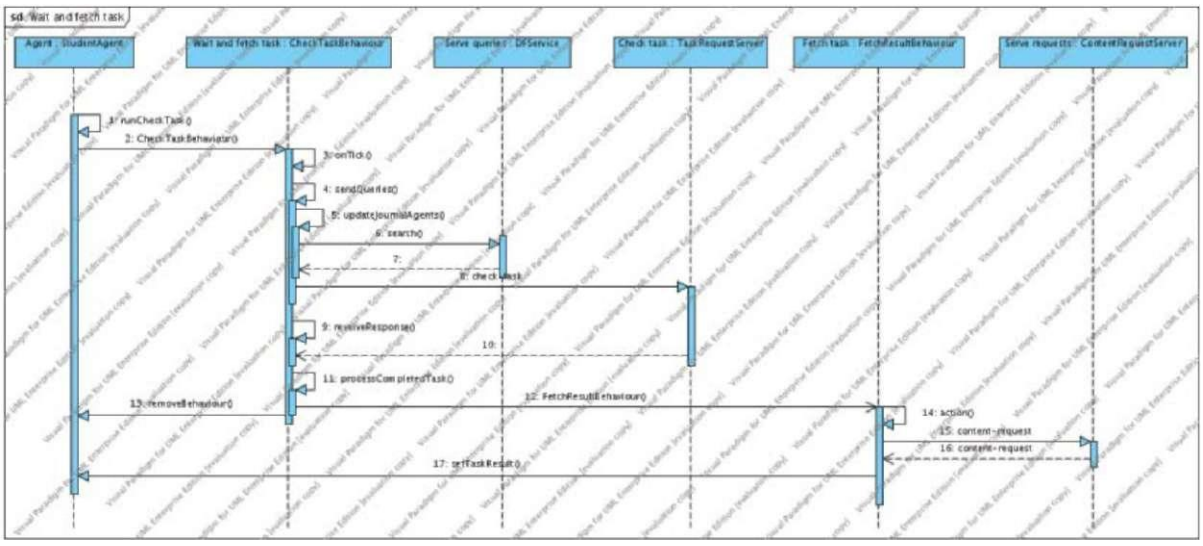

Рис. 3. Діаграма взаємодії (Sequence diagram)

лає запит до DF, які сервіси зареєстрували TaskJournal), посилає запити (queries, якщо журнал зареєстрований), додає до списку receiver усі журнали, які зареєстровані;

2) отримує відповіді. Якщо відповідь позитивна, то переходить до фази 3, якщо негативна, то повертається до фази 1;

3) створюється й додається агенту поведінка FetchResultBehavoiur, після чого CheckTaskBehaviour самознищується.

StudentAgent зчитує із сертифіката користу¬вача ім'я та прізвище студента й чекає, коли прийдуть результати (results). Коли вони з'являються, агент забирає результати з FileServerAgent.

Пакет Taskjournal складається з агента Task-JournalAgent і Behaviour TaskRequestServer. Він відповідає на питання, готове завдання чи ні та з якою оцінкою (у кожної роботи є своє унікальне ID (шифр)).

TaskJournalAgent реєструється в DF і додає до себе поведінку TaskRequestServer. Відзначає, що робота перевірена. Викладач не знає, кому належить робота, яку він перевіряє. Студент знає шифр своєї роботи й по ньому запитує результати перевірки.

TaskRequestServer керує агентом TaskJournal-Agent.

За допомогою функції action отримуємо повідомлення, обробляємо, виділяємо task id. Якщо завдання виконане, то отримуємо повідомлення confirm, якщо ні - отримуємо повідомлення disconfirm.

Пакет Informatorium складається з поведінки ContentRequestServer і агента Informatorium-Agent. Приймає запит, готує відповідь, відсилає відповідь у наступному повідомленні. Довідковий агент зберігає інформацію про курси, дати проведення екзаменів та іншу довідкову інформацію.

InformatoriumAgent.java стартує поведінку ContentRequestServer, що приймає запити користувачів та відповідає на них. Встановлює з' єднання з базою даних, що містить різну довідкову інформацію.

Інформаційна безпека в системі забезпечується за допомогою методів асиметричної криптографії. Асиметрична криптографія передбачає наявність у користувача двох ключів - відкритого та закритого. Ці ключі являють собою довгі числа.

Особливою задачею є створення умов безпечної передачі інформації між учасниками процесу. Вона розв'язується за допомогою сертифікації агентів (стандарт X.509) та іменування агентів (стандарт X.500). Взаємна аутентифікація проводиться за стандартом ISO 9897-3:1993, а мережевий обмін - із використанням протоколу SSL (Secure Socket Layer).

Контейнери keystore та truststore зберігають відкриті та закриті ключі для кожного користувача (у цих файлах міститься інформація, хто й кому видав сертифікат). Keystore-main зберігає ключі для File Server та Informatorium. Для Task Journal та Student - свої Keystore-student truststore-student та keystore-teacher truststore-teacher відповідно. У truststore зберігаються відкриті ключі, які пройшли аутентифікацію (тобто ключі яким ми довіряємо).

У даному програмному комплексі є дві фіксовані бази даних: File Server Data Base (використовується агентом FSA), Informatorium Data Base (використовується агентом Informatorium Agent) До того ж кожен агент TaskJournalAgent володіє окремою БД TaskJournalDB.

Усі БД реалізовані з допомогою HSQLDB (Hypersonic SQL Database), що не потребує створення окремого сервера БД. Підключення до агентної платформи без коректного сертифіката є неможливим.

Порядок старту агентів не визначений, але може бути таким:

- стартує Fileserver;
- стартує Informatorium;
- стартує агент TaskJournalAgent, реєструється в директорії DF та додає до себе поведінку TaskJournalAgent, починає працювати функція onTick (з CheckTaskBehaviour.java);
- стартує агент студента StudentAgent;
- стартує поведінка студента CheckTaskBehaviour, яка повинна отримати результат для завдання 1. Він оновлює список агентів журналів (які перед цим зареєструвалися в DF), виводить повідомлення, скільки знайшов журналів, і розсилає повідомлення всім відомим журналам;
- журнальний агент отримує query if, дивиться, чи перевірена контрольна робота (із допомогою функції isTaskComplete, яку викликає TaskRequestServer), і якщо результати перевірки існують, то розсилає повідомлення confirm, якщо ні, то disconfirm;
- Студент робить крок 2 - отримує повідомлення на запит, який він розіслав у кроці 1. Якщо отримує відповідь confirm, отже, задача 1 завершена, якщо disconfirm, то ні. Результат її виконання можна отримати шляхом додавання FetchResultBehaviour.java до агента, який належить CheckTaskBehaviour.java. Після того як FetchResultBehaviour додано, Check-TaskBehaviour самознищується. FetchResult-Behaviour висилає запит до Fileserver, в якому зазначає, що нам потрібен результат для роботи з унікальним ID (число). Fileserver перевіряє, чи є дані під цим ідентифікатором, і якщо вони є, то формується відповідь confirm performative. У тіло відповіді розміщується контент, який запитував користувач. Student отримує результат і виводить його на екран, після чого Behaviour, який відповідав за отримання результату, самознищується.

Для впровадження програмного комплексу в навчальному закладі потрібно створити сертифікаційний центр, провести генерацію сертифікатів для системних агентів (File Server, Informatorium), розповсюдити сертифікати серед викладачів та студентів і розмістити сертифікати користувачів на спеціалізовані smart карти чи USB ключі (наприклад, Aladdin eToken).

Висновок

Пропонований програмний комплекс не є повною системою підтримки дистанційного навчання, це агентна платформа для такої системи. Вона бере на себе всю безпеку, сумісність із різними мобільними пристроями, мережний обмін повідомленнями, синхронізацію між агентами. Наголосимо, що користуватися даним програмним комплексом можна не тільки з персонального комп'ютера, а і з мобільного телефону. Для побудови була використана платформа JADE LEAP, яка існує в таких модифікаціях: JADE LEAP J2SE - для виконання на ПК та серверах, JADE LEAP PJAVA - для виконання на PDA та SMARTфонів, JADE LEAP mid - для звичайних мобільних телефонів, що підтримують java.

1. *Sharma S., Kitchens F.* Web Services Architecture for M-Learning, EJEL,Vol.2, Issue 1, 2004.
2. The Mobilearn Project Vision. Available at http://www.mobilearn.org/vision/visiton.htm
3. *Глибовець А. М., Міщенко Я. М.* Побудова підсистеми тестування у програмній системі підтримки мобільного навчання // Наукові записки НаУКМа. Комп'ютерні науки. - 2007. - 37 с.
4. *Bellifemine F. L., Caire G. and Greenwood D.* Developing Multi-Agent Systems with JADE, 2007.

A. Glybovets

THE PROTOTYPE OF DISTANCE EDUCATION SYSTEM BASED ON AGENT TECHNOLOGY

A prototype of an agent-oriented software system for support of e-learning was described. This system is the base for building a subsystem for mobile learning support. The design was created using Visual paradigm and UML (Unified Modeling language). The model was built on the basis of a specialized tool - JADE (Java Agent Development Environment).