УДК 519.685

Глибовець М. М.

ІНСТРУМЕНТАРІЙ РОЗРОБКИ АГЕНТНИХ ПЛАТФОРМ

В роботі описано спеціалізований інструментарій створення агентних систем об´єктно-орієнтованого типу у вигляді програмного інтерфейсу. Він представляє собою розвинену бібліотеку класів мови Java, зібраних в один «пакунок».

***Ключові слова**: програмний агент, агентна платформа, спеціалізований інструментарій*

В работе описан специализированых инструментарий разработки агентных объектно-ориентированного типа в виде программного интерфейса. Он представляет развитую библиотеку классов языка Java, собранных в одну «упаковку».

***Ключевые слова:** программный агент, агентная платформа, специализированый инструментарий.*

In this paper the specific tools for development of object-oriented agent systems in form of software interface are described. They consists of well-developed library of Java classes collected in one «package».

***Key words:** program agent, agent platform, specific tool.*

Вступ

Процес створення агентної платформи (АП) без застосування спеціалізованого інструментарію змушує розробника самостійно вирішувати широкий набір проблем – від організації мережної взаємодії до парсінга XML та вибраної мови комунікації. Це значно ускладнює процес розробки і суттєво збільшує час розробки [1].

Для створення агентних систем традиційно використовується різноманітний спеціалізований інструментарій побудови програмних систем. Але, відомі нам спеціальні інструментарії не вирішують у повному обсязі зазначені проблеми. Серед них найважливішими є уникнення постійної взаємодії з бібліотеками мови програмування, операційною системою та задоволення можливості використання у різних обчислювальних системах (уніфікація АП).

Тому, ми вирішили розробити власний спеціалізований інструментарій створення агентних систем у вигляді програмного інтерфейсу. Використання розробленого інструментарію надає такі переваги розробнику АП:

– Розробник уникає необхідності безпосередньо використовувати функції операційної системи та бібліотеки мови програмування – замість цього, розробник використовує функції інструментарію, які є уніфікованими за викликом, і в яких враховані потреби агентної системи.

– При змінах у функціях мови програмування чи операційного середовища, необхідні лише зміни у коді інструментарію, а сама АП лишається незмінною.

– Досягається можливість створення сумісних одна з одною агентних платформ – за рахунок використання однакових функцій роботи з мережею, XML, базами даних, мовами комунікації агентів і т.п.

Розроблений інструментарій надає можливості простого вибору необхідного рівня абстракції, полегшує процес розробки (маючи реалізовані деякі загально-поширені функції), забезпечує швидке тестування створених агентів, полегшує процес повторного використання вже створеного (reuse).

Інструментарій має перебирати на себе функції, які є загальними для будь-якої агентної платформи: взаємодія з операційною системою, робота з мережею і апаратним забезпеченням, забезпечення можливості використання базових сервісів.

Створюваний інструментарій оформлений як бібліотека класів. Всі класи інструментарію базуються на одному «батьківському» класі – aebtClass.

Агенті платформи та агентні середовища

Питання про те, яку комп’ютерну програму варто кваліфікувати як агента чи багатоагентну систему, постійно знаходиться в стадії інтенсивного обговорення. Зокрема, ми підтримуємо формулювання приведені [2]:

«Агент – це сутність, що знаходиться в деякому середовищі, від якого вона одержує дані котрі відображають події, що відбуваються в середовищі, яка інтерпретує їх і виконує команди, що впливають на середовище. Агент може містити програмні й апаратні компоненти... Відсутність чіткого визначення світу агентів і присутність великої кількості атрибутів, з ним зв’язаних, а також існування великої розмаїтості прикладів агентів говорить про те, що агенти – це досить *загальна технологія*, що акумулює в собі кілька різних областей».

Зазвичай наводять два основних визначення агентів – так звані «слабке визначення» та «сильне визначення» [3].

Так само, як і з визначенням агентів, класифікацій агентів можна навести безліч. Одна з найпоширеніших класифікацій приводиться в роботі [4].

Під агентною платформою (агентним середовищем, Agent Environment) зазвичай розуміють набір програмних інтерфейсів (API – Application Programming Interface),

який надає можливості створення, життєвого циклу, обміну повідомленнями, зв'язку та доступу до інформації та баз знань агентів. АП наділяють функціями базового життєвого середовища агентів. Основними функціями АП є: керування життєвим циклом агентів (народження, життя, смерть), обмін повідомленнями між агентами, підтримка мобільності агентів, забезпечення безпеки (для локальних систем та самих агентів), підтримка низькорівневої інфраструктури (використання мережних функцій операційної системи для забезпечення взаємодії з іншою АП)

Далі уточнимо два поняття – агентна платформа та агентне середовище (АС). АП – набір програмних інтерфейсів. АС – конкретна реалізація агентної системи у вигляді спеціалізованого середовища, яке функціонує і в якому містяться якісь знання та можуть народжуватися агенти.

Одну з перших класифікацій інструментарію створення агентних платформ було приведено у [5]. Основний зміст цієї класифікації відображає таблиця 1.

Таблиця 1

Класифікація агентних інструментаріїв

	Властивості	**Технології**	**Зразки**
Мобільні агентні інструментарії	мобільність, комунікативність	Java (77%) Python(7%) Tcl(7%) C/C++(6%)	Concordia Gossip FarGo Aglets
Мультиагентні інструментарії	агентна взаємодія, комунікативність, координація, вирішення конфліктних ситуацій	Java	MadKit Zeus JADE JATLite MAST
Інструментарії загального призначення	Немає спеціальних вимог та властивостей	Java(68%) Prolog(8%) C/C++(8%) Інші(16%)	FIPA-OS Ascape
Інструментарії для інтернет-агентів	Інтерактивні та персоналізовані	Java	Microsoft Agent Voyager NetStepper

Зазвичай архітектуру агентної системи можна умовно поділити на три рівні: операційної системи (мережні послуги, апаратні засоби, багатозадачність, багатопотоковість, інші), АП (життєвий цикл агентів, взаємодія, ін.), агентів. Наш інструментарій надає можливість поділу рівня АП на три додаткові рівні: «базовий»– бере на себе основні взаємодії з операційною системою та основні операції роботи з мережею, апаратним забезпеченням, ін.; «сервіси» – сервіси, побудовані на основі базового рівня; «застосувань» – сама АП.

Архітектуру нашого інструментарію відображає UML – діаграма зображена на рис.1.

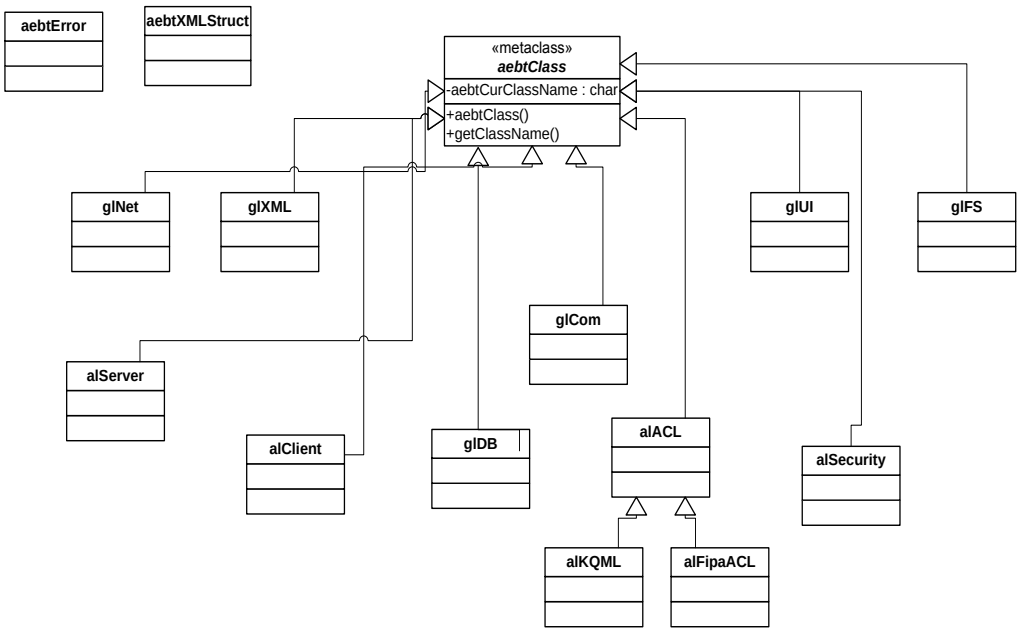

Рис. 1. UML-діаграма класів інструментарію

Поділ на рівні було проведено з точки зору використання функцій. До базового рівня були віднесені класи, що використовують лише функції операційної системи: glNet – автоматизує часто повторювані функції агентної платформи по роботі з мережею, glXML – робота з XML, glDB – робота з базами даних, glUI –

інтерфейс роботи з користувачем, glFS – робота з файловою системою.

До рівня сервісів належать класи: alServer – функції для організації сервера в клієнт-серверній взаємодії, alClient – функції для організації клієнта в клієнт-серверній взаємодії, alACL – функції для інтерпретації різних мов комунікації агентів, alSecurity – функції забезпечення безпечності роботи агентної платформи.

Враховуючи обрану мову реалізації, було вирішено реалізувати данні класи як інтерфейси Java 2, що забезпечить більшу гнучкість при створенні агентних платформ за рахунок можливості імплементації властивостей різних класів.

Принципи реалізації інструментарію

Перед початком реалізації інструментарію були сформовані вимоги до мови програмування. Вони базовані на вимогах до мов програмування агентів, описаних в [6]:

*1. Забезпечення переносності коду на різні платформи. Р*озроблений інструментарій призначений уніфікувати різні платформи, і, крім того, бути платформо-незалежним.

2. Підтримка мережної взаємодії. Ця вимога випливає з необхідності створення мобільних агентів. Мережна взаємодія забезпечить можливості легкої реалізації сервісів обміну повідомленнями та доступу то віддалених ресурсів.

3. Багатопотокова обробка. Надає можливість створення мультиагентів.

4. Має забезпечувати базові примітиви синхронізації потоків (семафори, монітори, критичні секції).

5. Об'єктна орієнтованість. Забезпечує більшу гнучкість системи та допоможе при повторному використанні.

Враховуючи ці вимоги, вибір був зроблений на користь мови Java2. Вона також добре підтримує і хорошу роботу з XML, файловою системою та користувацьким інтерфейсом.

Для забезпечення уніфікованості іменування класів та інтерфейсів інструментарію, ми прийняли такі правила іменування: базовий клас називається aebtClass; класи, що є допоміжними, та не входять до рівнів (базового та сервісів), мають префікс [aebt]; всі класи та інтерфейси базового рівня мають префікс [gl]; класи та інтерфейси рівня сервісів мають префікс [al]. Назва класу чи інтерфейсу починається з великої літери і іде зразу за префіксом.

Мережна взаємодія є однією з найважливіших функцій при розробці мобільної агентної системи. Модуль мережної взаємодії створений для уніфікації та зручного використання функцій мови Java по роботі з мережею. Цей клас надає своїм користувачам такі функції: передача рядків по мережі через сокети, приймання рядків через сокети, з'єднання з вказаним сокетом на вказаному комп'ютері, здійснює ДНС-обробку імені комп'ютера.

XML – сучасний Internet-стандарт, що слугує для забезпечення семантичної інтероперабельності у Web. Підтримка XML стала де-факто обов'язковою для будь-якої АП.

Для реалізації роботи з XML введений клас glXML. Цей клас працює так. Використовуючи функції Java, розпарюється XML-код, після чого заповнюється спеціальна структура типу aebtXMLStruct, яка складається з двох класів: aebtXMLStruct, aebtXMLTag. Передавання структури XML-файлу досягається за рахунок безмежного рівня вкладеності цих структур.

Робота з базами даних реалізована з використанням засобів JDBC мови Java. Основною задачею відповідного модуля інструментарію є уніфікація базових функцій: доступ до бази даних, виконання sql-запитів, блокування таблиці на читання/запис.

Користувацький інтерфейс реалізований з використанням бібліотеки Swing мови Java. На відміну від стандартних функцій, реалізована функціональність забезпечує більшу гнучкість та підтримує деякі додаткові засоби забезпечення безпеки та більшої пристосованості інтерфейсу саме до використання в агентних платформах. Функції модуля такі: створення вікон, діалогових вікон, вікон повідомлень; стандартні діалогові вікна та обробка реакції на поведінку, підтримка взаємодії з модулем безпеки та модулем файлової системи.

Для організації роботи з файловою системою введено клас glFS. Функції, які надає клас такі: створення, видалення, копіювання, перейменування файлів; створення, видалення, копіювання папок; читання/запис у файл; пошук файлів за маскою. Слід зазначити, що особливістю класу glFS є його тісна інтегрованість з класом alSecurity, що забезпечує можливості контролю над доступом до файлової системи користувача з боку агентів.

Мови комунікації агентів – це засіб отримання нових знань, вимог, коректування мети на основі нової інформації. Агенти створюють повідомлення визначеного формату. Ці повідомлення надсилаються визначеному адресату засобами агентного середовища. Останній приймає рішення, базуючись на інформації з цього повідомлення.

Кожна мова комунікації агентів має свою онтологію – формальний спільний ієрархічно структурований набір термінів для опису доменів, що можуть бути застосовані як основа для бази знань.

Компонентами онтології є концепція, предикат і дія [7]. Для створення онтологій надаються сервіси по створенню цих компонент та їх ієрархічне додавання до схеми (онтології).

В розроблений інструментарій закладено підтримку двох основних мов комунікації агентів – KQML та FIPA ACL. Клас alACL підтримує основні функції кодування та декодування повідомлень. Він є і базовим для двох інших класів – alKQML та alFipaACL.

Клас alACL є основним класом при обробці повідомлень. Він займається кодуванням/декодуванням повідомлення в формат, зручний при пересилці в мережі. Саме на цей клас покладений парсінг повідомлення в формат alACLMessage – тип, визначений спеціально для зручної роботи з розпарсеним повідомленням.

Класи alKQML та alFipaACL успадковуються від класу alACL та розширюють його додаючи до функцій парсінгу повідомлення можливості парсінга повідомлень відповідно мовою KQML та FIPA ACL.

При розробці агентної платформи необхідно врахувати декілька питань, без яких повноцінна робота платформи буде неможливою. Найважливішими з них є

розмежування і контроль доступу до файлової системи та за діями агентів. Для вирішення цих питань було введено клас alSecurity.

Перевірку можливостей цього інструментарію проведено при створенні АП, призначенням якої є забезпечення агентної взаємодії в системі дистанційного навчання. Використання інструментарію значно спростило процес проектування і скоротило час реалізації.

Висновки

Основним результатом проведеної роботи є розроблений інструментарій створення платформ агентних систем об´єктно-орієнтованого типу. Він представляє собою розвинену бібліотеку класів мови Java, зібраних в один „пакунок».

Розроблений інструментарій є актуальним для практичного розвитку агентних технологій. Він дозволить уніфікувати базову частину нових агентних платформ, спростить процес впровадження агентів у всі сфери інформаційних технологій. Інструментарій дозволить значно спростити та скоротити час розробки агентної платформи.

ЛІТЕРАТУРА

1. Анісімов А.В., Глибовець А.М., Жаб'юк В.Я. Основні архітектурні принципи побудови програмних систем реалізації мобільних агентів. // Вісник Київського національного університету імені Тараса Шевченка. —серія: фізико-математичні науки. – Випуск №3. – 2008. —С. 125-131
2. http://www.cselt.stet.it/fipa/fipa_rationale.htm
3. *Гороховський С. С.* Агентні технології: спроба критичного огляду // Наукові записки. Т. 18: Спеціальний випуск / Національний університет «Києво-Могилянська Академія». – К., 2000. – С. 391—395
4. http://w3.informatik.gu.se/~dixi/agent/class.htm
5. *Городецкий В. И., Грушинский М. С., Хабалов А. В.* Многоагентные системы // Новости искусственного интеллекта. – 1997. – № 1. – С. 15—30.
6. *Андон Ф.И., Коваль Г.И. и др*. Основы инженерии качества программных систем – К.: Академпериодика, 2002. – 504 с.
7. *Глибовец Н.Н., Глибовец А.Н., Шабинский А.С.* Применение онтологий и методов текстового анализа при создании интеллектуальных поисковых систем.// Проблемы управления и информатики №6 ноябрь-декабрь 2011. – с.95 -102.

Дата надходження статті до редколегії 17.04.2012 р.

ГЛИБОВЕЦЬ М. М. – д.ф.-м.н., професор, завідувач кафедри інформатики, декан факультету інформатики Національного університету «Києво-Могилянська академія».

Коло наукових інтересів: розподілені інтелектуальні системи, програмні системи підтримки електронного навчання.