УДК 681.3

Гломозда Д. К.

ВИКОРИСТАННЯ ПРОГРАМНИХ АГЕНТІВ ДЛЯ ОРГАНІЗАЦІЇ ЕФЕКТИВНОГО УПРАВЛІННЯ РОБОТОЮ ПРОГРАМНОЇ СИСТЕМИ ПІДТРИМКИ МЕРЕЖНОЇ СПІВПРАЦІ

Ключові слова: координація, мережі Петрі, агенти, JADE, рівневий контроль

У роботі розглядається побудова моделі системи програмної підтримки мережної співпраці та використання програмного агента для її ефективної реалізації. Доводиться, що запропонована модель координаційного механізму для часткових випадків таких систем зберігає свої властивості і залишається працездатною для довільної конфігурації системи. Пропонується схема багатоагентної системи, яка б моделювала роботу координаційного механізму.

Вступ

У статті продовжується розгляд задачі побудови високопродуктивної програмної системи підтримки мережної співпраці. У роботах [1; 2] була проаналізована модель часткового випадку такої системи. У цій статті доведемо, що така модель адекватно відображає роботу колаборативної системи довільної конфігурації.

Друга частина даної статті присвячена побудові агентної системи, яка б моделювала роботу нашої системи та служила основою програмної реалізації.

Моделювання роботи колаборативної системи довільної конфігурації

До складу колаборативної системи входять N користувачів, M сеансів, L ресурсів та координаційний механізм - сукупність позицій та переходів мережі Петрі, що зв'язує користувачів, сеанси та ресурси [2]. У загальному випадку мережа Петрі, що моделює роботу координаційного механізму нашої колаборативної системи, складатиметься з:

- 3 * N * M позицій та 4 * N * M переходів, що регламентують взаємовиключне створення сеансів;
- 9 * L позицій, що відповідають контролерам рівнів;
- 5 * N * L позицій та 10 * N * L переходів, що відповідають зв'язкам між контролерами рівнів та користувачами.

У праці [1] розглядалися та аналізувалися дві моделі колаборативної системи. Модель M1, що відповідала колаборативній системі з одним користувачем, одним сеансом, одним рівнем та одним ресурсом, та модель M2, в якій діяли два користувачі. Результати проведених досліджень засвідчили, що обидві мережні моделі обмежені, тобто кількість фішок мережі Петрі не перевищує певного максимального значення, та активні, бо немає безвихідних ситуацій. Постає питання, чи зможе модель, побудована за такими принципами, зберегти ці властивості, якщо в системі буде як завгодно багато користувачів, сеансів та/або ресурсів.

Доведемо, що це так. Спершу покажемо, що побудований згідно з нашою моделлю КМ здатен забезпечити взаємовиключний доступ до єдиного спільного ресурсу та взаємовиключний контроль єдиного спільного сеансу для довільної кількості користувачів.

Теорема 1. Координаційний механізм (далі КМ) вирішує задачу координації для колаборативної системи з довільною кількістю користувачів, одним сеансом та одним ресурсом, доступ до якого регламентується одним рівнем (далі ЗК1).

Доведення. Для доведення використаємо метод математичної індукції.

База індукції: КМ вирішує ЗК1 для моделі M1. Це справді так, бо відповідна мережа Петрі обмежена та активна, що було доведено у праці [1].

Крок індукції: Нехай Mn - модель, що містить n користувачів, один сеанс та один ресурс, доступ до якого регламентується одним рівнем. Нехай КМ вирішує ЗК1 для Mn. Тоді, за твердженням індукції, КМ має вирішувати ЗК1 для Mn + 1, іншими словами, додавання одного користувача не повинно призводити до недієздатності КМ. Так воно і є, як свідчить приклад моделі M2, що є розширенням моделі M1 з додаванням ще одного користувача і яка зберігає всі властивості моделі M1. Доведено.

Тепер розповсюдимо отриманий результат на колаборативну систему з довільною кількістю користувачів, сеансів, рівнів та ресурсів.

Теорема 2. КМ вирішує задачу координації для колаборативної системи з довільною кількістю користувачів, сеансів та ресурсів, доступ до яких регламентується своїм рівнем для кожного з них (далі ЗКn).

Доведення. Як зазначалося у праці [2], всі варіанти, коли кількість користувачів, сеансів та рівнів різна ($N = n, M = m, L = l; (n \neq m) \vee (l \neq n) \vee (m \neq l)$), можна окремо не розглядати, оскільки вони зводяться до випадку, коли N = M = L = min *(n, m, l) = x* (при $x \geq 2$).

Додавання додаткових сеансів не призведе до недієздатності КМ, оскільки, в найгіршому випадку, коли всі користувачі змагатимуться за один сеанс, КМ працює згідно з теоремою 1, а змагання окремих груп користувачів за різні сеанси зводиться до змагання меншої кількості користувачів за один сеанс, і в цьому разі КМ також працездатний за індукцією. Аналогічно виглядають справи зі змаганням за рівні. Доведено.

Як бачимо, наша модель здатна коректно відобразити процес роботи як завгодно складної та багатоелементної колаборативної системи. Керуючись цією інформацією, розглянемо процес побудови програмного агента, здатного моделювати роботу КМ нашої системи в як завгодно складному середовищі.

Схема агентної системи

При визначенні обчислювальної складності задачі верифікації КМ у роботі [2] було продемонстровано спорідненість цієї задачі із задачею верифікації агентів [3]. Зокрема, було показано, що КМ можна зіставити з агентом, що виконує задачу підтримки - утримує середовище в одному з допустимих станів [4, *344]*. Оскільки КМ у нашій моделі включає в себе механізми створення сеансів та контролери рівнів, що регламентують доступ до ресурсів, природно уявити його як агента, до якого надходять запити від користувачів (як людей, так і прикладних програм чи інших агентів) на використання потрібного ресурсу. Іншим варіантом реалізації може бути сукупність агентів, кожен із яких відповідає за свій сеанс або рівень. У деяких випадках доцільно використовувати проміжний варіант, коли одні агенти відповідають кожен за свій сеанс або ресурс, інші - за два, за три тощо. Наприклад, у тому випадку, коли ми хочемо унеможливити керівництво одночасно кількома сеансами чи одночасне використання якихось ресурсів одним і тим самим користувачем.

Таким чином, у загальному вигляді багатоагентна система, що моделює роботу КМ колаборативної системи, складатиметься з таких агентів:

1) User_Agent - агент-користувач, що використовує ресурси системи для вирішення своєї чи спільної з іншими агентами задачі;

2) Session_agent - агент, що відповідає за створення сеансу, доступ до нього та його закриття. Повинен слідкувати за тим, щоб у кожний момент часу у сеансу був один і тільки один керівник, тобто користувач, який має право його закрити. Зазвичай це той самий користувач, що створив сеанс, і його примусовий вихід із системи супроводжується закриттям сеансу. Але в разі дуже важливих сеансів доцільно зробити так, щоб цього не траплялося, і тоді контролер сеансу має взяти на себе керівництво сеансом доти, доки всі інші користувачі не від'єднаються від нього або доки попередній користувач не ввійшов у систему знову. Інший можливий варіант - обирання з-поміж користувачів сеансу нового керівника за одним з алгоритмів обрання лідера [5, *25-50]*.

3) Floor_agent - агент, що відповідає за надання взаємовиключного доступу до ресурсу. Також відповідає за коректне звільнення ресурсу в разі, якщо користувач, що зайняв його, аварійно від'єднується від системи. Може надавати можливість призупинення використання ресурсу без його звільнення, в цьому разі повинен слідкувати за тим, щоб пауза не затягувалася надовго, інакше примусово звільняє ресурс. Може підтримувати черги користувачів, тоді після звільнення ресурсу надає його тому, хто чекає найдовше. Реалізація рівневого контролю також може бути різною, більш детально її варіанти описано у праці [6, *2-3]*.

Структура взаємодії між об'єктами системи показана на рис. 1.

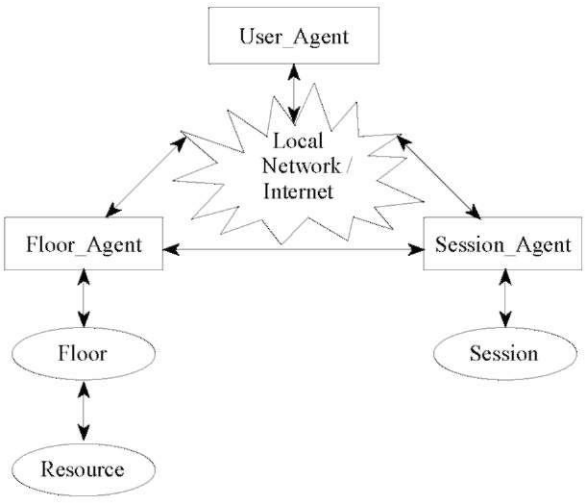

Рис. 1. Структура взаємодії об'єктів у системі

Для побудови системи було використано агентну архітектуру JADE [7; 8]. Структура системи виглядає так:

а) пакет Session - складається з агента Session agent та поведінки MaintainSessionBehaviour. Ця поведінка включає в себе всі операції агента із забезпечення коректного функціонування сеансу, зокрема коректне від'єднання користувачів та стеження за тим, щоб у кожний момент часу сеанс мав одного і тільки одного чітко визначеного керівника. Поведінка MaintainSessionBehaviour реалізована на базі класу CyclicBehaviour, тобто виконується нескінченно довго і зупиняється лише одночасно з агентом, що її запустив [9, *28]*.

б) пакет Floor - складається з агента Floor_agent та поведінки MaintainFloorBehaviour. Поведінка охоплює всі функції забезпечення взаємовиключного доступу до ресурсу та слідкує за тим, щоб жоден користувач не міг використовувати ресурс впродовж невизначено довгого часу. Якщо користувач призупиняє використання ресурсу, не звільнюючи рівень, і не поновлює роботу з ним упродовж певного, наперед визначеного проміжку часу, то у разі, якщо на доступ до цього ресурсу очікує інший користувач, контролер може примусово звільнити рівень для нього, а попереднього утримувача занести в чергу з пріоритетним правом на одержання ресурсу, коли від нього надійде команда «поновити роботу». Поведінка MaintainSessionBehaviour також реалізована на базі класу CyclicBehaviour.

в) пакет User - складається з агента User_Agent та його поведінок CreateJoinSessionBehaviour і UseFloorBehaviour. Перша поведінка відповідає за створення сеансу або приєднання до вже існуючого та, відповідно, від'єднання від нього та завершення, якщо більше ніхто цим сеансом не користується. Друга поведінка відповідає за доступ до ресурсу та його використання з можливістю тимчасового призупинення без звільнення рівня (щоб потім не довелося повторно на нього чекати). Обидві поведінки реалізовано на базі класу TickerBehaviour, тобто їх методи виконуються через рівні, наперед задані проміжки часу [9, *30]*. Таким чином моделюється певна послідовність дій користувачів, причому запуск кількох агентів з різними періодами очікування дає змогу відстежувати взаємодію кількох користувачів та роботу координаційного механізму.

Діаграму класів наведено на рис. 2.

Робота системи відбувається таким чином:

1) Стартують агенти-контролери сеансів та рівнів, реєструються в директорії DF (Directory Facilitator - агент, що надає іншим учасникам спільної роботи сервіс «жовтих сторінок», завдяки якому вони можуть знаходити та користуватися службами один одного [9, *9—10*]), після чого додають до себе поведінки MaintainSessionBehaviour та MaintainFloorBehaviour відповідно. При цьому сеанси вважаються нествореними, а рівні - вільними.

2) Стартують агенти-користувачі, реєструються в директорії DF, після чого додають до себе

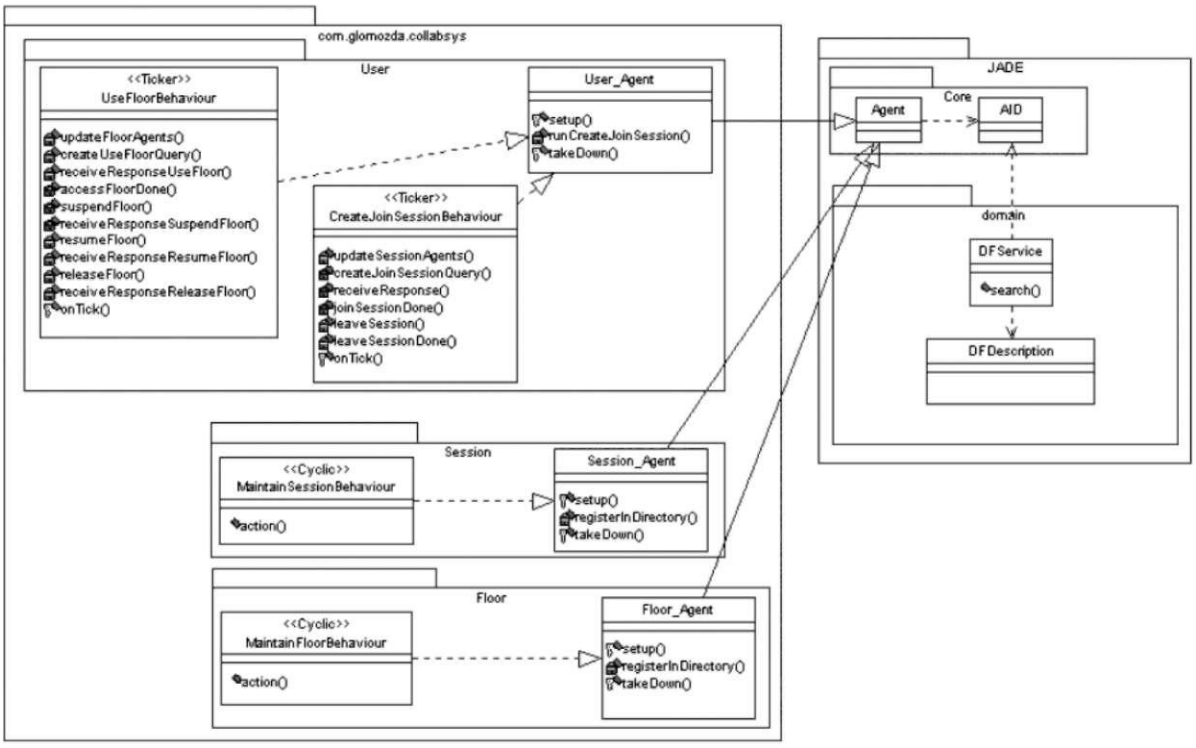

Рис. 2. Діаграма класів

поведінку CreateJoinSession. Отримавши від DF-агента список наявних сеансів, вони під'єднуються до того, який їм потрібен. Якщо сеанс ще не створений, користувач його створює і стає його керівником. Якщо створений - приєднується до нього як простий учасник.

3) Приєднавшись до сеансу, користувач додає поведінку UseFloorBehaviour. Якщо потрібний йому рівень із наданого DF-агентом переліку зайнятий, він заноситься в чергу з пріоритетним правом на доступ до ресурсу після його звільнення. Якщо рівень вільний, він його займає і працює доти, доки не звільнить сам або не буде від'єднаний системою.

4) Якщо користувач не є керівником сеансу та не займає рівень, він може без проблем вийти із системи. Якщо він є керівником сеансу, то може покинути систему лише тоді, коли ніхто інший цим сеансом не користується. В іншому разі користувачу доведеться зачекати, доки всі інші учасники залишать сеанс.

5) Після того як користувач використав і звільнив ресурс та від'єднався від сеансу, його поведінки UseFloorBehaviour та CreateJoinSessionBehaviour знищуються і агент припиняє роботу.

Запускаючи довільну кількість агентів-контролерів сеансів і рівнів та агентів-користувачів, можна моделювати поведінку як завгодно великої системи.

Висновки

Отримані результати дозволили нам побудувати зручний засіб для дослідження роботи протоколу рівневого контролю.

Надалі передбачається реалізація кількох підходів до керування сеансом, зокрема процедура обрання лідера з-поміж інших учасників у разі виходу з системи керівника сеансу, та процедур, що моделюватимуть дії системи у разі аварійного припинення сеансу чи проблем з ресурсом.

1. *Гломозда Д. К., Глибовець М. М.* Формальна модель функціонування колаборативного середовища // Тези доповідей Третьої Міжнародної конференції «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD'2006). - Київ (Україна), 2006. - С. 225-230.
2. *Глибовець М. М., Гломозда Д. К.* Складність задачі верифікації координаційного механізму системи програмної підтримки мережної співпраці // Тези доповідей Четвертої Міжнародної конференції «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD'2007). - Бердянськ (Україна), 2007. - С. 58-62.
3. *Wooldridge M., Dunne P. E.* The Computational Complexity of Agent Verification // Intelligent Agents VIII: Proceedings of the Eigth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001). - Berlin: Springer-Verlag, 2002. - P. 115-127.
4. *Wooldridge M.* The Computational Complexity of Agent Design Problems // Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000). - Boston, MA, 2000. - P. 341-348.
5. *Lynch N.* Distributed Algorithms. - San Mateo, CA: Morgan Kaufmann Publishers, 1996. - 827 p.
6. *Myers B. A., Chuang Yu Shan A., Tjandra M. et al.* Floor Control in a Highly Collaborative Co-Located Task. - [Submitted for publication]. - May 22, 2006. - <www.cs.cmu.edu/~pebbles/papers/pebblesfloorcontrol.pdf>.
7. *Глибовец Н. Н.* Использование JADE (JavaAgent Development Environment) для разработки компьютерных систем поддержки дистанционного обучения агентного типа // Образовательные технологии и общество. - 2005. - Т. 8, № 3. - С. 325-345.
8. *Bellifemine F., Caire G., Poggi A., Rimassa G.* JADE: A White Paper // EXP in search of innovation. - 2003. - Vol. 3, n. 3. - P. 6-19.
9. *Bellifemine F., Caire G., Trucco T., Rimassa G.* JADE Programmer's Guide. - [Cited. 2005, June 18] - Available from <http://jade.tilab.com/doc/programmersguide.pdf>.

D. Glomozda

USING SOFTWARE AGENTS TO ORGANIZE EFFECTIVE CONTROL OVER FUNCTIONING OF A NETWORK COLLABORATION SUPPORT SOFTWARE SYSTEM

A model of a network cooperation support software system is built and the usage of software agent to implement it effectively is considered in the article. It is proved that the proposed model of coordination mechanism for the special cases of such systems preserves its properties and stays correct in the case of an arbitrary system configuration. A scheme of multi-agent system which models the functioning of the coordination mechanism is proposed.