УДК 004.031.2

ЄФРЕМОВ К.В.,
МАЗУР Р.Ф.

СИСТЕМНИЙ ТА СТРУКТУРНИЙ РІВНІ ОРГАНІЗАЦІЇ МУЛЬТИАГЕНТНИХ СИСТЕМ З МОЖЛИВІСТЮ АДАПТАЦІЇ ПРОЦЕСУ ОБРОБКИ ДАНИХ

У статті розглядаються системний рівень подання мультиагентної системи та структурна організація її агентів. Агент представляється у вигляді набору елементів, що будують граф із можливими шляхами обробки вхідних заявок. Запропоновано механізм обробки заявок агентом, який надає можливість адаптації системи до класу задач, що розв'язуються, та динамічної зміни цілей агентів.

The paper deals with system level representation of multi-agent systems and structural organization of its agents. The agent is presented as a set of elements that construct a graph of possible routes of processing incoming requests. What is offered is the way an agent handles requests that allows the system to adapt to the classes of tasks solved and dynamic changes of agent's goals.

Вступ

Прогрес у розвитку засобів обчислювальної техніки забезпечує можливість розв'язувати все більш широкий клас задач за допомогою програмних систем. Це обумовлює необхідність застосування нових підходів до їх проектування та розробки. Один із таких підходів оснований на застосуванні ідеї колективної діяльності сукупності агентів – мультиагентних систем (МАС).

Одним із успішних застосувань МАС стало агентно-орієнтоване моделювання, яке надало можливості створення адекватних моделей у таких сферах, як: біологія, соціологія, машинобудування та ін. Зокрема, представивши у вигляді агентів вірус імунодефіциту людини та лімфоцити, за допомогою МАС у роботі [7] моделювався перебіг захворювання ВІЛ-інфікованих з метою визначення причин значної диференціації проміжків часу між зараженням та проявом СНІД у різних хворих. У роботі [10] за допомогою МАС була змодельована система контролю за рухом повітряного транспорту. Варто відзначити, що в подібних системах поведінка кожного з агентів точно визначається відповідно до об'єктів, які за допомогою них моделюються.

Іншим напрямком розвитку МАС є вироблення концепцій агентно-орієнтованого проектування (АОП) програмних систем може використовуватися у сферах електронної комерції, управління та моніторингу телекомунікаційних систем, оптимізації транспортних систем чи промислового виробництва, аналізу інформації та інформаційних середовищ (наприклад, Інтернет), автоматичного планування зустрічей, електронних розваг чи інтерактивних ігор [9].

За останні роки було запропоновано ряд методологій для розробки мультиагентних систем, таких як TROPOS [1], Prometheus [6], ADELFE [3], Gaia [11], у яких передбачається розгляд нотацій агентів, починаючи з ранніх стадій проектування. Для реалізації подібних проектних рішень можуть використовуватися програмні каркаси, такі як JADE [2]. Варто відзначити, що втілення даних підходів має досить багато спільного із сервісно-орієнтованою архітектурою [5] та багатошаровим програмним забезпеченням.

Постановка задачі

Не зважаючи на прогрес в області розробки методологій АОП, на сьогоднішній день, на жаль, ще не вироблено достатньо підходів для реалізації таких властивостей агентів, які забезпечують адаптивність системи. Тому метою даної роботи стала розробка моделі організації МАС з адаптивними агентами, використання якої дає можливість оптимізувати процес обробки даних в системі при динамічній зміні класу задач, що розв'язуються, та цілей агента.

Визначення мультиагентної системи

Наразі дослідники не дійшли згоди стосовно однозначних визначень агента та мультиагентної системи, однак більшість із них погоджуються зі ствердженнями, наведеними в [12]. Зокрема, агентом є комп'ютерна система, яка розміщена в певному середовищі та здатна виконувати автономні дії з метою досягнення поставлених під час проектування цілей.

Система називається мультиагентною, якщо вона складається з набору автономних агентів, що можуть взаємодіяти між собою [4]. Такі системи використовуються для розв'язання задач, рішення яких є неможливим або занадто складним при використанні одного агента чи монолітної системи. Важливою особливістю МАС є те, що її глобальна функція не задається чи проектується, вона виникає на основі взаємодії агентів між собою в наслідок синергетичного ефекту.

У роботі [12] було виділено набір таких властивостей: автономність, соціальність, реактивність та проактивність, якими в тій чи іншій мірі повинен володіти агент. Останні дві властивості тісно пов'язані з адаптивністю, тобто такою властивістю системи, яка полягає в тому, що вона може виконувати поставлені задачі у кілька способів та має достатні знання про свою будову для здійснення змін у процесі роботи.

З метою оптимізації витрат на досягнення цілей агенти мають включати в себе засоби для оцінки власної поведінки та продуктивності разом із здатністю до перепланування власних дій з метою підвищення ефективності своїх операцій [8].

Системний рівень організації МАС

Як уже було сказано, складовими частинами мультиагентної системи є множина агентів та їхнє середовище. Оскільки така система не має інших складових, необхідний зв'язок з іншими системами здійснюється через агентів. Кожен із агентів здатен аналізувати стан середовища, у якому знаходиться, та генерувати певний вплив на це середовище. Ці вхідні та вихідні дані агента *A* мають свої алфавіти (*X* та *Y* відповідно), а сам агент може розглядатися як перетворення над даними в цих алфавітах $A: X \to Y$.

Дані на вході агента можуть містити інформацію про вхідні аргументи його перетворення з алфавітом X_D та про директиви контролю цього перетворення з боку середовища з алфавітом X_C. Тобто, $X = X_D \times X_C$.

Насправді, вхідний алфавіт агента характеризує дані, у яких він може бути зацікавлений. Вихідний – дані, які він може надати системі. Це означає, що факт появи даних на вході агента ніяким чином не гарантує появу даних на його виході. Однак, якщо агент має вихідний алфавіт, то вважається що в процесі роботи системи він обов'язково спродукує дані (варто відзначити, що тривалість роботи системи ніяк не обмежується), і в контексті всієї тривалості роботи системи агент здійснюватиме перетворення. Справедливе і протилежне твердження: якщо агент не надає відомостей про свій вихідний алфавіт, то він здатен лише отримувати вхідні дані. Для таких агентів вихідний алфавіт рівний $O = \{\theta\}$, тобто пустим даним. Таким чином, можливі три типи агентів:

1. агенти-перетворення (з повними вхідним та вихідним алфавітом), $A: X \to Y$;
2. агенти-споживачі (з пустим алфавітом вихідних даних) , $A: X \to O$;
3. агенти-генератори (з пустим алфавітом вхідних аргументів), $A: O \times X_C \to Y$.

Існування агентів останніх двох типів обґрунтовується необхідністю зв'язку з іншими зовнішніми системами. Далі розглядатимуться агенти першого типу.

Поставимо умову, що сенсори агентів отримують від середовища дані, представлені кортежем $R = \langle X', Y', x \rangle$, який несе інформацію про заявку на виконання перетворення $X' \to Y'$ над аргументами *x*. При цьому, оскільки агент є автономним, він здатен самостійно зробити запит типу *R* до середовища.

Щоб називатися адаптивною та автономною, система повинна володіти механізмом самостійної оцінки власного стану та винесення рішення про необхідність змін у роботі. Для здійснення такої оцінки у системі існує *p* параметрів роботи агента. У такому випадку стан агента може описуватися вектором $\bar{s} \in \mathrm{R}^p$, який містить значення кожного з параметрів.

Стуктурний рівень організації агента

Нехай агент *A* визначається набором елементів $a_i, i = \overline{1,n}$, кожен з яких може виконувати перетворення f_{a_i} над підміножиною вхідного алфавіту агента X^i. При цьому результатом такого перетворення є підмножина вихідного алфавіту агента Y^i. Тобто,

$$A = \left\{ a_i \middle| f_{a_i} : X^i \to Y^i \right\}, i = \overline{1,n},$$
$$X = \bigcup_i X^i, Y = \bigcup_i Y^i.$$

Семантика кожного такого перетворення визначається парою вхідного та вихідного алфавітів та символом перетворення $\left(X^i, Y^i, f_{a_i}\right)$. Дані

про елементи, а саме про їхній вхідний та вихідний алфавіт, заносяться до внутрішнього реєстру агента.

Вважатимемо, що набір елементів в агенті може змінюватися завдяки реконфігурації. При цьому поява нового елемента може означати для агента або розширення його функціональності, або появу альтернативних засобів обробки даних.

Знання про внутрішню структуру агента можуть бути представлені у вигляді орієнтованого графа G з множиною вершин V та множиною дуг E. Множина перетворень, які здійснюють елементи агента, є підмножиною прямого добутку його вхідного та вихідного алфавітів:

$$A \subseteq X \times Y = \bigcup_i X^i \times \bigcup_i Y^i .$$

Дана множина елементів агента складає множину вершин графа G, тобто $A=V$. Множина дуг є підмножиною усіх відношень на множині A та визначається як

$$E = \left\{(j,k) : Y^j \subseteq X^k\right\};$$
$$j = \overline{1,n};\ k = \overline{1,n}.$$

При появі на вході агента заявки $R = \langle X', Y', x \rangle$, він може відреагувати на неї, виконавши відповідне перетворення. При цьому необхідною умовою обробки є

$$\begin{cases} X' \subseteq X, \\ Y' \subseteq Y; \end{cases} \Rightarrow \exists \left\{j \middle| j \in [1,n]\right\}, \left\{k \middle| k \in [1,n]\right\}:$$
$$X' = \bigcup_{\{j\}} X^j \cap Y' = \bigcup_{\{k\}} Y^k.$$

Виконання даної умови означає, що існують такі елементи агента, вхідний та вихідний алфавіти яких відповідають заявці, що надійшла.

Достатньою умовою виконання заявки агентом є існування непустої множини шляхів $\{P_{j,k}\}$ між відповідними вершинами j та k:

$$P_{j,k} = \{p_{j.k}\} = \left\{\langle a_j, ..., a_k \rangle\right\}.$$

При цьому вважатимемо, що передача даних по одній із дуг графа G пов'язана зі зміною стану агента ($\Delta\bar{s}$). Тобто існує множина функцій $\{\varphi : E \to \mathrm{R}^p\}$, які обчислюють вартість кожного з переходів у графі для агента.

Таким чином, структуру S агента можна описати через множину його елементів-перетворень A, вектор стану $\bar{s}$ та множину функцій, які визначають вартість кожного з переходів по графу G:

$$S = \left\langle A, \bar{s}, \{\varphi\} \right\rangle.$$

Граф G може мати декілька маршрутів між будь-якими парами вершин j та k, що дає можливість зміни шляху обробки заявки в залежності від вхідних даних x, поточного набору елементів $A = \{a_i\}$, та значень параметрів стану агента $\bar{s}$. При цьому потрібно розв'язувати задачу оптимізації.

Процес обробки заявки в агенті

Із появою заявки R на вході агента у випадку, коли вона може бути оброблена, в агенті починається просування пакетів з даними. Даний пакет C у момент часу τ є четвіркою, яка складається з унікального ідентифікатору поточного запиту $u \in \mathrm{N}$, алфавіту даних, які обробляються M, поточних даних d для обробки елементом агента та шляху P, який даний пакет пройшов в агенті (упорядкована підмножина елементів останнього):

$$C^\tau = \left\langle u, M^\tau, d^\tau, P^\tau \right\rangle,\ P^\tau \subseteq A.$$

При $\tau = 0$ маємо $M^\tau = X', d^\tau = x, P^\tau = \{\ \}$.

Після обробки пакета елементом a_t $(t \in [1,n])$ агента, частини M та d змінюються відповідно до вихідного алфавіту поточного елемента та результатів здійсненого перетворення:

$$\begin{cases} M^{\tau+1} = Y^t, \\ d^{\tau+1} = f_{a_t}(d^\tau), \\ P^{\tau+1} = P^\tau \cup a_t. \end{cases}$$

Далі відбувається передача пакета $C^{\tau+1}$ наступним елементам. На агента покладається задача визначення, кому з елементів необхідно передати заданий пакет. Останній може бути доставлений лише тому елементу, який має відповідний вхідний алфавіт. Отже, при прийнятті рішення про пункт призначення пакета агент формує вибірку елементів із множини доступних (тих, які можуть обробити поточні дані), керуючись результатами аналізу поточного стану системи. Така вибірка розміром з m елементів може бути описана наступним чином:

$$\left\{a_i \middle| M^\tau \subseteq X^i\right\} \subseteq A, i = \overline{1,m}.$$

Слід вказати, що ситуація з декількома маршрутами можлива, коли агент має в своєму складі елементи, чиї вхідні алфавіти перетинаються:

$$m>1 \Rightarrow \exists\, j,k \in [1,n]: X^j \cap X^k \neq \{\ \}.$$

Зміни у роботі системи можуть бути здійснені шляхом модифікації рішень стосовно доставки пакета через граф *G* агента.

Агент повинен вибрати маршрут з найменшою вартістю для просування пакету. Тоді, коли такий маршрут невідомий, виконується широкомовна передача пакета елементам, які можуть обробити дані, що передаються всередині. Із усіх можливих маршрутів, утворених таким чином, вибирають ті, які дають мінімальний час проходження пакета. Такі маршрути запам'ятовуються та використовуються для подальших обробок відповідних заявок.

Виходячи зі свого поточного стану та набору функцій φ, агент може прийняти рішення про «небажаність» проходження пакету по певній дузі графа *G* або маршруту. У такому випадку агент може уповільнювати проходження даних по вибраній дузі і корегувати таким чином вибір маршруту.

Отже, агент отримує можливість зміни способів обробки вхідних заявок у залежності від власних цілей, які визначаються функцією вартості φ.

Така реконфігурація маршруту здійнюється у випадку появи в структурі агента елемента, який може входити до альтернативного маршруту, або зміни вартості маршрутів обробки заявки.

Висновки

У роботі на системному рівні подання МАС виділені три типи агентів, запропоновані їхня структурна організація та процес обробки заявок в них, які створюють можливість динамічної структурної реорганізації агентів з метою оптимізації процесу роботи МАС.

Перелік посилань

1. Bresciani P. Tropos: An Agent-Oriented Software Develpment Methodology / Bresciani P., Giorgini P., Giunchiglia F. – Trento : Kluwer Academic Publishers, 2003.
2. Caire F. JADE: A software framework for developing multi-agent applications. Lessons learned / Caire F., Poggi A., Rimassa G. // Information and Software Technology. – 2008. – 50. – pp. 10-21.
3. Carole B. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering : ESAW'02 Proceedings of the 3rd international conference on Engineering societies in the agents world III / Carole B., Sylvain P., Gauthier P. - Berlin : Springer-Verlag, 2003.
4. D'Inverno M. Understanding agent systems / D'Inverno M., Luck M., Luck M. M. – Springer, 2004. – 2 : p. 240.
5. Erl T. Service-oriented architecture: concepts, technology, and design. – Prentice Hall Professional Technical Reference, 2005.
6. Padgham L. The Prometheus Methodology / Padgham L., Winikoff M. // Engineering Applications of Artificial Intelligence. –2004. – 18(2).
7. Perrin D. An agent-based approach to immune modelling / Perrin D., Ruskin H. J., Burns J., Crane M. // Lecture Notes in Computer Science. – Springer, 2006. – 3980. – pp. 612-621.
8. Robertson R. Self-adaptive software: applications: Second International Workshop, IWSAS 2001 (Balatonfűred, Hungary, May 17-19, 2001) / Robertson R., Shrobe H. – Springer, 2003. – Vol. 2.
9. Russell S. J. Artificial Intelligence: A Modern Approach / Russell S. J., Norvig P. – Prentice Hall, 2009. – 3.
10. Wolfe S. Comparing Route Selection Strategies in Collaborative Traffic Flow Management: IEEE/WIC/ACM International Conference on IAT / Wolfe S., Jarvis P., Enomoto F., Sierhuis M. – 2007.
11. Wooldbridge M. The Gaia Methodology for Agent-Oriented Analysis and Design / Wooldbridge M., Jennings N. R., Kinny D. // Autonomous Agents and Multi-Agent Systems. – Kluwer Academic Publishers, 2000. – 3. – pp. 285-312.
12. Wooldridge M. Intelligent agents: theory and practice / Wooldridge M., Jennings N. R. // The Knowledge Engineering Review. – 1995. – 10(2). – pp. 115-152.