ДВОВИМІРНА КЛАСИФІКАЦІЯ АРХІТЕКТУР ПРОГРАМНИХ СИСТЕМ НА ОСНОВІ LLM-АГЕНТІВ

Киселевич В.В.
асистент кафедри комп'ютерних наук та інформаційних технологій
Житомирський державний університет імені Івана Франка, Україна

Постановка проблеми. Архітектури програмних систем на основі великих мовних моделей найчастіше подають як лінійний перелік, у якому послідовно згадують одноагентну схему, схему з викликом інструментів, мультиагентну схему та схему з окремим кроком перевірки результату. Оглядові праці фіксують складнішу картину: порівняння понад двадцяти репрезентативних систем агентних робочих потоків показує співіснування кількох груп патернів [1]. Емпіричні дані про відмови теж не підтверджують лінійності: на корпусі з понад 1600 розмічених траєкторій виведено таксономію з 14 режимів відмов, згрупованих у три категорії, а саме проблеми проєктування системи, неузгодженість між агентами та недостатню верифікацію завдання [2]. Потрібен спосіб упорядкування архітектурних рішень, який не змішує різні підстави поділу.

Аналіз останніх досліджень. Фундаментальні таксономії описують агента з погляду внутрішньої будови: виокремлюють модульну пам'ять, структурований простір дій і узагальнений процес ухвалення рішень [3]. Окрема лінія стосується власне програмної архітектури: у SALLMA перелічено обмеження одноагентних рішень і запропоновано поділ на операційний рівень та рівень знань [4]. Аналіз простору проєктування показує, що формулювання запитів і топологія взаємодії є окремими вимірами оптимізації [5], а оснащення агента описано в огляді засобів навчання роботи з інструментами [6]. Патерн з провідним агентом, який планує та перепланує роботу після помилок, реалізовано в Magentic-One [7], ієрархічне управління зі структурованим протоколом обміну реалізовано в TalkHier [8]. Вплив середовища виконання показано окремо: агентно-комп'ютерний інтерфейс дає pass@1 12,5 відсотка на SWE-bench без заміни моделі [9], а виконуваний код замість викликів у форматі JSON підвищує успішність на величину до 20 відсотків [10; 11]. Числові оцінки з праць [3; 5; 7; 8; 10; 11] походять із препринтів. Спільним

є те, що організаційну схему та засоби контролю дій описують в одному ряду, без розмежування підстав поділу.

Мета. Мета роботи полягає у побудові класифікації архітектур агентних систем, що задовольняє вимогу єдності підстави поділу, та в перевірці твердження про щонайменше двовимірність простору архітектурних рішень.

Виклад основного матеріалу. Перелік «одноагентний, з інструментами, мультиагентний, з верифікацією» ставить в один ряд дві різні ознаки. Перші дві позиції відповідають на питання про кількість обчислювальних одиниць та спосіб їхньої взаємодії, тобто описують організацію; дві інші відповідають на питання про засоби, якими розширюють і перевіряють дії, тобто описують оснащення. Правило поділу понять вимагає, щоб члени поділу виокремлювалися за однією підставою і взаємно виключали один одного. Тут ця вимога порушена: одноагентна система з викликом функцій належить одночасно до першого і до другого члена переліку [6; 10], а мультиагентна система з окремим кроком перевірки належить до третього і четвертого [2]. Перетин членів свідчить, що перед нами дві накладені одна на одну класифікації.

Виокремлено дві осі з власною єдиною підставою кожна. Вісь організації агентів має три значення: одноагентна конфігурація, мультиагентна кооперативна (рівноправні агенти без керівного вузла) та оркестрована (виділений агент планує та розподіляє підзадачі) [7; 8]. Вісь шару надійності має чотири значення за обсягом зовнішнього контролю над діями: базовий шар, шар з інструментами [6; 10], шар з верифікацією [2; 4] і шар з harness, тобто кероване середовище виконання, яке визначає перелік доступних дій та межі кожного кроку [9; 11]. Підставою першої осі є топологія взаємодії, підставою другої є склад засобів контролю, що узгоджується з висновком про незалежність топології від решти проєктних рішень [5].

Сформульовано умову незалежності осей: дві осі є незалежними, якщо для кожного значення першої множина допустимих значень другої залишається тією самою. Умова справджується в обох напрямах. Мультиагентність не диктує верифікації, оскільки саме брак перевірки завдання утворює окрему категорію відмов таких систем [2]. Верифікація не диктує мультиагентності, оскільки самоперевірку описано як етап роботи одного агента з інструментами [6]. Оснащення й топологію оптимізують окремими процедурами [5].

Побудовано матрицю «організація агентів × шар надійності» розміром три на чотири (табл. 1). Клітинки заповнено конфігураціями, що мають описані в літературі відповідники: одноагентна з інструментами відповідає асистентові розробника з викликом функцій [6; 10], одноагентна з harness відповідає роботі в керованому середовищі [9; 11], оркестрована з інструментами відповідає провідному агентові зі спеціалізованими виконавцями [7; 8], а оркестрована з верифікацією передбачає окремого агента перевірки [4].

Дві клітинки з дванадцяти залишаються порожніми, і обидві відповідають поєднанню шару з harness із розподіленою організацією. Це не випадковість добору джерел. Harness передбачає єдину точку контролю середовища, через яку проходить кожна дія і в якій обчислюються межі допустимого наступного кроку [9; 11]. Мультиагентна кооперативна організація такої точки не має за визначенням, бо рівноправні агенти діють паралельно й кожен володіє власним каналом дій. Оркестрована організація має внутрішню точку керування, проте вона належить самій системі, тоді як harness має бути зовнішнім щодо агентів, інакше межі кроку встановлює той, кого обмежують. Звідси випливає напрям подальших досліджень, а саме побудова зовнішнього контуру контролю для групи агентів без центрального вузла [2].

Таблиця 1

Матриця «організація агентів × шар надійності»

Організація	**Базовий**	**З інструментами**	**З верифікацією**	**З harness**
Одноагентна	Запит без інструментів	Виклик функцій	Самокритика або оцінювач	Кероване середовище
Мультиагентна кооперативна	Обмін репліками	Рівноправні агенти з інструментами	Виконавець і критик	Немає відомих реалізацій
Оркестрована	Розподіл підзадач	Оркестратор і виконавці	Окремий агент перевірки	Немає відомих реалізацій

Джерело: розробка автора

Практичний наслідок матриці полягає в тому, що два проєктні рішення ухвалюються окремо: питання про кількість агентів і спосіб їхнього узгодження не дає відповіді на питання про засоби контролю дій.

Тому мінімальний опис конфігурації складається з пари координат з обґрунтуванням вибору за кожною віссю.

Обмеження. Робота не містить власного експерименту, тому не доводить переваги жодної клітинки матриці за якістю чи витратами. Числові оцінки запозичені з чужих вимірювань на різних бенчмарках і базових моделях, тому не зіставні між собою; частина походить із препринтів [3; 5; 7; 8; 10; 11]. Твердження про порожні клітинки стосується опрацьованої вибірки з одинадцяти джерел і означає відсутність описаної реалізації, а не доведену неможливість такої конфігурації.

Висновки. Обґрунтовано, що лінійний перелік архітектур агентних систем є логічно некоректним, бо змішує ознаку організації з ознакою оснащення й дає перетин членів поділу. Виокремлено дві осі з єдиною підставою кожна та сформульовано умову їхньої незалежності. Побудовано матрицю з дванадцяти клітинок, у якій десять мають описані відповідники, а дві залишаються порожніми, причому обидві відповідають поєднанню зовнішнього контролю середовища з розподіленою організацією. Ця прогалина визначає напрям подальшої роботи.

Список використаних джерел:

1. Yu C., Cheng Z., Cui H. [та ін.] A Survey on Agent Workflow – Status and Future. 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD). 2025. P. 770–781. DOI: 10.1109/icaibd64986.2025.11082076.
2. Cemri M., Pan M. Z., Yang S. [та ін.] Why Do Multi-Agent LLM Systems Fail? Advances in Neural Information Processing Systems 38. 2025. P. 135676–135729. DOI: 10.52202/085713-4082.
3. Sumers T. R., Yao S., Narasimhan K. [та ін.] Cognitive Architectures for Language Agents : препринт. arXiv:2309.02427. 2023. URL: https://arxiv.org/abs/2309.02427.
4. Becattini M., Verdecchia R., Vicario E. SALLMA: A Software Architecture for LLM-Based Multi-Agent Systems. 2025 IEEE/ACM International Workshop New Trends in Software Architecture (SATrends). 2025. P. 5–8. DOI: 10.1109/satrends66715.2025.00006.
5. Zhou H., Wan X., Sun R. [та ін.] Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies : препринт. arXiv:2502.02533. 2025. URL: https://arxiv.org/abs/2502.02533.
6. Xu W., Huang C., Gao S. [та ін.] LLM-Based Agents for Tool Learning: A Survey. Data Science and Engineering. Vol. 10, No. 4. 2025. P. 533–563. DOI: 10.1007/s41019-025-00296-9.
7. Fourney A., Bansal G., Mozannar H. [та ін.] Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks : препринт. arXiv:2411.04468. 2024. URL: https://arxiv.org/abs/2411.04468.
8. Wang Z., Moriyama S., Wang W. [та ін.] Talk Structurally, Act Hierarchically: A Collaborative Framework for LLM Multi-Agent Systems : препринт. arXiv:2502.11098. 2025. URL: https://arxiv.org/abs/2502.11098.

9. Yang J., Jimenez C. E., Wettig A. [та ін.] SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. Advances in Neural Information Processing Systems 37. 2024. P. 50528–50652. DOI: 10.52202/079017-1601.
10. Wang X., Chen Y., Yuan L. [та ін.] Executable Code Actions Elicit Better LLM Agents : препринт. arXiv:2402.01030. 2024. URL: https://arxiv.org/abs/2402.01030.
11. Wang X., Li B., Song Y. [та ін.] OpenHands: An Open Platform for AI Software Developers as Generalist Agents : препринт. arXiv:2407.16741. 2024. URL: https://arxiv.org/abs/2407.16741.