DOI 10.36074/logos-14.08.2026.009

МУЛЬТИАГЕНТНІ АРХІТЕКТУРИ НА ОСНОВІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ В ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ВАРТІСТЬ ДЕКОМПОЗИЦІЇ

Киселевич Володимир Володимирович[1]

1. асистент кафедри комп'ютерних наук та інформаційних технологій
Житомирський державний університет імені Івана Франка, УКРАЇНА

Розподіл ролей між кількома спеціалізованими агентами, побудованими на великій мовній моделі (LLM), став самостійним архітектурним патерном розробки програмного забезпечення: систему подають як команду, де кожен агент відповідає за окрему стадію життєвого циклу. Такий підхід реалізовано в системах ChatDev [1] і MetaGPT [2], співпрацю агентів у визначених ролях під час генерації коду досліджено в праці [3], а систематичний огляд напряму подано в праці [4].

Поширене припущення полягає в тому, що декомпозиція задачі між ролями поліпшує результат сама по собі. Проте емпіричні дані свідчать, що виграш не є безумовним: у праці [5] зазначено, що приріст показників мультиагентних систем на популярних тестових наборах часто виявляється незначним, а в праці [6] показано, що один агент у багатоетапному діалозі досягає рівня однорідних мультиагентних процесів на семи наборах задач за меншої обчислювальної вартості. Отже, питання полягає не в тому, чи розширює декомпозиція коло розв'язуваних задач, а в тому, якою ціною вона це робить.

Мета дослідження полягає в тому, щоб зіставити виграш від декомпозиції з супутніми втратами та сформулювати умову, за якої перехід до мультиагентної архітектури є обґрунтованим.

Виграш від декомпозиції має дві складові. Першою є розширення кола задач: складну задачу поділяють на підзадачі, кожна з яких лежить у межах спроможності окремого агента. Другою є спеціалізація: рольовий запит звужує простір відповідей агента, що підвищує узгодженість результату з очікуваним форматом [3]. Контрольоване зіставлення архітектур [7] підтверджує наявність

виграшу й окреслює його межу: мультиагентні схеми поліпшують узагальнюваність, проте ціною істотно більших накладних витрат і зростання ризику дрейфу міркувань на складних задачах.

Втрати, породжені тією самою декомпозицією, доцільно розглянути за трьома напрямами.

Зростання вартості. Кожна передача проміжного результату між агентами є окремим запитом до моделі, а контекст наступного агента містить історію попереднього обміну. Оцінимо це аналітично. Нехай ланцюжок містить M агентів, постановка задачі має обсяг s, а кожен агент додає відповідь обсягом o. Якщо контекст передають без ущільнення, то i-й агент отримує $s + (i - 1)o$ токенів, а сумарний обсяг входу становить $M{\cdot}s + o{\cdot}M(M - 1)/2$, тобто зростає квадратично з числом ланок. За жорсткого обмеження контексту величиною C зростання лінійне, $M{\cdot}C$, але ціною втрати частини історії обміну. Ця оцінка є наслідком самої схеми обміну, а не властивістю конкретної реалізації; вона справджується за припущення про послідовну передачу й сталий обсяг відповіді, тому конкретні коефіцієнти залежать від протоколу передачі та стратегії ущільнення.

Зниження передбачуваності. Кожен агент є недетермінованою ланкою. Якщо ймовірність коректного проходження однієї ланки дорівнює p, а ланки вважати незалежними, то ймовірність коректної траєкторії ланцюжка з M ланок становить p^M: за $p = 0{,}9$ і $M = 4$ вона дорівнює 0,66. Припущення про незалежність є ідеалізацією, проте воно показує напрям ефекту: та сама задача за тих самих умов дає ширший розкид результатів, ніж в одноагентній системі. Значущість цього чинника підтверджують і емпіричні дані: у праці [5] серед виявлених режимів збою окрему категорію становить неузгодженість між агентами, тобто розбіжність, яка виникає саме на стиках ланок, а не всередині окремого агента.

Накопичення помилок. Наступний агент приймає хибний проміжний результат як вхідні дані. У праці [5] на основі ретельного аналізу 150 траєкторій побудовано таксономію збоїв, яка охоплює 14 режимів, згрупованих у три категорії: хиби проєктування системи, неузгодженість між агентами й недоліки перевірки завдання; узгодженість між експертами-анотаторами на цьому етапі становила 0,88 за коефіцієнтом каппа. Таксономію застосовано до набору з понад 1600 траєкторій із семи поширених мультиагентних середовищ. Виокремлення перевірки завдання в самостійну категорію збоїв показує, що йдеться про повторюваний конструктивний дефект схеми взаємодії, а не про випадкову похибку окремого агента.

Окремого зіставлення потребують дві форми організації. У кооперативній моделі агенти обмінюються результатами напряму, без центрального вузла [1]:

система лишається гнучкою, проте помилка поширюється ланцюжком, і місце її виникнення важко локалізувати. В оркестрованій моделі центральний компонент розподіляє підзадачі та приймає проміжні результати [2]: контрольованість і локалізація помилок поліпшуються, проте оркестратор стає єдиною точкою відмови та вузьким місцем за пропускною здатністю. Зіставлення показує, що оркестрація не усуває проблеми накопичення помилок, а зосереджує відповідальність за неї в одному компоненті. Це зіставлення є аналітичним: контрольовані порівняння мультиагентної архітектури з одноагентною за єдиною методикою вже виконано [6; 7], проте прямого порівняння кооперативної та оркестрованої форм між собою в розглянутих джерелах не виявлено, що само по собі вказує на прогалину.

Із зіставлення випливає умова обґрунтованості переходу. Мультиагентна архітектура виправдана тоді, коли задача справді декомпозовна на підзадачі з перевірюваними проміжними результатами, і коли для кожної передачі між агентами передбачено механізм контролю: перевірку формату, виконавчу перевірку проміжного артефакту або явну умову відхилення. Цей висновок узгоджується з даними праці [5], де недоліки перевірки завдання утворюють одну з трьох категорій режимів збою. За відсутності такого механізму декомпозиція збільшує вартість і розкид, не додаючи надійності. Тому рішення про перехід до мультиагентної архітектури слід ухвалювати разом із рішенням про механізм контролю помилок між агентами, а не окремо від нього.

Обмеження. Наведене зіставлення спирається на опубліковані емпіричні дані та не є власним експериментальним вимірюванням; воно має характер аналітичного узагальнення. Оцінки вартості й передбачуваності подано як моделі за явно вказаних припущень: послідовна передача, сталий обсяг відповіді, незалежність ланок. Тому вони показують напрям і порядок ефекту, а не універсальні константи. Кількісні співвідношення вартості й точності залежать від задачі, моделі та реалізації середовища. Джерела [6] і [7] є препринтами, тому їхні числові результати слід трактувати як попередні.

Висновки. Показано, що виграш від декомпозиції в мультиагентних архітектурах не є безумовним: розширення кола задач і спеціалізація супроводжуються зростанням обсягу оброблених токенів, квадратичним за відсутності ущільнення контексту, зниженням передбачуваності через збільшення числа недетермінованих ланок і накопиченням помилок між агентами. Зіставлення кооперативної та оркестрованої форм показало, що оркестрація повертає контрольованість ціною появи єдиної точки відмови. Сформульовано умову обґрунтованості переходу до мультиагентної архітектури: наявність механізму контролю на кожній передачі проміжного результату. Напрямом подальшої роботи є емпіричне зіставлення

одноагентної та мультиагентної конфігурацій за співвідношенням вартості й частки розв'язаних задач на спільному наборі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

[1] Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., Xu, J., Li, D., Liu, Z., & Sun, M. (2024). ChatDev: Communicative agents for software development. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 15174–15186). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.810

[2] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., & Schmidhuber, J. (2023). MetaGPT: Meta programming for a multi-agent collaborative framework. arXiv. https://doi.org/10.48550/arXiv.2308.00352

[3] Dong, Y., Jiang, X., Jin, Z., & Li, G. (2024). Self-collaboration code generation via ChatGPT. ACM Transactions on Software Engineering and Methodology, 33(7), 1–38. https://doi.org/10.1145/3672459

[4] He, J., Treude, C., & Lo, D. (2025). LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. ACM Transactions on Software Engineering and Methodology, 34(5), 1–30. https://doi.org/10.1145/3712003

[5] Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M. A., Gonzalez, J. E., & Stoica, I. (2025). Why do multi-agent LLM systems fail? In Advances in Neural Information Processing Systems 38 (NeurIPS 2025). https://doi.org/10.52202/085713-4082

[6] Xu, J., Koesdwiady, A., Bei, S., Han, Y., Huang, B., Wang, D., Chen, Y., Wang, Z., Wang, P., Li, P., & Ding, Y. (2026). Rethinking the value of multi-agent workflow: A strong single agent baseline. arXiv. https://doi.org/10.48550/arXiv.2601.12307

[7] Xu, Q., Sheng, Z., Chen, Z., & Huang, J. (2026). A systematic study of LLM-based architectures for automated patching. arXiv. https://doi.org/10.48550/arXiv.2603.01257