СЕКЦІЯ XIII. КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

HARNESS ЯК САМОСТІЙНИЙ ШАР ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ АГЕНТНИХ СИСТЕМ НА ОСНОВІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

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

Агентна система на основі великої мовної моделі (LLM) не зводиться до самої моделі. Базовий цикл роботи агента, що чергує міркування й дію, описано в парадигмі ReAct [1], проте між розробником і моделлю діє ще й програмна обв'язка, яку в англомовній літературі називають harness: вона формує системний запит, надає інструменти та виконує їхні виклики, керує вмістом контекстного вікна, задає політику повторних спроб, ізолює небезпечні дії в пісочниці, протоколює траєкторію виконання.

Те, що конструкція цієї обв'язки впливає на результат, показано ще в праці [2]: спеціалізований інтерфейс між агентом і середовищем істотно змінює здатність тієї самої моделі редагувати файли, орієнтуватися в репозиторії та запускати тести. Попри це обв'язку й досі не виокремлюють у самостійну одиницю аналізу та звітування: у систематичному огляді застосувань агентних систем в інженерії програмного забезпечення [3] її не подано як окремий чинник. Практичний наслідок такий: якщо якість приписують виключно моделі, то результати вимірювань переносять з однієї конфігурації на іншу як властивість моделі, а зміни в обв'язці не фіксують у звітності взагалі. Актуальності питанню надає темп змін: за даними праці [4], поширені відкриті обв'язки випускають понад два оновлення на добу й накопичують тисячі звернень за кілька місяців, тобто змінюються значно частіше, ніж оновлюється наукова звітність про них.

Аналіз останніх досліджень свідчить, що таке перенесення не є обґрунтованим. У праці [4] виконано контрольований поздовжній експеримент, у якому, на відміну від попередніх робіт, модель зафіксовано, а змінювали лише обв'язку: на 35 послідовних випусках однієї обв'язки (Qwen Code CLI) за 50 стратифікованими задачами SWE-bench Verified простежено коливання якості й пов'язано їх із конкретними змінами у вихідному коді. Практики регулярно повідомляють про погіршення якості після оновлення обв'язки, проте послідовно пояснюють його моделлю, а не самою обв'язкою. У праці [5] на 432 прогонах, що перетинають шість моделей чотирьох рівнів спроможності з трьома режимами обв'язки (полегшений, збалансований, суворий), спростовано припущення про монотонний зв'язок: для фронтирної діалогової моделі (Gemini 2.5 Flash) посилення структурованості обв'язки знижувало частку верифікованих успішних розв'язань на 29–38 відсоткових пунктів, тимчасом як для фронтирної моделі з розширеним міркуванням (Qwen3.5-122B) саме суворий режим дав найвищий результат. У праці [6] за аналізом вихідного коду 13 відкритих обв'язок побудовано архітектурну таксономію за 12 вимірами й показано, що обв'язки не зводяться до дискретних класів: стратегії керування циклом простягаються від фіксованого конвеєра до пошуку за деревом Монте-Карло, кількість інструментів коливається від 0 до 37, а стратегій ущільнення контексту виявлено сім.

Мета дослідження полягає в тому, щоб обґрунтувати виокремлення harness у самостійний архітектурний шар і показати наслідки такого виокремлення для методики оцінювання агентних систем.

Термін «проміжний шар» щодо обв'язки вжито вже в праці [4]; тут це положення розгорнуто до трьох перевірюваних ознак самостійності шару. Першою ознакою є власний предмет проєктування: обв'язка розв'язує задачі, яких модель не розв'язує взагалі, а саме добір вмісту контексту, обробку відмов інструментів, розмежування дозволених і небезпечних дій; перелік цих задач узгоджується з таксономією [6], де їх зведено до трьох рівнів, а саме до керування циклом, інтерфейсу інструментів і середовища та розподілу ресурсів. Другою ознакою є незалежний життєвий цикл: за даними праці [4] обв'язки оновлюються з високою частотою, тому за незмінної моделі система з часом поводиться інакше. Третьою ознакою є вимірюваний власний внесок: у працях [4] і [5] варіювання самої лише обв'язки за фіксованої моделі змінює

підсумкові показники якості, отже, внесок обв'язки не є нульовим і не поглинається внеском моделі. Незалежним свідченням на користь третьої ознаки є праця [7], де на 217 задачах у 34 середовищах виявлено систематичний розрив між здатністю моделі розв'язати задачу та її здатністю дотриматися обмежень, заданих обв'язкою.

Ключовою для практики є немонотонність цього внеску. Природне припущення полягає в тому, що суворіша обв'язка дає надійніший результат, а спроможнішій моделі потрібно менше структурного скерування. Дані праці [5] спростовують обидві частини припущення: напрям впливу залежить від типу моделі, причому для передової діалогової та для передової міркувальної моделей він протилежний. Водночас автори [5] прямо застерігають, що кожен рівень спроможності представлено в їхньому експерименті однією моделлю, тому спостереження слід трактувати як специфічні для моделі. З огляду на це висновок формулюємо обережно: з наявних даних не випливає, що суворіша обв'язка краща, отже, посилання на таку залежність не може бути підставою для перенесення конфігурації між моделями.

Практичні наслідки для методики оцінювання такі. По-перше, одиницею оцінювання є пара «модель і обв'язка», а не модель окремо: показник, здобутий в одній обв'язці, не характеризує модель поза нею. По-друге, конфігурацію обв'язки не можна переносити між моделями без повторного вимірювання, оскільки напрям впливу залежить від моделі [5]. По-третє, у звітності про експерименти з агентами слід зазначати версію обв'язки нарівні з версією моделі; без цього результат не відтворюваний, бо обв'язка змінюється частіше за модель [4]. По-четверте, порівняння двох моделей є коректним лише за тотожної обв'язки, а порівняння двох обв'язок коректне лише за тотожної моделі. Внеском цієї роботи є не сам факт вимірювання внеску обв'язки, а зведення наслідків у перелік правил звітування, з якого дві останні вимоги, тобто умови тотожності при попарному порівнянні, у розглянутих джерелах явно не сформульовані.

Обмеження. Праці [4–6] на момент підготовки роботи є препринтами й не пройшли рецензування, тому їхні числові результати слід трактувати як попередні; прорецензованими є джерела [1–3] і [7]. Кожна з праць [4–6] спирається на обмежені набори моделей і задач: у [5] кожен рівень спроможності представлено однією моделлю, а сам показник здобуто на синтетичному наборі HEAT-24 із 24 задач; у [4]

контрольований експеримент виконано на одній обв'язці. Переносним є не конкретне число, а сам факт наявності власного внеску обв'язки та відсутності підтвердженої монотонної залежності. Крім того, таксономія [6] показує високу різнорідність обв'язок, через що поняття «суворіша обв'язка» потребує окремого означення в кожному дослідженні.

Висновки. Harness обґрунтовано як самостійний архітектурний шар агентної системи за трьома ознаками: власний предмет проєктування, незалежний життєвий цикл і вимірюваний власний внесок у якість. Показано, що цей внесок немонотонний, тобто посилення структурованості обв'язки не гарантує кращого результату й для окремих моделей погіршує його. Практичним наслідком є визнання пари «модель і обв'язка» неподільною одиницею оцінювання та вимога зазначати версію обв'язки нарівні з версією моделі. Напрямом подальшої роботи є розроблення мінімального переліку параметрів обв'язки, обов'язкових до оприлюднення в емпіричних дослідженнях агентних систем.

Список використаних джерел:

1. Yao S., Zhao J., Yu D., Du N., Shafran I., Narasimhan K., Cao Y. ReAct: Synergizing Reasoning and Acting in Language Models. The Eleventh International Conference on Learning Representations (ICLR 2023). 2023. DOI: 10.48550/arXiv.2210.03629.
2. Yang J., Jimenez C. E., Wettig A., Lieret K., Yao S., Narasimhan K., Press O. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. Advances in Neural Information Processing Systems 37 (NeurIPS 2024). 2024. DOI: 10.52202/079017-1601.
3. He J., Treude C., Lo D. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. ACM Transactions on Software Engineering and Methodology. 2025. Vol. 34, No. 5. P. 1–30. DOI: 10.1145/3712003.
4. Ben Sghaier O., Li H., Adams B., Hassan A. E. Don't Blame the Large Language Model: How Agent Harness Evolution Shapes Coding Agent Quality. arXiv preprint arXiv:2607.03691. 2026. DOI: 10.48550/arXiv.2607.03691.
5. Cho Y.-E. It's Not the Capability: Harness Sensitivity Is Non-Monotone Across LLM Agent Tiers. arXiv preprint arXiv:2605.26731. 2026. DOI: 10.48550/arXiv.2605.26731.
6. Rombaut B. Inside the Scaffold: A Source-Code Taxonomy of Coding Agent Architectures. arXiv preprint arXiv:2604.03515. 2026. DOI: 10.48550/arXiv.2604.03515.
7. Ding D., Liu S., Yang E., Lin J., Chen Z., Dou S. et al. OctoBench: Benchmarking Scaffold-Aware Instruction Following in Repository-Grounded Agentic Coding. Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2026. P. 5958–5978. DOI: 10.18653/v1/2026.acl-long.269.