УДК 004.413.5:004.415.28

Методы оценки архитектур ПО, синтезированных с помощью LLM

Безрученко А. Ю., Кравец А. Г.

Волгоградский государственный технический университет

г. Волгоград

В последние годы большие языковые модели (LLM) получили широкое распространение, в том числе в разработке программного обеспечения: они генерируют код по уже спроектированной архитектуре, проектируют архитектуру по описанным требованиям и восстанавливают её по существующему коду [1]. Часть архитектурных решений тем самым принимается этими моделями, что делает оценку таких решений самостоятельной задачей.

Мы проанализировали 42 работы, в которых оценивается вывод языковых моделей применительно к архитектуре или её представлению, и установили, что в них систематически подменяется объект оценки. Результат либо сопоставляется с эталонным артефактом — по совпадению имён элементов и по функциональным тестам [2], — либо оценивается качество диаграммы классов как таковой: применяются шкалы с межоценщиковой надёжностью, метрики связности и правила проектирования, но ни один критерий не выходит за пределы артефакта, и сами авторы отмечают, что улучшения диаграммы не доказывают выигрыша в рантайм-характеристиках [3]. Ни в одной из рассмотренных работ не показано, что величина, измеренная на представлении, согласуется с качеством архитектуры.

Отдельная находка касается производительности: под ней измеряется трудоёмкость и стоимость самого LLM-конвейера, тогда как производительность спроектированной системы не измеряется ни разу, хотя инструментарий её предсказания по модели архитектуры существует и LLM не требует [4; 5].

Из этих находок вытекают требования к методу оценки.

1. Метод должен явно указывать, к какому классу артефактов он применим: архитектура или представление архитектуры. Если метод применяется к представлению, должно быть показано, что измеряемая величина на нём согласуется с качеством архитектуры.

2. Архитектура должна проверяться на соответствие требованиям, вытекающим из её назначения. Сегодня ни модельная оценка производительности, ни моделирование угроз, ни многокритериальный анализ не применяются ни разу, поэтому заявленные атрибуты остаются декларациями.

3. Метод должен быть воспроизводимым: необходимо описать, кто оценивает архитектуру — эксперт, LLM или автоматическая метрика, — по какой шкале и как разрешаются расхождения; при нескольких метриках описать веса агрегации, а при использовании LLM — процедуру калибровки их оценок человеком-экспертом. Элементы этой процедуры встречаются в корпусе по отдельности: двусторонняя слепая оценка с критерием согласованности $\kappa \geq 0.60$ [3], доверительный интервал на 272 представлениях [8], — а в одной из работ в разделе ограничений прямо признано, что оценки LLM-судьи не сверялись с экспертными оценками [9]. Но ни одна работа не собирает их в один протокол.

4. Оценка должна сопровождаться мерой неопределённости: объёмом выборки и, для вероятностных и агрегированных мер, доверительным интервалом. Без этого нельзя судить о значимости различий между моделями. Для детерминированных формальных проверок это требование неприменимо, поскольку их результат точен [6; 7]; для них вместо интервала требуется обоснование границ применимости.

Проверка применимости существующих методов проведена по инвентарю из 20 семейств и 92 метода оценки качества. В исследованном корпусе применены 8 методов, ещё 15 — лишь частично, обычно в урезанном виде или силами смежного инструмента, 69 не применены вовсе, а 7 семейств из 20 не применены ни разу. Ни один из восьми применённых методов не удовлетворяет

всем четырём требованиям, а приведённые доли пробелов являются нижней оценкой. Это указывает на необходимость адаптации существующих методов, а не их переноса, и предложенные требования могут служить спецификацией такого метода.

Список использованной литературы

1. Esposito, M. Generative AI for Software Architecture. Applications, Challenges, and Future Directions / M. Esposito, X. Li, S. Moreschini, N. Ahmad, T. Cerny, K. Vaidhyanathan, V. Lenarduzzi, D. Taibi // arXiv preprint. — 2025. — arXiv:2503.13310.

2. He, X. From What to How: Bridging User Requirements with Software Development Using Large Language Models / X. He, R. Chen, J. Cao // arXiv preprint. — 2026. — arXiv:2602.13611.

3. Liang, J. UML Class Diagram Evaluation and Repair Strategies based on LLMs / J. Liang, P. Liang, C. Wang // arXiv preprint. — 2026. — arXiv:2608.28800.

4. Becker, B. The Palladio Component Model for Model-driven Performance Prediction / B. Becker, S. Koziolek, R. Reussner // Journal of Systems and Software. — 2009. — Vol. 82, № 1. — P. 3–22.

5. Feiler, P. The Architecture Analysis & Design Language (AADL): An Introduction / P. Feiler, D. Gluch, J. Hudak. — Carnegie Mellon University, 2006. — CMU/SEI-2006-TN-011.

6. Jackson, D. Software Abstractions: Logic, Language, and Analysis / D. Jackson. — MIT Press, 2002.

7. Clarke, E. M. Model Checking / E. M. Clarke, O. Grumberg, D. A. Peled. — MIT Press, 1999.

8. Li, M. Benchmarking Requirement-to-Architecture Generation with Hybrid Evaluation / M. Li, L. Zhang, S. Yan, F. Liu, L. Li, Y. Liu, X. Lian // arXiv preprint. — 2026. — arXiv:2604.06683.

9. Szczepanik, K. Collaborative LLM Agents for C4 Software Architecture Design Automation / K. Szczepanik // arXiv preprint. — 2025. — arXiv:2510.22787.