УДК 004.415.2:004.8

ІНЖЕНЕРНІ ПАТЕРНИ ПРОЄКТУВАННЯ ТА ВІЗУАЛІЗАЦІЙНІ ПІДХОДИ В СИСТЕМАХ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ

Кроть В.С.
vktr244@gmail.com
НТУУ «Київський політехнічний інститут
ім. Ігоря Сікорського»
Марченко С. В.
Київ, Україна

Генеративний штучний інтелект стрімко інтегрується у прикладні програмні продукти: чат-боти, інструменти генерації коду та агентні системи, здатні виконувати складні багатокрокові задачі. Разом із тим розробники систематично стикаються з повторюваними інженерними проблемами – нестабільністю відповідей моделей, високими витратами обчислювальних ресурсів, галюцинаціями, складністю контролю поведінки та забезпеченням безпечності результатів генерації.

В експлуатаційній практиці такі проблеми проявляються як труднощі масштабування GenAI-рішень, обмежена відтворюваність результатів і потреба у додаткових механізмах моніторингу та аналізу процесів генерації. Повторюваний характер цих викликів свідчить про формування нового класу архітектурних рішень – патернів проєктування GenAI-систем, які забезпечують інтеграцію недетермінованих когнітивних сервісів у традиційні програмні архітектури.

Проаналізувати інженерні патерни проєктування систем генеративного штучного інтелекту, визначити їхню роль у підвищенні надійності та ефективності програмних продуктів, а також дослідити можливості використання візуалізаційних підходів для контролю та оптимізації процесів генерації.

Патерни GenAI-систем формують інженерний шар взаємодії з недетермінованими моделями машинного навчання, що зміщує фокус проєктування з організації внутрішньої структури коду на управління

контекстом виконання, трасування процесів генерації та оптимізацію ресурсних характеристик системи.

Промпт-орієнтовані підходи, зокрема Chain-of-Thought, Few-Shot Prompting і Role Prompting, спрямовані на структуризацію запиту до мовної моделі з метою підвищення точності відповідей [1]. Вони фактично виконують функцію декларативного програмування поведінки моделі. Водночас використання складних промпт-патернів призводить до збільшення довжини контексту, що негативно впливає на затримки виконання та вартість обробки запитів у виробничих середовищах. Таким чином виникає інженерний компроміс між стабільністю результату та ефективністю використання обчислювальних ресурсів.

Ефективність GenAI-систем значною мірою визначається управлінням контекстним вікном моделі. Практичними рішеннями є резюмування попередніх повідомлень, кешування системних інструкцій та декомпозиція задач на незалежні підзапити. У масштабних системах такі механізми реалізуються у вигляді окремого програмного шару керування контекстом, що дозволяє балансувати між якістю генерації та ресурсною ефективністю.

Патерн роботи зі знаннями Retrieval-Augmented Generation забезпечує підвантаження релевантних документів із зовнішніх сховищ у контекст моделі, що дозволяє зменшити ризик галюцинацій і підвищити актуальність відповідей [2]. Разом із тим RAG-архітектури породжують нові виклики, пов’язані з вибором стратегій пошуку, синхронізацією джерел знань та необхідністю візуального аналізу залежностей між retrieved-контекстом і результатами генерації.

Оскільки мовні моделі не мають довготривалої пам’яті, архітектори GenAI-систем реалізують її програмно. Короткотривала пам’ять (STM) представлена контекстним вікном, тоді як довготривала (LTM) організовується через векторні сховища embeddings. В агентних системах використовується також епізодична пам’ять – структурована інформація про виконані дії та їхні результати. Інженерна реалізація таких механізмів часто передбачає ведення

журналів виконання та використання засобів візуалізації історії взаємодії агента з середовищем.

Навички (skills) у GenAI-системах можуть розглядатися як програмно організовані модулі поведінки, що визначають способи виконання типових задач мовною моделлю або агентом. Такі модулі можуть включати інструкції, приклади використання, обмеження та опис доступних інструментів, що дозволяє повторно застосовувати їх у різних сценаріях генерації. Використання skills сприяє зниженню складності системного промпту, підвищує керованість поведінки агента та наближає GenAI-архітектури до модульних підходів класичного програмування. Подібні підходи до організації дій та використання інструментів у мовних моделях розглядаються в роботах, присвячених агентним архітектурам і взаємодії reasoning-та acting-компонентів [5].

Патерни безпеки та експлуатаційного контролю Guardrails, sandboxing і human-in-the-loop забезпечують передбачуваність поведінки моделей у критичних сценаріях [3]. Їх застосування супроводжується впровадженням інструментів моніторингу та візуалізації потоків генерації, що дозволяє інженерам виявляти небезпечні або некоректні сценарії роботи системи.

На відміну від традиційних програмних систем, де візуалізація переважно використовується для опису структурних залежностей компонентів, у GenAI-архітектурах вона виконує функцію інженерного контролю над динамікою генеративних процесів та управління якістю інтеграції мовних моделей у програмні продукти [3]. Одним із ключових підходів є візуалізація конвеєрів генерації, що відображає послідовність стадій обробки запиту: підготовку контексту, retrieval, генерацію відповіді, постобробку та валідацію. Представлення таких процесів у вигляді sequence-діаграм або графів виконання дозволяє інженерам виявляти надлишкові виклики моделі, оптимізувати latency і зменшувати витрати токенів. Подібні підходи застосовуються у практиці побудови LLM-pipeline-архітектур та систем оркестрації генеративних задач [4].

Візуалізація потоків знань у RAG-архітектурах, де retrieval-процеси моделюються як графи залежностей між запитом, retrieved-документами та

сформованими твердженнями. Це забезпечує можливість трасування походження відповідей моделі та спрощує виявлення некоректних або нерелевантних джерел контексту. Дослідження retrieval-augmented моделей показують, що прозорість джерел контексту є критичною умовою підвищення довіри до результатів генерації [2].

У агентних системах важливу роль відіграє візуалізація поведінкових сценаріїв, що може реалізовуватися через state-machine моделі або timeline-подання виконаних дій. Такі інструменти дозволяють аналізувати прийняття рішень агентом, оцінювати ефективність використання пам’яті та виявляти зациклення або неочікувані переходи між задачами. Подібні підходи застосовуються у фреймворках автономних агентів і систем планування дій [5].

Крім того, практичного значення набуває операційна візуалізація експлуатаційних характеристик GenAI-систем, зокрема dashboards контролю latency, throughput, вартості генерації та якості відповідей. На відміну від класичних систем моніторингу, ці показники безпосередньо впливають на семантичну якість результатів, що створює нові вимоги до інженерного аналізу продуктивності. Концепції observability у розподілених системах дедалі частіше адаптуються до задач моніторингу LLM-викликів та поведінки генеративних сервісів [6]. Разом із тим використання візуалізаційних підходів має обмеження. Надмірна деталізація графів виконання або retrieval-структур може ускладнювати інтерпретацію поведінки системи, а також збільшувати витрати на інструментальну підтримку. Це вимагає формування узагальнених моделей візуалізації, які поєднують інформативність і масштабованість.

У роботі проаналізовано інженерні патерни проєктування систем генеративного штучного інтелекту, що формують окремий шар взаємодії з недетермінованими когнітивними сервісами. Показано, що застосування промпт-орієнтованих підходів, механізмів керування контекстом, інтеграції зовнішніх знань і програмної організації пам’яті дозволяє підвищити передбачуваність поведінки моделей, оптимізувати використання обчислювальних ресурсів та забезпечити масштабованість GenAI-рішень у

виробничих середовищах. Обґрунтовано інженерну роль візуалізаційних підходів як інструменту аналізу та оптимізації генеративних процесів, що сприяє підвищенню прозорості архітектури, спрощує трасування сценаріїв взаємодії з моделлю та підтримує контроль експлуатаційних характеристик систем.

Список використаних джерел:

1. Wei J., Wang X., Schuurmans D., Bosma M., Ichter B., Xia F., Chi E. H., Le Q. V., Zhou D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models // NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing System. 2022. Vol. 35, Article No. 1800. P. 24824–24837.
2. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W.-t., Rocktäschel T., Riedel S., Kiela D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020. Vol. 33, Article No. 793. P. 9459–9474.
3. Subramaniam B., Fowler M. Emerging Patterns in Building GenAI Products // Martin Fowler. 2025. 25 Feb. URL: martinfowler.com/articles/gen-ai-patterns/ (дата звернення: 20.03.2026).
4. Lu Q., Zhu L., Xu X., Xing Z., Harrer S., Whittle J. Towards Responsible Generative AI: A Reference Architecture for Designing Foundation Model based Agents // IEEE Software. 2024. Volume 41, Issue 6. P. 91-100. URL: DOI: https://doi.org/10.1109/MS.2024.3406333.
5. 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). URL: openreview.net/forum?id=WE_vluYUL-X (дата звернення: 20.03.2026).
6. Dong L., Lu Q., Zhu L. AgentOps: Enabling Observability of LLM Agents // arXiv. 2024. URL: arxiv.org/abs/2411.05285 (дата звернення: 20.03.2026).