УДК 004.89

АГЕНТНЫЙ ПАЙПЛАЙН ГЕНЕРАЦИИ АРХИТЕКТУРНЫХ АРТЕФАКТОВ КАК ПРОМЕЖУТОЧНЫЙ ЭТАП МЕЖДУ ТРЕБОВАНИЯМИ И КОДОГЕНЕРАЦИЕЙ

Боцман А.Е.[1], Федотова К. Д.[1], Баталенков С. С.[1]
Научный руководитель – Филатова А.А.[1]
[1]Университет ИТМО
botsman_artem@mail.ru

Работа выполнена в рамках темы НИР №625134 «Исследование и разработка фронтирных методов искусственного интеллекта и их приложений»

Введение

Быстрый прогресс в области больших языковых моделей привел к появлению мультиагентных систем (МАС), которые могут автоматизировать отдельные этапы жизненного цикла разработки программного обеспечения [1-4]. При этом в практике многих решений сохраняется разрыв между этапом анализа требований и этапом генерации кода. Агент-кодер получает либо слишком краткий контекст, либо вынужден полностью самостоятельно достраивать архитектурные решения. Такой подход повышает риск ошибок проектирования, галлюцинаций и непредсказуемости результата.

С инженерной точки зрения критически важно формировать до кодогенерации слой промежуточных архитектурных артефактов: структурированные требования, сценарии использования, описание модулей, их ответственности, формат взаимодействия и интерфейсные контракты. Именно эти артефакты снижают неопределенность, позволяют согласовывать решения с заказчиком и создают проверяемую основу для дальнейшей реализации. При этом, в большинстве исследований на данный момент нет устойчивой практики генерации полноценного набора архитектурных артефактов перед этапом кодогенерации. Данная работа позиционируется как практический шаг к закрытию или уменьшению этого разрыва и предлагает агентный пайплайн генерации архитектурных артефактов в контуре МАС по автоматизации процесса разработки программного обеспечения.

Основная часть

В работе рассматривается многошаговый пайплайн. На первом шаге пайплайна по переданным на вход требованиям и пользовательским сценариям происходит генерация набора архитектурных модулей с описанием назначения, покрытия функциональных и нефункциональных требований, предполагаемого технологического стека и состава компонентов. На втором шаге для сгенерированных архитектурных модулей определяется порядок их реализации и интерфейсы взаимодействия, которые выступают контрактом для агента-кодера. На третьем шаге блок архитектурных артефактов, совместно с требованиями, которые система получила на вход, выступают входом для этапа генерации визуализационных материалов в виде UML-диаграмм.

Реализация агентного пайплайна выполнена в виде набора связных графов LLM-агентов на базе LangGraph [5]. Контроль качества артефактов на всех стадиях обеспечивается несколькими механизмами: строгой типизацией промежуточных структур, итеративных циклов контроля с отдельным агентом, выполняющим роль критика и проверкой выходного формата через структурированные схемы. Генерация диаграмм производится в формате Mermaid [6], поскольку он одновременно удобен для

LLM-генерации и является подходящим для визуализации, хранения в репозитории, а также легко встраивается в инженерную документацию.

Ключевой инженерный эффект такого промежуточного слоя, в виде генерации архитектурных артефактов, заключается в снижении когнитивной нагрузки на агента-кодера. Вместо необходимости заново выводить архитектуру из исходного текстового запроса, кодогенератор получает согласованный пакет артефактов с явными границами модулей и ожидаемыми контрактами. Это снижает вероятность ошибок композиции и повышает воспроизводимость результата. UML-диаграммы, в свою очередь, позволяют наглядно провалидировать получившуюся архитектуру, межмодульное взаимодействие и покрытие требуемых пользовательских сценариев.

Таким образом, в работе предложен и реализован агентный подход к генерации архитектурных артефактов как промежуточного слоя между анализом требований и кодогенерацией. На практическом кейсе показано, что разработанный пайплайн позволяет повысить качество последующей кодогенерации за счет предоставления заранее сформированных контрактов агенту-кодеру. Предложенный подход может быть внедрен в корпоративных контурах МАС-разработки и при разработке продуктовых решений, где требуется прозрачная трассировка требований и предсказуемость инженерных решений.

Литература

1. Qian C. et al. Communicative Agents for Software Development // arXiv preprint arXiv:2307.07924. 2023.
2. Hong S., Zhuge M., Chen J. et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework // arXiv:2308.00352. 2023.
3. Wu Q., Bansal G., Zhang J. et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation // arXiv:2308.08155. 2023.
4. Yang J., Lee C., Li S. et al. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering // arXiv:2405.15793. 2024.
5. LangGraph Documentation. URL: https://langchain-ai.github.io/langgraph/
6. Mermaid Documentation. URL: https://mermaid.js.org/intro/getting-started.html