УДК 004.89

ИТЕРАТИВНАЯ САМОКРИТИКА В МУЛЬТИАГЕНТНОМ LLM-ПАЙПЛАЙНЕ ГЕНЕРАЦИИ СПЕЦИФИКАЦИЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Пышкин Д.А.[1], Куприянова И. Л.[1], Шипицын А. А.[1]
Научный руководитель – Филатова А.А.[1]
[1]Университет ИТМО
denispyshkin2007@gmail.com

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

Введение

Большие языковые модели (LLM) активно применяются для автоматизации этапов жизненного цикла разработки программного обеспечения. Ряд мультиагентных систем моделирует взаимодействие участников процесса разработки, распределяя роли между LLM-агентами. ChatDev [1] организует диалог агентов в ролях CEO, программиста и тестировщика для генерации кода через обсуждение. MetaGPT [2] вводит этап формирования Product Requirements Document и описания архитектуры перед кодогенерацией. Инструменты GPT-Engineer [3] и SWE-Agent [4] ориентированы на непосредственную генерацию или модификацию кода по пользовательскому запросу. Фреймворки AutoGen [5] и LangGraph [6] предоставляют инфраструктуру для построения мультиагентных пайплайнов с гибким управлением состоянием и маршрутизацией.

Существующие подходы к генерации проектной документации в таких системах можно разделить на несколько категорий. Одни системы не генерируют документацию, переходя от пользовательского запроса непосредственно к коду, другие формируют краткие промежуточные описания логики поведения системы, которые служат контекстом для агента-программиста, но не являются самостоятельным документом. Отсутствие качественной документации до этапа кодогенерации порождает ряд проблем. Агент-кодер, не имея чётких и непротиворечивых инструкций, вынужден самостоятельно интерпретировать неоднозначные требования, что приводит к ошибкам реализации. Без зафиксированных спецификаций невозможно сформировать приёмочные тесты до написания кода, а также теряется возможность уточнить детали и зафиксировать договорённости с пользователем-заказчиком до начала разработки, из-за чего несоответствия между ожиданиями заказчика и результатом обнаруживаются только на этапе готового продукта, когда стоимость исправлений максимальна.

Существующие исследования в области мультиагентной генерации требований демонстрируют, что разделение ролей и введение механизмов критики могут улучшать качество формируемых артефактов. Однако в таких исследованиях присутствует ряд ограничений. Отсутствует формализованный анализ накопления ошибок по стадиям пайплайна. Не исследованы границы применимости различных стратегий контроля – без критика, с самокритикой, с внешним критиком. Нет систематического сопоставления сложности задачи и необходимого уровня контроля. Практически не анализируется downstream-эффект – насколько полученная документация пригодна для последующей архитектурной проработки и кодогенерации.

Основная часть

Целью данной работы является исследование влияния уровня контроля в процессе генерации проектной документации на надёжность распространения требований и пригодность артефактов к дальнейшей реализации.

В работе рассматривается многошаговый пайплайн.

Шаг 1: диалог с пользователем, формирование функциональных требований (ФТ).

Шаг 2: генерация сценариев использования (Use Cases) на основе сгенерированных функциональных требований.

Шаг 3: генерация описания архитектурных модулей по сгенерированным функциональным требованиям и Use Cases.

В рамках данного пайплайна исследуется, как изменяются требования при переходе между стадиями при различных стратегиях контроля: без дополнительного контроля, с усиленными инструкциями генератору, с самокритикой и с отдельным агентом-критиком.

Новизна работы заключается в формализации метрик надёжности распространения требований по стадиям пайплайна, экспериментальном исследовании накопления ошибок при переходе между стадиями генерации артефактов, выявлении зависимости между сложностью задачи и необходимым уровнем контроля, а также в анализе downstream-нагрузки на этапы архитектурной и кодовой реализации.

Эксперименты проводятся на наборе задач различной сложности, различающихся по количеству требований, уровню сложности разрабатываемой системы и степени неопределённости исходного описания. Для каждой задачи пайплайн запускается в четырёх режимах контроля. На каждом этапе фиксируются значения метрик сохранности требований, инъекции неподтверждённых требований, искажения ограничений и наличия противоречий, а также анализируется влияние полученных артефактов на downstream-этап. Дополнительно анализируется характер накопления ошибок по стадиям.

Результаты работы позволяют обоснованно выбирать архитектуру мультиагентной системы автоматизированной разработки: использовать упрощённые пайплайны для малых и средних задач, вводить специализированные механизмы критики для сложных проектов и прогнозировать рост ошибок при масштабировании требований. Таким образом, работа вносит вклад в систематизацию подходов к построению надёжных мультиагентных систем автоматизированной разработки программного обеспечения.

Литература

1. Qian C. et al. Communicative Agents for Software Development // arXiv preprint arXiv:2307.07924. 2023.
2. Hong S. et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework // arXiv preprint arXiv:2308.00352. 2023.
3. GPT-Engineer. URL: https://github.com/gpt-engineer-org/gpt-engineer (дата обращения: 2025).
4. Yang J. et al. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering // arXiv preprint arXiv:2405.15793. 2024.
5. Wu Q. et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation // arXiv preprint arXiv:2308.08155. 2023.
6. LangGraph Documentation. LangChain, Inc. URL: https://langchain-ai.github.io/langgraph/ (дата обращения: 2025).