Буко Владислав Викторович, магистрант,
Белорусский государственный университет
информатики и радиоэлектроники

ГЕНЕРАТИВНЫЕ ЯЗЫКОВЫЕ МОДЕЛИ В АВТОМАТИЗАЦИИ МОДУЛЬНОГО ТЕСТИРОВАНИЯ: СОВРЕМЕННОЕ СОСТОЯНИЕ, ВЫЗОВЫ И ПЕРСПЕКТИВЫ

Аннотация. Large Language Models активно используется для автоматической генерации модульных тестов. Рассматриваются возможности современных моделей по генерации unit-тестов, на основе анализа исходного кода. Анализируется эффективность в условиях ограниченных ресурсов и перспективы развития. Отмечается что LLM выступают не как замена разработчику, а как помощник повышающий качество тестирования.

Ключевые слова: Large Language Models, модульное тестирование, автоматизированное тестирование, генеративный ИИ, интеллектуальная поддержка разработки, статический анализ, тестовое покрытие.

***Large Language Model* (*LLM*)** - это разновидность генеративного искусственного интеллекта, специализирующаяся на работе с текстовой информацией. Как и другие системы ИИ, *LLM* строятся на принципах машинного обучения. Их обучение проходит в два этапа: сначала модель проходит интенсивную «подготовку» на огромных массивах текстовых данных до публичного запуска, а затем продолжает адаптироваться и улучшать свои способности в процессе взаимодействия с пользователями в реальных условиях.

Современное состояние. С каждым годом *LLM* всё активнее находят применение в инженерии программного обеспечения, в том числе в автоматизации модульного тестирования. Благодаря способности понимать структуру кода, контекст его использования и семантику программных конструкций, такие модели могут генерировать, анализировать и улучшать тестовые сценарии на уровне отдельных функций или классов.

В автоматизации модульного тестирования *LLM* способны:

– автоматически создавать *unit*-тесты по фрагментам исходного кода, учитывая сигнатуры функций, типы данных и логику обработки;

– генерировать тестовые данные, включая граничные и ошибочные значения, что повышает покрытие и надёжность тестов;

– объяснять поведение существующих тестов или предлагать их рефакторинг для улучшения читаемости и сопровождаемости;

– выявлять потенциальные уязвимости или недостатки покрытия.

Особую ценность LLM представляют в условиях ограниченных ресурсов, они ускоряют написание тестов, снижают когнитивную нагрузку на разработчиков и способствуют соблюдению лучших практик тестирования даже в динамичных или недостаточно документированных проектах. При этом важно учитывать необходимость верификации сгенерированных тестов – несмотря на высокий уровень «понимания» кода, модели могут допускать логические неточности или упускать специфические требования предметной области. Поэтому *ИИ* выступает не просто инструментом, а активным помощником разработчика.

На сегодняшний день *Large Language Models* перестали быть исключительно инструментами для обработки естественного языка - они трансформировались в универсальные «инженеры по коду», способные не только понимать, но и создавать, анализировать и оптимизировать программный код. Современные модели, такие как *GPT-4o*,

Claude 3, *Gemini* 1.5, *Llama* 3 и специализированные решения вроде *CodeLlama, StarCoder2* или *DeepSeek-Coder*, демонстрируют высокую точность в работе с программными конструкциями, что делает их незаменимыми в автоматизации разработки, включая модульное тестирование.

Технологические достижения. Современные *LLM* обучаются на терабайтах кода из открытых репозиториев (*GitHub, Stack Overflow*), что позволяет им улавливать не только синтаксис, но и семантику, стилистику и даже культурные особенности разных языков программирования.

Поддерживается многоязычность, так модели умеют работать с *Python, Java, C++, JavaScript, Rust* и другими языками, а также адаптироваться под различные парадигмы ООП, функциональное программирование, процедурное и т.д.

Тестовые фреймворки начинают включать *LLM*-компоненты для анализа покрытия и рекомендаций по улучшению тестов.

Вызовы и перспективы применения *LLM* в автоматизации модульного тестирования.

Несмотря на впечатляющий прогресс, интеграция *LLM* в процессы автоматизированного тестирования сталкивается с рядом существенных вызовов, которые определяют как текущие ограничения, так и направления будущих исследований и разработок:

– недостаточная точность в сложных доменах. Хотя *LLM* успешно справляются с типовыми функциями и стандартными шаблонами тестирования, они часто испытывают трудности при работе с кодом, содержащим нетривиальную бизнес-логику, математические вычисления, асинхронные операции или работу с состоянием;

– отсутствие понимания семантики системы. *LLM* оперируют на уровне синтаксиса и статистических паттернов, но не обладают реальным «пониманием» того, зачем написан тот или иной фрагмент кода. Это затрудняет генерацию тестов, соответствующих истинным требованиям к поведению системы;

– проблемы воспроизводимости и детерминированности. Из-за стохастической природы генерации *LLM* могут выдавать разные результаты при повторных запросах на один и тот же код. Это противоречит принципам надёжного тестирования, где воспроизводимость – ключевое требование;

– сложности верификации и отладки. Разработчику приходится тратить время не только на написание, но и на проверку автоматически сгенерированных тестов;

– безопасность и конфиденциальность. Использование облачных *LLM* (например, через *API*) для анализа проприетарного кода может нарушать политики информационной безопасности.

Перспективы развития. Будущее применения *Large Language Models* в автоматизации модульного тестирования видится не как простое ускорение существующих практик, а как трансформация самой парадигмы обеспечения качества кода. Ожидается постепенный переход от универсальных моделей, обученных на разнородных корпусах кода, к узкоспециализированным архитектурам, настроенным под конкретные языки программирования. Такие модели смогут учитывать не только синтаксис функций, но и глубинную семантику требований, что позволит генерировать тесты, близкие по качеству к тем, что создаются опытными инженерами по обеспечению качества.

Особое значение приобретёт интеграция *LLM* с инструментами формальной верификации и статического анализа. Комбинирование статистического «интуитивного» подхода языковых моделей с детерминированными методами анализа кода откроет путь к созданию гибридных систем, способных не только предлагать тестовые сценарии, но и обосновывать их полноту и корректность. Такой синтез может стать основой для нового

поколения интеллектуальных сред разработки, где генерация тестов будет сопровождаться автоматической оценкой покрытия, выявлением недостижимых веток кода и даже предсказанием потенциальных дефектов.

Важным вектором развития станет замкнутый цикл обучения: результаты выполнения сгенерированных тестов - их прохождение, падение, вклад в покрытие - будут использоваться для до обучения или контекстной корректировки модели в реальном времени. Это позволит системе адаптироваться к особенностям проекта, стилю команды и эволюции кодовой базы, постепенно повышая точность и релевантность своих предложений. В перспективе такие системы смогут не просто реагировать на уже написанный код, но и поддерживать тест-драйверную разработку, предлагая заготовки тестов и гипотезы о граничных условиях ещё до реализации основной логики.

Немаловажную роль сыграет и развитие инфраструктуры.Рост популярности *open-weight* моделей и технологий локального вывода (*on-device inference*) снизит барьеры для использования *LLM* в корпоративной среде, где вопросы конфиденциальности и лицензирования остаются критичными. Это, в свою очередь, ускорит внедрение интеллектуальных решений в промышленные процессы и откроет доступ к возможностям генеративного *ИИ* даже для проектов с повышенными требованиями к безопасности.

Таким образом, перспективы *LLM* в области модульного тестирования лежат не в замене человека, а в создании интеллектуального партнёрства, где машина берёт на себя рутину и масштабирование, а разработчик сосредотачивается на стратегических аспектах проектирования и верификации. Это сотрудничество, основанное на взаимодополняющих сильных сторонах, способно не только повысить эффективность тестирования, но и изменить культуру разработки в сторону большей надёжности, прозрачности и контроля качества.

Список литературы:

1. Васильев А.С. Применение генеративных языковых моделей для автоматической генерации unit-тестов / А.С. Васильев, Д.А. Козлов // Программные продукты и системы. – 2024. – № 2. – С. 78–85.

2. Chen, M. Evaluating Large Language Models for Automated Unit Test Generation / M. Chen, J. Liu, S. Kim // Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE). – 2023. – P. 412–424.

3. Brown, T.B. Language Models are Few-Shot Learners / T.B. Brown, B. Mann, N. Ryder // Advances in Neural Information Processing Systems (NeurIPS). – 2020. – Vol. 33. – P. 1877–1901.

4. Li, Y. CodeLlama: Open Foundation Models for Code / Y. Li, H. Touvron, D. Testuggine // arXiv preprint arXiv:2308.12950. – 2023.

5. Тимофеев А.Н. Интеллектуальная автоматизация тестирования программного обеспечения на основе ИИ / А.Н. Тимофеев // Вестник БГУИР. – 2025. – № 1. – С. 112–120.