УДК 004.89

ЭКОНОМИЯ ЛИ? ИСПОЛЬЗОВАНИЕ СИСТЕМ ИИ ДЛЯ РАЗРАБОТКИ КОДА В СТАРТАПАХ

Губарев К.Л. (ИТМО)**, Махортова Д.В.** (ИТМО)**, Милицкая А.А**. (ИТМО),
Николаев Н.А. (ИТМО)
Научный руководитель: кандидат технических наук, ординарный доцент ФПИн Фёдоров Д.А (ИТМО)

Введение. На 2025 год 33% всех запросов в Claude Code, специализированной LLM (большой языковой модели) для написания кода, были связаны с разработкой кода для стартапов против 13%, идентифицированных как корпоративные [1], что отражает широкое использование систем искусственного интеллекта (ИИ) для разработки кода среди стартаперов. Это позволяет сэкономить ресурсы на оплату труда разработчиков, но в то же время создаёт проблемы с надёжностью кода.

Основная часть. Сравнение затрат на разработку типового программного модуля (CRUD + авторизация, 40 человеко-часов без ИИ) показывает, что применение генеративных ИИ-ассистентов статистически связано с ростом продуктивности разработчиков на 20–55% в зависимости от условий эксперимента [2–3] В рандомизированном исследовании GitHub Copilot время задач сократилось на ≈55% [2], а согласно полевому исследованию Becker et al. [3], средний прирост продуктивности опытных разработчиков при использовании современных ИИ-ассистентов составляет 26%. Такой эффект означает, что фаундер, использующий ИИ, может закрыть тот же объём задач быстрее, чем без него, что приводит к снижению прямых денежных затрат на этапе MVP.

С другой стороны код, сгенерированный ИИ, имеет свои структурные особенности, дефекты и уязвимости. Исследование на базе БГУИР [4] (1000 файлов с кодом, написанным человеком (КНЧ), и более 900 файлов, сгенерированных ChatGPT 4o и o3-mini-high) показало, что длина человеческих файлов варьируется от 0 до более чем 6000 строк, тогда как код, сгенерированный ИИ ограничен диапазоном 0–400 строк с пиком 50–100 строк. В противоположность этому, код, сгенерированный ИИ, имеет жестко ограниченный диапазон длины — от 0 до 400 строк, причем наибольшее количество файлов сосредоточено в диапазоне 50–100 строк [4, Рис. 1б, с. 50]. Это указывает на склонность ИИ к использованию стандартизированных, шаблонных решений и избеганию сложных алгоритмических конструкций [4, с. 49].

Масштабное исследование [5] с участием трех современных LLM (ChatGPT, DeepSeek-Coder, Qwen-Coder) показывает, что Python человеческий код имеет 158 тыс. дефектных файлов (при общем числе дефектов 429 тыс.), в то время как модели показывают худшие результаты: Qwen-Coder — 178 тыс. дефектных файлов (575 тыс. дефектов), DeepSeek — 174 тыс. (320 тыс. дефектов). ChatGPT показал себя лучше среди ИИ-моделей — 111 тыс. дефектных файлов (185 тыс. дефектов) [5]. В Java картина иная: человеческий код чище (76 тыс. дефектных файлов, 202 тыс. дефектов), а все ИИ-модели демонстрируют значительно большее число дефектных сэмплов (например, DeepSeek: 155 тыс. дефектных файлов, 242 тыс. дефектов) [5]. Особого внимания заслуживают результаты, касающиеся безопасности. Анализ с помощью статического анализатора Semgrep, проведенный в рамках того же исследования [5], выявил, что ИИ-код содержит значительно большее количество уязвимостей высокого риска (high-risk), классифицируемых по Common Weakness Enumeration (CWE). В ИИ-коде на Python обнаружено на 5 тыс. больше таких уязвимостей, а на Java — на 18 тыс. больше, с

фокусом на использовании небезопасных API (unsafe APIs) и инъекциях (injection flaws) [5]. Это коррелирует с выводами экспертов из CSO Online [6], которые отмечают, что ИИ-ассистенты, такие как GitHub Copilot, генерируя код без полного понимания контекста приложения, могут усиливать глубокие архитектурные уязвимости (deeper vulnerabilities) [6].

Вывод. Таким образом, наиболее эффективным способом разработки кода с ИИ является совместное использование труда LLM и профессионального разработчика, где ИИ выступает ассистентом, ускоряющим рутинные операции и генерацию базовых структур, а профессиональный разработчик обеспечивает архитектурный контроль, ревью кода, безопасность и рефакторинг. Это значительно повышает скорость разработки, сохраняя качество написанной программы [3], что значительно снижает затраты на исправление ошибок.

Список использованных источников

1. Anthropic Economic Index: AI’s impact on software development // Anthropic – URL: https://www.anthropic.com/research/impact-software-development (дата обращения: 27.02.2026).
2. Peng, S. et al. (2023). The Impact of AI on Developer Productivity: Evidence from a Randomized Controlled Trial. https://arxiv.org/abs/2302.06590
3. Becker, J., Rush, N., Barnes, E., Rein, D. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity // arXiv. - Ithaca: Cornell University, 2025. - 2507.09089.
4. Желтко Ю.Ю., Одинец Д.Н. Сравнительный анализ программного кода, написанного человеком и сгенерированного ИИ // 61-я научная конференция аспирантов, магистрантов и студентов БГУИР. – Минск: БГУИР, 2025. – С. 49-52.
5. CSO Online. AI coding assistants amplify deeper cybersecurity risks. – 2025, September 25. https://www.csoonline.com/article/4062720/ai-coding-assistants-amplify-deeper-cybersecurity-risks.html
6. Pendyala V. S., Thakur N. B. Performance and interpretability analysis of code generation large language models // Neurocomputing. – 2025. – Vol. 656. – 131461