УДК 004.896

Пущенко Александр Денисович, студент, Российский технологический университет Московский институт радиотехники, электроники и автоматики, г. Москва

«ВАЙБ-КОДИНГ» ИЛЛЮЗИЯ ПРОДУКТИВНОСТИ С РАЗРУШИТЕЛЬНЫМИ ПОСЛЕДСТВИЯМИ ДЛЯ ИИ И ДАННЫХ

Аннотация: Статья посвящена феномену «вайб-кодинга» – нового подхода к разработке ПО, при котором программист задаёт задачи на естественном языке, а код генерирует ИИ. Определяется связь вайб-кодинга с концепциями no-code и визуального программирования. Приводится анализ иллюзии продуктивности при поверхностном программировании и безусловном доверии ИИ-сгенерированному коду. Обсуждается рост технического долга и падение качества ПО при широком использовании таких методов. Рассматриваются изменения в профессии разработчика и инженерной культуре: как переход от низкоуровневого кодирования к высокоуровневому управлению ИИ меняет профессиональные навыки. Особое внимание уделяется рискам для прогресса в области ИИ из-за отвлечения специалистов от глубоких исследований, а также угрозам для data science и аналитики данных при некритичном использовании AI-инструментов. Приводятся примеры из публицистики и исследований (Habr, Forbes, IEEE/ArXiv и др. 2023–2025 гг.), мнения экспертов. Показано, что без инженерной дисциплины и надлежащего контроля «вайб-кодинг» может привести к завалу проектов, уязвимостям и искажённым данным. Сделан вывод о необходимости ответственного подхода: ИИ – лишь инструмент, который требует опыта, тщательного тестирования и осознанного применения.

Ключевые слова: вайб-кодинг, искусственный интеллект, продуктивность, технический долг, качество ПО, профессия программиста, инженерная культура, data science.

"Vibe coding" is an illusion of productivity with devastating consequences for AI and data.

Abstract: The article explores the phenomenon of “vibe coding” – an emergent approach where programmers describe tasks in natural language and AI generates the code. We define its relation to no-code and visual programming paradigms. We analyze the illusion of productivity resulting from superficial programming and blind trust in AI-generated code. The growth of technical debt and decline in software quality under such practices are examined. The impact on the software engineering profession and culture is discussed: how the shift from low-level coding to AI-driven development changes professional skills. Special attention is paid to risks for AI research progress due to diversion from deep investigation, as well as threats to data science and analytics when AI tools are used uncritically. Examples from recent publications (Habr, Forbes, IEEE/ArXiv 2023–2025) and expert opinions are given. It is shown that without engineering discipline and proper oversight, vibe coding can lead to project failure, security flaws and skewed data. We conclude with a call for responsibility: AI is only a tool that demands expertise, thorough testing, and conscientious use.

Keywords: vibe coding, artificial intelligence, productivity illusion, technical debt, software quality, programming profession, engineering culture, data science.

Введение

Понятие «вайб-кодинг» (от англ. *vibe coding*) введено в начале 2025 г. и быстро стало обсуждаемым в IT-сообществах[1]. Его суть – разработчик описывает задачу на естественном языке, а ИИ (LLM) генерирует код по этому описанию. Forbes определяет вайб-кодинг как «новый подход к программированию, при котором разработчик описывает задачи на естественном языке, а ИИ генерирует соответствующий код»[1]. По сути, это похоже на парное программирование, где напарником выступает не человек, а большая языковая модель[1]. Известные специалисты отмечают, что вайб-кодинг «не совсем программирование – ты просто говоришь что-то, запускаешь, копируешь-вставляешь, и всё в основном работает»[1]. На Habr уточняют: «Это не low-code и не no-code… Это диалог с ИИ, который генерирует код по текстовому

[1]Pshinnik K. Программирование по вайбу: как AI-ассистенты меняют индустрию разработки. Forbes, 29.04.2025.

описанию»[2]. Иными словами, вайб-кодинг сочетает высокоуровневый ввод на естественном языке (prompt engineering) с генерацией кода, но конечный продукт остаётся кодом, который нужно запустить и отладить.

В некотором смысле вайб-кодинг продолжает тенденцию упрощения разработки: наряду с no-code и визуальным программированием он снижает порог входа. Например, IBM подчёркивает, что такие инструменты позволяют даже пользователям без навыков программирования быстро создавать прототипы приложений[3]. Стартапы и образовательные проекты (университет «Зерокодер» и др.) уже учат студентов «программированию на Python с AI», где фокус смещён с синтаксиса на формулировку задач ИИ[3]. Таким образом, вайб-кодинг можно рассматривать как логическое развитие no-code/low-code: входной барьер ещё ниже, код «пишет» ИИ. Однако за этой привлекательностью скрываются серьёзные проблемы и риски, к которым мы перейдём далее.

Иллюзия продуктивности и доверие к ИИ-коду

При поверхностном программировании с помощью ИИ может возникнуть «иллюзия продуктивности»: программирование идёт очень быстро, но качество результата оказывается сомнительным. Индустриальные отчёты уже фиксируют тревожные эффекты: исследование GitClear за 2020–2024 гг. обнаружило резкий рост дублирования кода в проектах с AI-copilot, вплоть до 4-кратного увеличения количества «copy/paste» блоков по сравнению с традиционным рефакторингом[4]. Другими словами, вместо того чтобы написать и отлаживать качественную логику, многие разработчики просто принимают код от ИИ «как есть», что приводит к клонированию и фрагментированию проекта.

Важным фактором является слепое доверие ИИ. A. Shmueli (JIT) описывает «гало-эффект авторитета» AI: часто код, сгенерированный ИИ, воспринимается как безусловно правильный и безопасный[5]. Программисты,

[2] Osaka (Mushchuk S.) Вайб-кодинг с ИИ: разработка без кода или шаг в бездну? Habr.com, 18.04.2025.
[3] IBM. What is Vibe Coding? IBM Blog, 2023.
[4] GitClear. AI Copilot Code Quality: 2025 Research Report. 2025.
[5] Shmueli A. AI-Generated Code: The Security Blind Spot Your Team Can’t Ignore. JIT, 02.04.2025.

особенно неопытные, могут попросту копировать ответы ИИ, не понимая их сути – возникает «разрыв понимания» (comprehension gap) между тем, что деплойят, и тем, что реально понимают разработчики[5]. Исследователь C. Penn демонстрирует на практике опасность «слепого доверия» к AI-ответам: при копировании результатов Google AI Overview в презентации легко закрались критические ошибки, о чём он предупреждает как о «рискованной привычке»[6].

Быстрый старт не гарантирует качества. Анализ Habr показывает, что при слепом вайб-кодинге (когда разработчик пускается в диалог с ИИ без анализа и проверки) проект запускается за минуты, но в итоге качество оценивается как «авось — ой» с «катастрофическим» техническим долгом[2]. В то же время «вайб с инженерной дисциплиной» требует больше времени на старт (примерно полчаса), но даёт хорошо поддерживаемый код и низкий долг[2]. Таким образом, «иллюзия быстроты» может привести к тому, что проекту придётся тратить в разы больше усилий на исправление, чем сэкономлено на первоначальном «быстром» коде.

Кроме того, эксперты отмечают, что ИИ склонен давать лаконичные, но потенциально уязвимые решения: фокусируясь на скорости, модели зачастую повторяют небезопасные паттерны из обучающих данных[5]. Они практически не учитывают требования безопасности и специфические контексты применения, если о них не попросили явно[5]. Ошибки в интеграции кода, нарушение архитектурных принципов и отсутствие тестов могут «скрываться» за красивой внешней функциональностью. Итог: мнимая производительность сменяется катастрофой стабильности и безопасности.

Технический долг и качество ПО.

Все описанные процессы неизбежно порождают технический долг. AI-сгенерированный код, особенно при слепом приёме, часто имеет нелинейную структуру и дублирующие блоки[4]. К нему требуется тщательная оптимизация.

[6] Penn C. Mind Readings: Blind Trust in AI Overviews. TrustInsights.blog, 04.2025.

IBM подчёркивает, что код из «вайба» нужно доводить до ума: он может иметь «прототипный» статус, требуя рефакторинга, а иногда не подходит для масштабируемых и распределённых систем без глубокой переработки[5]. Без должной архитектуры, документации и тестов многократное повторение запросов к ИИ приводит к росту «халтурного» кода.

Как показывает практика, автоматизация мелких задач с помощью ИИ высвобождает время, но увеличивает объём рутины в долгосрочной перспективе. Так, разработчики отмечают, что с ростом использования AI-помощников удлиняется «быстрый код», но ускоряется накопление долгов и ошибок. В отчёте CIO прямо сказано: «AI-инструменты формируют то, как разработчики борются с техническим долгом, однако человеческое суждение всё ещё необходимо для понимания контекста»[7]. Иными словами, автоматический рефакторинг и генерация кода могут помочь снизить скуку, но не заменят взвешенный инженерный подход.

Влияние вайб-кодинга на качество ПО частично подтверждают эмпирические данные. Как уже упоминалось, GitClear отмечает всплеск дублей кода и короткосрочной «горячей» правки, при том, что доля переиспользования кода («moved code») снижается[4]. Это означает, что команды всё чаще предпочитают копировать и править, а не архитектурно улучшать. Увеличение короткоживущего кода осложняет понимание системы и будущую поддержку. Получается, что даже если текущая версия «собралась быстрее», то её дальнейшая доработка будет идти медленнее.

Кроме технического долга, повышается риск скрытых уязвимостей. Если за генерацией кода не следует тщательное ревью, многие распространённые ошибки из открытых источников мигрируют в новые проекты[5]. Часто формулировки запросов не учитывают нюансы архитектуры приложения, так что код работает только в узком контексте. По сути, разработчики внедряют блоки кода, которых не понимают, и уже не видят их логики. Без осознанной

[7] Fruhlinger J. Can AI solve your technical debt problem? CIO, 29.04.2025.

проверки безопасность страдает: IBM обращает внимание, что такого рода код «часто исключён из ревью и проверок безопасности»[5] и «легко может содержать уязвимости, остающиеся незамеченными»[5].

Все эти факторы означают, что «скорость ради скорости» может оказаться ложной экономией. Парадоксально, но инструмент, созданный для экономии времени, может потребовать ещё больше ресурсов на исправление последствий своей работы. В конце концов, как замечено в индустрии, «тот, кто не читает код, рано или поздно будет читать логи»[2], – альтернативный, но горький итог слепого принятия AI-кода.

Влияние на профессию разработчика и инженерную культуру.

Развитие вайб-кодинга меняет ожидания к роли программиста. С одной стороны, оно упрощает некоторые задачи: начинающие без глубоких знаний языков программирования уже могут прототипировать приложения, как отмечают эксперты Forbes[1]. Анализ ArXiv подчёркивает, что вайб-системы «процветают на этапах прототипирования и образования»[8], позволяя фокусироваться на концепциях и архитектуре вместо низкоуровневого синтаксиса.

Однако вместе с этим меняются профессиональные требования. Возникает потребность в новых навыках: умение формулировать задачи (prompt engineering) становится столь же важным, как умение пользоваться Git или SQL[2]. Специалисту всё более требуется мыслить не в строках кода, а в намерениях, архитектуре и верификации результатов ИИ. Habr подчёркивает, что такой подход требует «опыта и здравого смысла»: ИИ «не альтернатива инженерии, а её дополнение»[2]. Другими словами, роль разработчика смещается от писателя строк кода к архитектору, наставнику и контролёру AI-процессов.

Сами программисты реагируют по-разному. Опрос Evans Data Corporation показывает, что около 30 % разработчиков полагают, что их задачи в

[8] Sapkota R., Roumeliotis K., Karkee M. Vibe Coding vs. Agentic Coding: Fundamentals and Practical Implications of Agentic AI. Cornell U., arXiv:2505.19443, 2025

перспективе может заменить AI[9]. Это вызывает тревогу и неопределённость: будет ли профессия более востребована или устареет? Одновременно компании и университеты уже начинают внедрять обучение сотрудничеству с ИИ, подчёркивая, что бездумное «отключение мозга» недопустимо. Один из авторов (Forbes) отмечает, что вайб-кодинг даёт рабочие прототипы «даже людям без глубоких знаний»[1], но задаётся вопросом, какие профессии останутся после этого.

В итоге культура разработки должна эволюционировать. Необходимо вырабатывать новые практики: проверка и тестирование AI-кода, парное проектирование с AI, код-ревью и автоматические анализаторы (SAST/DAST) должны стать обязательными. Как резюмирует один из авторов Habr: «ИИ – не зло и не спасение. Это новый класс инструментов, мощный и удобный. Но он требует опыта и здравого смысла»[2]. Именно инженерная дисциплина (безопасность, поддерживаемость, тесты, документация) станет главным отличием профессионального подхода от хайпа.

Угроза прогрессу ИИ из-за замедления глубинных исследований.

Повсеместное увлечение генеративными инструментами может отвлекать ресурсы от фундаментального развития ИИ. Если большая часть усилий проектных команд и стартапов уходит на быстрые продукты на основе «черного ящика», то рискну цитировать идею «Большого Забвения»: поверхностные решения превращают обучение специалистов в потребление готовых ответов, а не в глубокое понимание алгоритмов. Хотя прямых научных публикаций на эту тему немного, уже звучат предупреждения: чрезмерная зависимость от AI-инструментов потенциально ведёт к «утрате навыков критического мышления и фундаментальной грамотности» среди инженеров[9]. Известно, что когда люди привыкли полагаться на подсказки (будь то автозаполнение кода или ответы нейросетей), они менее склонны изучать внутренние механизмы и лучшие практики.

[9] BrainHub Editorial Team. Is There a Future for Software Engineers? The Impact of AI. 2025.

Кроме того, финансирование смещается: капиталы идут в продукты поверхностного ускорения, а не в сложные исследования. Это может замедлить прогресс в ИИ. Например, одобрение проектов «скрытых агентов» (agentic AI) и «чёрных ящиков» обещает большую краткосрочную выхлоп, но падение инвестиций в новые архитектуры и методы приведёт к тому, что мы будем бороться с ограничениями нынешних LLM, не создавая принципиально новых решений. Ученые Oak Ridge National Laboratory прогнозируют, что к 2040 г. машины будут писать большую часть кода, но чтобы обеспечить это, потребуются прорывы в управлении гетерогенностью вычислений. Пока внимание сфокусировано на воровстве производительности, мы рискуем не решить «продуктивных» проблем ИИ уровня безопасности, объяснимости и адаптивности.

Риски для Data Science и аналитики данных.

Избыточное доверие генеративному ИИ опасно не только для кода, но и для аналитики. Data Scientist при работе с большими данными опирается на тщательное тестирование гипотез и понимание статистики. Использование AI для построения моделей или объяснения данных может привести к «галлюцинациям» – правдоподобным, но ложным выводам. Например, при автоматической генерации отчетов или графиков LLM может неосознанно воспроизвести предвзятости обучающих данных[10]. PwC подчёркивает широкий спектр рисков генеративного AI: нарушения приватности, предвзятость, регуляторные и правовые проблемы, а также сложности с объяснимостью выводов[10].

На практике это означает: аналитики, полагающиеся на ответы ChatGPT или GPT-4, могут невольно включить в выводы недостоверную информацию или неверно интерпретировать закономерности. Недостаток контекстного понимания ИИ – даже в научных данных – способен привести к ошибочным рекомендациям. Эксперт-маркетолог C. Penn подчёркивает, что «нужно

[10] PwC. Managing the risks of generative AI. PwC Tech Effect, 2023.

проверять информацию, выданную ИИ, прежде чем полагаться на неё»[6]. Это правило тем более важно для данных: если ИИ неверно оценит тренды или неправильно нормализует данные, последствия могут быть драматичны – от неверных бизнес-решений до ложных научных выводов. Таким образом, риски «неосознанного» AI-программирования распространяются и на область аналитики: без надёжной верификации каждая AI-генерированная метрика и предсказание может быть ошибкой или манипуляцией.

Выводы

В результате «вайб-кодинг» не должен означать отключение разума. Напротив, как подчёркивают эксперты, ИИ – лишь инструмент, а не замена инженерной мысли[2]. Главный урок – это инженерная дисциплина: АИ следует использовать как ассистента, а не как архитектора[2]. Необходимо всестороннее тестирование кода, регулярный код-ревью, анализ безопасности и поддерживаемость. Опытные разработчики должны обучать новоиспечённых программистов пониманию основ алгоритмов, а не лишь формулировке промптов. Программист будущего, возможно, будет писать меньше строк, но должен осмысленно задавать требования и критически оценивать результаты.

Нельзя забывать и об ответственности: данные, модели и код – всё это создаётся нами, людьми. Даже самый продвинутый AI — это «настоящее с багами», требующее контроля[11]. Вспоминая цитату из Habr: «Не вайб против инженерии, а вайб через инженерию»[2]. Лишь гармоничное сочетание инновационных AI-инструментов с традиционными инженерными практиками позволит извлечь максимум пользы и избежать разрушительных последствий. В эпоху новых технологий принцип «не отключай голову» остаётся неизменным — управляйте вайбом, пока вайб не начал управлять вами[2].

Литература

[11]Mushchuk S. Разделы XI–XII: Метрики и заключение. Habr.com, 2025.

1. Pshinnik K. Программирование по вайбу: как AI-ассистенты меняют индустрию разработки. Forbes, 29.04.2025.
2. Osaka (Mushchuk S.) Вайб-кодинг с ИИ: разработка без кода или шаг в бездну? Habr.com, 18.04.2025.
3. IBM. What is Vibe Coding? IBM Blog, 2023.
4. GitClear. AI Copilot Code Quality: 2025 Research Report. 2025.
5. Shmueli A. AI-Generated Code: The Security Blind Spot Your Team Can’t Ignore. JIT, 02.04.2025.
6. Penn C. Mind Readings: Blind Trust in AI Overviews. TrustInsights.blog, апрель 2025.
7. Fruhlinger J. Can AI solve your technical debt problem? CIO, 29.04.2025.
8. Sapkota R., Roumeliotis K., Karkee M. Vibe Coding vs. Agentic Coding: Fundamentals and Practical Implications of Agentic AI. Cornell U., arXiv:2505.19443, 2025.
9. BrainHub Editorial Team. Is There a Future for Software Engineers? The Impact of AI. 2025.
10. PwC. Managing the risks of generative AI. PwC Tech Effect, 2023.
11. Mushchuk S. Разделы XI–XII: Метрики и заключение. Habr.com, 2025.