ВЛИЯНИЕ ИНСТРУМЕНТОВ ГЕНЕРАТИВНОГО ИИ НА ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Клемятенко П.В., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Гламаздин И.И. – старший преподаватель

Аннотация. В данной работе рассматривается влияние инструментов генеративного искусственного интеллекта на жизненный цикл разработки программного обеспечения. Анализируются изменения в продуктивности разработчиков, качестве создаваемого кода и распределении ролей между человеком и интеллектуальными системами. Показано, что применение ИИ сопровождается как повышением эффективности при решении типовых задач, так и ростом рисков, связанных с качеством и безопасностью. Особое внимание уделяется зависимости результатов использования ИИ от уровня профессиональной подготовки разработчиков. Сформулированы рекомендации по интеграции ИИ в процессы разработки и обозначены направления трансформации подготовки специалистов.

Современная разработка программного обеспечения переживает фундаментальный сдвиг: большие языковые модели (LLM) превратились из инструментов автодополнения в агентов, способных участвовать в создании архитектуры, генерации тестов и рефакторинге. Эффект от внедрения генеративного ИИ неоднозначен. С одной стороны, фиксируется рост скорости написания кода (до 50% экономии времени на шаблонных задачах) [1], с другой — увеличивается число сбоев при выпуске программных продуктов, возникает риск эрозии критического мышления у разработчиков [2]. Кроме того, при отсутствии тщательной проверки сгенерированный код может снижать общее качество программной системы [3]. В контексте жизненного цикла разработки программного обеспечения влияние генеративного ИИ проявляется на всех этапах — от формализации требований до сопровождения и модернизации систем, что обуславливает необходимость комплексного анализа его воздействия на процессы разработки.

Восприятие продуктивности напрямую зависит от типа решаемых задач. Согласно опросу 65 разработчиков [2], наибольшее снижение трудозатрат (60% респондентов) наблюдается при рефакторинге и проверке кода. ИИ позволяет быстро генерировать повторяющиеся блоки кода и определения переменных, избавляя от рутины. Однако его возможности ограничены при работе со сложными задачами: при необходимости внести изменения в крупный проект ИИ-инструмент может затрагивать код, не связанный с поставленной задачей. Исследование практиков [3] показывает, что спектр подходов к использованию ИИ варьируется от «vibe coding» (свободное взаимодействие) до «agentic coding» (автономные агенты), что влияет на распределение задач между человеком и ИИ. При этом влияние ИИ на продуктивность проявляется неоднородно и зависит от характера задач:

– при решении типовых задач наблюдается значительное сокращение времени разработки;

– при работе с устоявшимися кодовыми базами повышается скорость анализа и навигации;

– при реализации новых архитектурных решений эффект снижается из-за необходимости глубокой проверки;

– при исправлении ошибок возможно как ускорение процесса, так и внесение дополнительных дефектов.

Качество сгенерированного кода варьируется в зависимости от типа задачи. Наибольший объем правок требуется при генерации кода и тестов, что указывает на необходимость существенной доработки. Написание тестов и исправление ошибок требуют корректировок среднего уровня, тогда как рефакторинг и проверка кода — лишь умеренных. Наименьшее количество изменений требуется при генерации документации. Эти данные подтверждают, что обязательной практикой становится оценка качества генерируемого кода [2]. Проблема качества носит системный характер и затрагивает несколько аспектов:

– корректность реализуемой логики и соответствие требованиям;

– устойчивость к ошибкам и граничным условиям;

– безопасность и отсутствие уязвимостей;

– соответствие архитектурным ограничениям проекта;

– читаемость и поддерживаемость кода.

Таким образом, проверка результатов генерации должна рассматриваться как неотъемлемый этап разработки.

Ключевым фактором успешного использования ИИ выступает предметная экспертиза: разработчики, глубоко знающие контекст кодовой базы, лучше способны оценить предложенные решения. При этом наблюдается закономерность: чем выше способность пользователя верифицировать код, тем ниже его удовлетворенность от работы с ИИ-помощниками, тогда как для пользователей с ограниченными навыками выявления ошибок такие инструменты представляют высокую потенциальную ценность, но одновременно вводят риски из-за недостаточной проверки [2].

Восприятие продуктивности обратно коррелирует с опытом разработки. Риск эрозии навыков особенно актуален для начинающих специалистов: традиционный путь накопления опыта нарушается, когда ИИ берет на себя выполнение рутинных функций, что ставит вопрос о трансформации подготовки кадров [3]. Данная зависимость указывает на перераспределение ролей в процессе разработки: ИИ выступает в качестве инструмента усиления, эффективность которого определяется уровнем компетенций пользователя.

Генеративный ИИ не отменяет необходимость экспертизы, но смещает вектор создания ценности. По мере автоматизации рутинного кодирования возрастает значимость компетенций более высокого порядка: точности формулировки требований, архитектурного мышления и управления процессом разработки. Вместо непосредственного написания кода инженер всё чаще выполняет координирующую функцию, формулируя задачи, уточняя требования и проверяя результаты. Естественный язык становится важным средством взаимодействия с системой, а когнитивная нагрузка смещается от синтаксических аспектов к архитектурным решениям, управлению доменной моделью и интеграции компонентов. Узкие места разработки перемещаются в область анализа кода, тестирования, обеспечения безопасности и системного мышления [3].

На основе практического опыта инженеров сформулирован ряд рекомендаций для эффективного использования ИИ в разработке. Важно выбирать инструмент в соответствии с задачей: для написания отдельных функций достаточно встроенной генерации, тогда как для масштабных изменений предпочтительнее использовать инструменты с агентной архитектурой, обеспечивающие автономное планирование, выполнение и итеративную корректировку задач. Генерацию кода целесообразно предварять документированием и разработкой тестов, что способствует повышению качества результатов. Перед началом работы рекомендуется формировать детализированный план решения задачи, разбивая её на управляемые этапы. Существенное значение имеет корректное формирование запросов к ИИ с предоставлением полного контекста, включая архитектурные ограничения и требования к стилю кода. Для сохранения согласованности разработки следует использовать сопроводительную документацию проекта, фиксирующую ключевые решения и зависимости. Наконец, каждая сгенерированная часть кода должна проходить экспертизу с использованием контрольных списков, оценивающих безопасность, корректность и производительность [2]. Представленные рекомендации отражают формирование новых практик разработки, ориентированных на совместную работу человека и интеллектуальных систем.

Выявленные изменения требуют пересмотра подходов к подготовке специалистов. Авторы [3] подчеркивают необходимость смещения фокуса образовательных программ: от изучения синтаксиса — к решению задач и архитектурному мышлению; от индивидуального написания кода — к работе с ИИ-инструментами и оценке качества результатов; от изолированных заданий — к проектно-ориентированному обучению. Кроме того, актуализируется задача формирования навыков ответственного использования ИИ, включая понимание его ограничений и потенциальных рисков.

Использование ИИ в разработке программного обеспечения порождает значимые этические и правовые вопросы, связанные с интеллектуальной собственностью. Ключевыми являются проблемы принадлежности сгенерированного кода и компенсации создателям данных, использованных для обучения моделей [3]. Компании выражают обеспокоенность возможными юридическими последствиями применения такого кода без его модификации. В ответ организации внедряют внутренние политики использования или обучают модели на собственных данных, стремясь минимизировать риски. Дополнительную сложность представляет трансграничный характер разработки, требующий согласования правовых подходов.

Генеративный ИИ выступает не только инструментом повышения продуктивности, но и фактором трансформации процессов разработки и профессиональных ролей. Устойчивость его преимуществ зависит от осознанной интеграции в организационные процессы, прозрачности управления и сохранения ответственности разработчика за конечный результат. В этой связи перспективными направлениями дальнейших исследований являются количественная оценка влияния ИИ на этапы разработки и разработка методик его безопасного применения.

Список использованных источников:

1. Deniz, B. K. Unleashing developer productivity with generative AI [Electronic resource] / B. K. Deniz, C. Gnanasambandam, M. Harrysson, A. Hussin, S. Srivastava // McKinsey & Company. – Mode of access: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai. – Date of access: 25.03.2026.

2. Gurgul, V. The State of Generative AI in Software Development: Insights from Literature and a Developer Survey [Electronic resource] / V. Gurgul, R. Gubela, S. Lessmann // arXiv.org. – Mode of access: https://arxiv.org/abs/2603.16975. – Date of access: 25.03.2026.

3. Chang, H. Coding With AI: From a Reflection on Industrial Practices to Future Computer Science and Software Engineering Education [Electronic resource] / H. Chang et al. // arXiv.org. – Mode of access: https://arxiv.org/abs/2512.23982. – Date of access: 25.03.2026.