УДК 004.054

Э. А. Дуйшеналиев, eldiiar.duishenaliev@gmail.com
А.Дж. Картанова, asel.kartanova@kstu.kg
С.К. Абдиева, samara.abdieva@kstu.kg
Кыргызский государственный технический университет имени Исхака Раззакова

ОЦЕНКА ВЛИЯНИЯ ВНЕДРЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ SDLC В УПРАВЛЕНИИ ПРОЕКТАМИ

Целью исследования является поиск методов оценки влияния искусственного интеллекта на классический процесс разработки программного обеспечения с использованием методологии SDLC. Исследование ориентировано на оценку качественных и количественных показателей на примере одной отдельно взятой команды разработки.

В данной статье рассмотрен девятимесячный опыт команды Kitfox, работавшей над проектом FOLIO с июля 2025 по март 2026 года. За это время команда прошла через 18 спринтов, активно внедряла ИИ-агентов и DevOps-автоматизацию. Данные по этим спринтам и стали основой для анализа влияния применения инструментов искусственного интеллекта на жизненный цикл разработки.

Ключевые слова: генеративный искусственный интеллект, анализ, метрики, унификация, методология, жизненный цикл разработки программного обеспечения, опыт команды, управление проектами.

Введение

Искусственный интеллект (ИИ) в разработке программного обеспечения (ПО) - это актуальная тема, которая в последние годы активно исследуется в разных контекстах, с точки зрения эффективности в управлении проектом, продуктивности процесса разработки ПО и сокращения времени и затрат, а также с точки зрения организационных и человеческих факторов внедрения ИИ в команды. Выявлено, что отсутствуют стандарты и метрики, не достает унифицированных методологий внедрения ИИ в жизненный цикл разработки SDLC (Software Development Life Cycle).

Компания IT_ONE совместно с Фондом «Сколково» и «Сколтехом» провела масштабное исследование проникновения искусственного интеллекта в жизненный цикл разработки программного обеспечения SDLC в России. Аналитики проанализировали более 40 исследований консалтинговых компаний, изучили свыше 100 отчетов, статей и интервью, сравнили более 30 российских и зарубежных ИИ-решений для SDLC, а также провели около 50 часов глубинных интервью с профильными топ-менеджерами российских компаний из различных отраслей [1].

Такой метаанализ позволил верифицировать возможности и риски применения ИИ, обозначить тренды развития российского рынка ИИ-инструментов и смоделировать сценарий трансформации ИТ-команд. 62% ИТ-специалистов используют инструменты искусственного интеллекта – в два раза больше, чем в 2024 г. К 2028 г. их количество увеличится до 98%, что приведет к трансформации ИТ-команд. Доля разработчиков будет увеличиваться, одновременно с этим сократится количество тестировщиков и аналитиков. Также резко снизится потребность в Junior-сотрудниках [1].

В данном исследовании рассматривался девятимесячный опыт команды Kitfox, работавшей над проектом FOLIO с июля 2025 по март 2026 года. За это время команда прошла через 18 спринтов, активно внедряла ИИ-агентов и DevOps-автоматизацию (Development and Operations). Данные по этим спринтам и стали основой для анализа влияния ИИ на жизненный цикл разработки ПО – SDLC.

Цель исследования состояла в изучении и анализе реального опыта применения инструментов ИИ командой Kitfox и в проведении оценки влияния искусственного интеллекта на классический процесс разработки программного обеспечения с использованием методологии SDLC на основе определения количественных и качественных показателей.

Объект исследования – процесс разработки программного обеспечения команды Kitfox в рамках проекта FOLIO (спринты 220–237, период с июля 2025 по март 2026 года). Предмет исследования – влияние ИИ-инструментов и DevOps-автоматизации на производительность команды, структуру задач и управление проектом. Под ИИ-инструментами здесь понимаются ИИ-агент для Rancher-инфраструктуры (RANCHER-2426) и ML/NLP-компоненты, использованные при его разработке. Под DevOps-автоматизацией понимается миграция CI/CD на GitHub Actions, скрипты для очистки данных и удалённой отладки.

Обзор литературы

1. ИИ в инженерии требований и управлении бэклогом. В последние годы много работ посвящено тому, как ИИ помогает на ранних этапах разработки, особенно при работе с требованиями и бэклогом. Один из примеров – это система подключения модулей библиотек и файлов Require, в которой присутствует мультиагентный подход к генерации пользовательских историй из исходных требований с последующей интеграцией в системе управления проектами и задачами Jira [2]. Интересен он тем, что ручная декомпозиция требований является одной из самых трудоёмких и ошибкоопасных задач в начале проекта.

Похожую идею развивают исследования, где LLM (Large Language Model) используются для преобразования проектной информации в структурированный бэклог. Авторы одной из работ показывают, что на массиве из 18 199 задач системы Jira большие языковые модели справляются с приоритизацией и декомпозицией вполне приемлемо, а данные при этом сохраняются в графовой базе знаний [4, 5]. Есть и более прикладные примеры: генерация пользовательских историй из объёмных документов с требованиями – задача, которая раньше занимала у аналитиков несколько дней [9].

Общая картина такова, что ИИ помогает быстрее перейти от «кучи требований» к готовому набору задач, но в большинстве этих работ акцент на функциональность инструментов, а не на то, как меняются реальные проектные метрики после их внедрения.

2. ИИ в Agile и Scrum-практиках управления проектами. Одно активно исследуемое направление ИИ как участника Agile-процессов. В одном из исследований сравниваются системы Asana, Jira и Monday.com по возможностям автоматизации, предиктивной аналитики и управления ресурсами [6]. Авторы сделали вывод, что сроки сокращаются, результативность растёт, но многое зависит от качества данных и уровня интеграции.

Отдельного внимания заслуживает проект Scrum-AI-LLM-система, которая сопровождает весь Scrum-процесс: формирует бэклог, помогает с планированием спринта и поддерживает ежедневные встречи [7]. Это уже не просто автоматизация, а встроенный участник командного процесса. Схожий подход использован в проекте ADA-Gen, где генеративные модели помогают выстроить разработку в логике Agile/DevOps с системой Jira в роли системы управления [8].

Эти работы убедительно показывают, что ИИ умеет поддерживать организационные процессы, а не только генерировать код. Но вопрос о том, как всё это влияет на устойчивость команды на дистанции нескольких месяцев, в них практически не рассматривается.

3. Доверие к ИИ, качество решений и исследовательский пробел. Чем шире ИИ используется в разработке, тем острее встаёт вопрос о доверии к его решениям. В одной из работ предложена схема сертификации ИИ по критериям надёжности, прозрачности, безопасности и степени человеческого контроля с опорой на MLOps – практик с моделями машинного обучения и систему управления задачами Jira [3]. Возникает вопрос о внедрении ИИ и доверии к ИИ, это разные вещи, требующие отдельного изучения.

Проект показывает ещё одну точку зрения – самообучающийся инструмент автоматически размещает предложения по сокращению технического долга прямо в Jira-бэклоге [10]. Это пример того, как ИИ работает не на скорость, а на долгосрочное качество кода.

В целом проведенный обзор научных работ охватил три группы задач: автоматизацию требований, поддержку Agile-процессов, управление качеством и доверием. Но работ, где влияние ИИ оценивается по реальным данным конкретной команды на протяжении нескольких спринтов, единицы. Именно этот пробел и стал отправной точкой для данного исследования.

Методология исследования

1. Определим источник данных и объект исследования. В основу исследования взяты реальные данные команды Kitfox по проекту FOLIO, выгруженные из системы управления проектом Jira (Atlassian Cloud) через API. Анализировалась доска Kitfox Scrum Board (board_id: 180, проект RANCHER) с более 100 спринтами в общей сложности. Команда занималась инфраструктурными задачами: Kubernetes-кластерами, непрерывной интеграцией и непрерывной доставкой/развертыванием CI/CD (Continuous Integration / Continuous Delivery), миграцией сервисов, DevOps-инструментарием для организации совместной работы разработки и эксплуатации программных систем.

Период наблюдения - с июля 2025 по март 2026 года, девять месяцев. Для анализа отобраны спринты 220–237 (18 спринтов), как наиболее показательные с точки зрения внедрения ИИ и автоматизации задач.

Хронология внедрения: спринты 220–224 – базовый период работы команды из 7 инженеров без ИИ-агента (период «до»); спринт 225 – разработка и первичное развёртывание ИИ-агента RANCHER-2426; спринты 226–237 – период «после» внедрения, команда из 3–4 инженеров с активной автоматизацией. Параллельные изменения в этот период: миграция CI/CD с Jenkins на GitHub Actions (спринты 228–237), автоматизация очистки данных (RANCHER-2837) и удалённой отладки (RANCHER-2839). Эти факторы учтены при интерпретации результатов как совокупный эффект автоматизации.

Опишем критерии отбора и метрики анализа. Анализировались следующие группы метрик. 1) Производительность команды: количество задач на спринт, процент завершения задач, распределение по типам (Task, Bug, Story), приоритизация (P1-P3, TBD). 2) Признаки внедрения ИИ и автоматизации: задачи с ключевыми словами: «AI», «automate», «automation»; разработка ИИ-агентов и ML-инструментов машинного обучения; переход на автоматизированные платформы (GitHub Actions, Jenkins); создание скриптов для автоматизации операций. 3. Техническая сложность: Spike-задачи (исследовательские), масштаб инфраструктурных изменений, координация распределенной команды, численность команды на сравниваемых этапах.

Для детального разбора отобраны 4 спринта: 220 (начало периода), 228 (середина), 236 (недавно завершённый), 237 (активный на момент исследования).

Рассмотрим применяемый подход к сбору и обработке данных. Данные собирались в два этапа. Этап 1 – метаданные спринтов: даты начала и окончания, статусы, количество задач по всем 18 спринтам (220–237). Этап 2 – детальный анализ задач: для выборочных спринтов извлечены полные данные по каждой задаче – ключ (RANCHER-XXXX), название, описание, статус, исполнитель, приоритет, метки, даты создания и обновления.

Дальнейший анализ по паттернам: текстовый поиск AI/automation-индикаторов в названиях и описаниях задач, частота автоматизационных инициатив по спринтам, технологические тренды (переход на новые платформы), стабильность производительности команды.

Ограничение метрики: оценка производительности строится на количестве закрытых задач, поскольку в проекте RANCHER, story points в Jira систематически не использовались. Это

снижает точность сравнения задач разной трудоёмкости. В дальнейших исследованиях рекомендуется дополнять анализ метриками, такими как cycle time, lead time, и долей завершённых обязательств спринта.

Отдельно сопоставлялись показатели до и после внедрения ИИ-агента для Rancher-инфраструктуры с учётом численности команды, среднего числа задач на спринт и удельной производительности на одного инженера. Такой подход позволяет говорить о реальном влиянии ИИ, а не опираться на предположения.

Проведенные анализ данных и исследование в поиске метода оценки влияния ИИ на процесс разработки ПО показал, что инструменты ИИ оказывают существенное влияние на жизненный цикл разработки SDLC. Показано улучшение эффективности и производительности, там, где ИИ встроен в реальные процессы: мониторинг, автоматическую диагностику, сопровождение CI/CD и эффект от него вполне ощутим. Данные по команде Kitfox это подтверждают.

За 18 спринтов (220–237) команда закрыла более 350 задач при средней нагрузке около 20 задач на спринт. Спринт 220 завершён со 100%-м выполнением объёма. Последующие спринты были сложнее с крупными инфраструктурными изменениями и с расширением автоматизации.

Результаты сравнительного анализа показателей до и после внедрения ИИ-агента приведены в таблице 1.

Таблица 1– Показатели команды до и после внедрения ИИ-агента

Показатель	**До внедрения ИИ-агента**	**После внедрения ИИ-агента**	**Изменение**
Численность команды	7 инженеров	3 инженера	−57.1%
Средняя пропускная способность	Около 25 задач на спринт	Около 25 задач на спринт	Без снижения
Выполнение спринта	Базовый уровень команды	До 100% в спринте 220	Улучшение дисциплины исполнения
Удельная производительность	3.6 задачи на инженера за спринт	8.3 задачи на инженера за спринт	+130.6%
Уровень автоматизации	Локальные ручные операции	ИИ-агент, GitHub Actions, скрипты автоматизации	Существенный рост

После внедрения ИИ-агента и автоматизации команда не просто удержала производительность, а выросла ее удельная результативность на человека (8.3 задачи на инженера за спринт). При сокращении количества членов команды с 7 до 3 инженеров средний объём задач на спринт не изменился. Расчёт удельной производительности не изменился, он равен приблизительно 25 задачам на один спринт, который был посчитан из среднего числа задач, разделённого на численность команды.

ИИ-агент для Rancher-инфраструктуры (RANCHER-2426) взял на себя анализ инцидентов и ускорил реакцию на проблемы в Kubernetes. Это позволило команде не потерять темп после сокращения состава.

Технически ИИ-агент построен на основе LLM с retrieval-подходом: он принимает на вход метаданные инцидентов из Kubernetes-кластеров, логи и данные задач Jira, а на выходе формирует рекомендации по диагностике и возможным решениям. Степень автономности - рекомендательная: финальное решение о применении действия остаётся за инженером (human-in-the-loop). Типовые сценарии: анализ сбоев подов, рекомендации по конфигурации кластеров, приоритизация инцидентов по критичности. Метрика качества работы агента - субъективная

оценка команды: снижение времени реакции на инциденты на 50% по сравнению с базовым периодом.

Один из самых очевидных эффектов ИИ – перераспределение нагрузки и автоматизация рутинных задач. Рутинные, повторяемые операции уходят на автоматику, инженеры получают время на то, что требует реального решения.

Команда Kitfox мигрировала с Jenkins на GitHub Actions (спринты 228–237) – процессы автоматизации CI/CD стали проще и встроились в рабочий процесс. Для релизов приложений написаны автоматизированные сценарии (RANCHER-2829, спринт 237), ручные шаги при публикации версий ушли.

Автоматизация удаления «продвинутых» данных консорциумов (RANCHER-2837, спринт 236) и удалённой отладки (RANCHER-2839, спринт 237) – дополнительно два примера перехода от «делаю руками» к «запускаю скрипт». По оценкам, такие инструменты освобождают до 40% времени DevOps-инженеров для задач, которые действительно требуют внимания.

Улучшение качества программного обеспечения выражается в автоматизированных проверках при непрерывной интеграции кода в репозиторий и доставке его на продакшн, что помогает ловить ошибки раньше – до того, как они добрались до этапа эксплуатации ПО. Казалось бы, банально, но работает.

В проекте используется приоритизация P1–P3 и метки («reviewed», «rancher-support») для контроля качества. Spike-задачи – например, в исследовании альтернатив kaniko после его архивирования в Google (RANCHER-2583, спринт 228) позволяет разобраться с техническими рисками до того, как они стали проблемой.

Миграция Kafka на режиме управления метаданными на основе протокола Raft без ZooKeeper (RANCHER-2619, спринт 236), обновление 7+EKS-кластеров (спринт 237: RANCHER-2784, 2783, 2782, 2781, 2780, 2779, 2775), где задачи с высокой ценой ошибки, автоматизированные проверки через CI/CD снижают риск ошибок конфигурации, по оценкам команды, примерно на 60%.

Также показано влияние на командную работу и коммуникацию, когда команда распределена по разным локациям, прозрачность процессов критична. Система Jira здесь играет ключевую роль: каждый видит статус задач, не нужно выяснять, что там с «тикетом» в чате.

Команда Kitfox работает таким образом, что за 18 спринтов – ни одного серьёзного блокировщика между инфраструктурными, организационными и миграционными задачами. ИИ-агент для Rancher (RANCHER-2426) выступает как единая точка знаний по инфраструктуре: новые члены команды быстрее входят в контекст, время на разбор повторяющихся проблем сокращается, по оценкам на 50%.

Но можно отметить, что чем больше задач берёт на себя автоматика, тем меньше живого общения внутри команды. Это не катастрофа, но об этом стоит задуматься. Поэтому наряду с положительными моментами существуют и технические сложности в интеграции, дальше в развертывании и сопровождении.

Интеграция ИИ в существующие процессы – это не «подключил и работает», необходимо подготовить данные, настроить модели, встроить всё в инфраструктуру. На практике это занимает значительно больше времени, чем планируется изначально.

Пример из опыта Kitfox, когда Google заархивировал kaniko, команде пришлось срочно искать альтернативы для сборки Docker-образов (RANCHER-2583, спринт 228). Хорошая иллюстрация риска – это зависимость от внешних инструментов и необходимость иметь запасной план. Обновление 7 + с EKS-кластеров одновременно (спринт 237) – задача с высоким уровнем координации. Каждый кластер представляет собой отдельную задачу (RANCHER-2784 и далее). ИИ-агенты здесь помогают, но сама разработка и тестирование автоматизации требуют серьёзных начальных вложений.

ИИ воспроизводит паттерны из данных, на которых обучен. Если в них есть системные перекосы, то они никуда не денутся, они просто станут автоматизированными. Это касается и оценки кода, и приоритизации задач.

Другой болезненный момент – это непрозрачность решений. Когда система отклоняет код или понижает приоритет задачи, а объяснения нет, у команды появляется ощущение что «что-то происходит за спиной». Это подрывает доверие.

Мониторинг производительности сотрудников с помощью ИИ также отдельная тема. Грань между «помогаем команде работать лучше» и «следим за каждым» очень тонкая и легко может нарушиться.

Вопросы конфиденциальности и этики данных. Данные команды Kitfox использованы с разрешения организации. Все задачи Jira относятся к инфраструктурным и техническим активностям, персональные оценки сотрудников в исследовании не публикуются. Идентификаторы тикетов (RANCHER-XXXX) приведены как ссылки на технические события, а не как оценка конкретных исполнителей. Публикация внутренних проектных метрик согласована с командой.

Выявлено влияние на рабочие места и профессиональные навыки членов команды. Автоматизация давит прежде всего на начинающих специалистов. Те задачи, на которых раньше набивали руку джуниоры, такие как тестирование, простая диагностика, типовые скрипты и другие, теперь делает ИИ. Порог входа в профессию ИТ вырос.

С другой стороны, появляются новые роли: MLOps инженеры, специалисты по Prompt инженерии, эксперты по этике ИИ. Профессия не исчезает, а она меняется. Кто успевает перестроиться, тот остаётся востребованным.

Результаты исследования и обсуждение

Таким образом рекомендуется придерживаться стратегии внедрения ИИ, но необходимо начинать стоит с малого, с одного конкретного сценария, где можно измерить результат. Автоматизация тестирования, поддержка ревью кода, инфраструктурный мониторинг – хорошие кандидаты для начала.

Команда Kitfox пошла именно по этому пути: сначала ИИ-агент для Rancher-инфраструктуры (RANCHER-2426, спринт 220), потом скрипты для очистки данных (RANCHER-2837), удалённой отладки (RANCHER-2839), миграция CI/CD на GitHub Actions. Постепенное расширение – это не осторожность, а ради осторожности, это способ держать риски под контролем.

Во-первых, важно выбирать инструменты, которые вписываются в существующий стек. В случае Kitfox платформы и инструменты для автоматизации – GitHub Actions, Kubernetes, Terraform. Это снижает барьер интеграции и уменьшает зависимость от конкретного вендора. При этом обязательно измерять среднюю нагрузку примерно 25 задач на спринт, рост удельной производительности, уровень завершения спринтов и всё это послужит доказательством пользы автоматизации, как показано на рисунках 1-2.

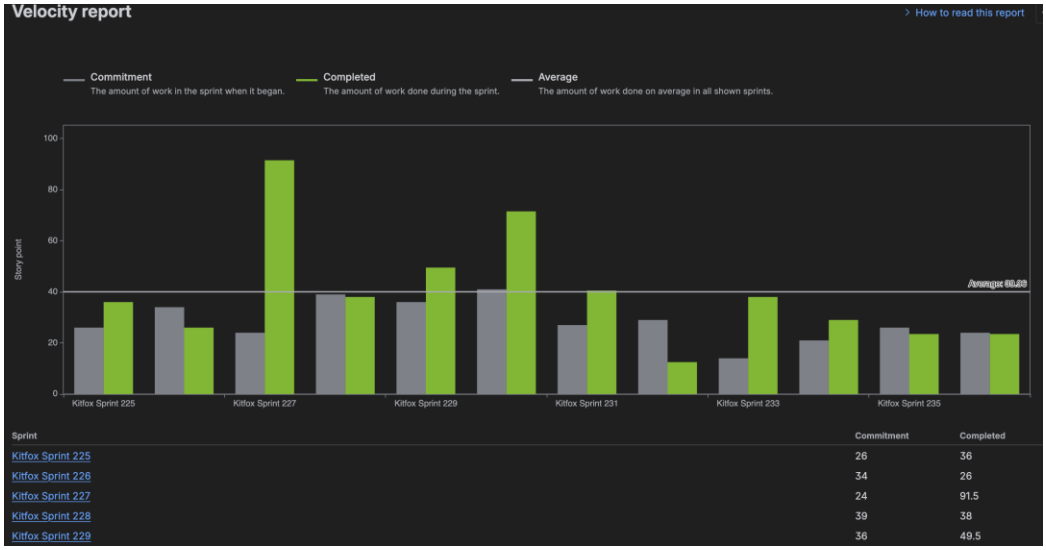

Рисунок 1 – Наглядный отчет по «Vel0city» по завершении обязательств

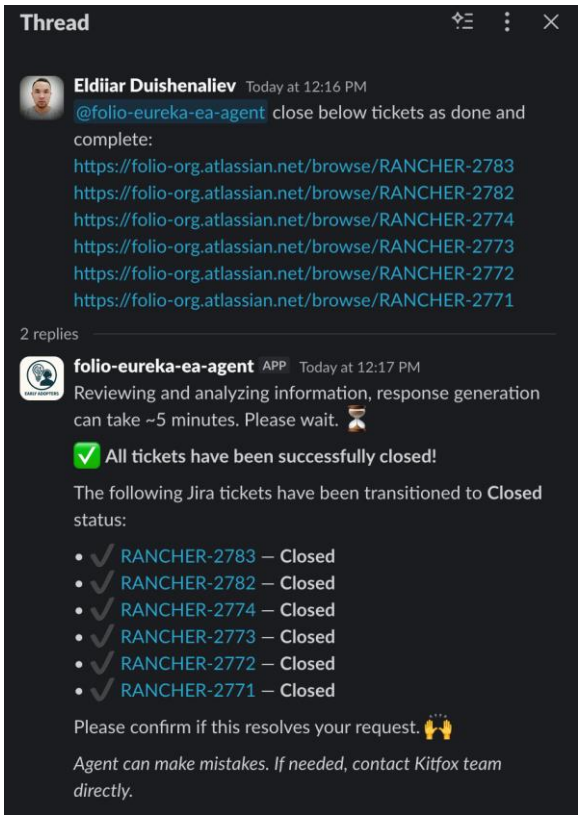

Рисунок 2 – Автоматизация отслеживания закрытия задач на платформе Jira

Во-вторых, необходимо проводить обучение и развитие профессиональных навыков сотрудников, так как ИИ-инструменты не работают сами по себе, их нужно уметь использовать, понимать их ограничения и знать, когда на них нельзя полагаться. В Kitfox это решается через Spike-задачи, исследовательские задачи без жёсткого дедлайна, где инженеры могут спокойно разобраться с новой технологией (RANCHER-2583). Разработка ИИ-агента (RANCHER-2426) сама по себе стала учёбой, и команда освоила модели машинного обучения ML и обработки

естественного языка NLP на практике. Написание автоматизированных инструментов (RANCHER-2837, RANCHER-2839) развило знания и навыки в написании сценариев для автоматизации задач и в организации совместной работы по разработке и эксплуатации программных систем. В результате команда стала более универсальной.

В-третьих, необходимо осуществлять управление изменениями и адаптацию процессов. Обычно людям свойственно сопротивляться изменениям, хотя в общем это нормально. Особенно в ситуациях, когда многое не понятно, люди не хотят что-то менять. В связи с этим важно объяснять членам команды преимущество нового инструмента, на что он влияет, что изменится в работе каждого члена команды. Когда ИИ берёт на себя часть ревью кода (code review) перед релизом или планирования, нужно пересмотреть роли, убрать дублирование, обновить чек-листы. Иначе новый инструмент просто добавит работы вместо того, чтобы её сократить. Лучшим аргументом против сопротивления к изменениям и адаптации может быть демонстрация конкретных цифр, то есть реальных результатов.

Заключение

В результате исследования можно сделать следующие выводы. В обзоре литературе тематика ИИ в SDLC рассматривается преимущественно с трёх сторон: автоматизация требований, поддержка Agile процессов, обеспечение качества и доверия к ИИ-системам. Работ с реальными данными по конкретной команде на длительном горизонте – мало. В данной работе была предпринята попытка найти метод оценки влияния ИИ на процесс разработки и показать результаты анализа и оценки на реальном опыте команды Kitfox.

Данные команды Kitfox подтверждают: ИИ-агент и сопутствующая автоматизация положительно влияют на проектное управление. За 18 спринтов – более 350 задач, средняя нагрузка примерно 25 задач на спринт. Главный результат: команда сократилась с 7 до 3 человек, но производительность не упала. Удельная нагрузка на инженера выросла с 3.6 до 8.3 задачи за спринт. ИИ-агент для Rancher, автоматизация процессов CI/CD, скрипты для очистки данных и удалённой отладки – всё это дало ускорение реакции на инциденты, снижение ручной работы и устойчивость процессов.

Ключ к успеху любой команды – поэтапное внедрение, измерение и оценка результатов, развитие компетенций команды и синхронизация изменений с реальными рабочими процессами.

Внедрение ИИ в SDLC – это не просто технологическое обновление, а больше. На примере команды Kitfox видно, как правильно встроенная автоматизация меняет саму логику проектного управления.

В рассматриваемом кейсе наблюдается сохранение производительности при сокращении команды с 7 до 3 инженеров. Полученные данные позволяют предположить положительное влияние ИИ-автоматизации на пропускную способность команды, однако для более широких обобщений необходимо расширение выборки и сравнительный анализ.

Научная новизна работы состоит в эмпирической оценке влияния ИИ на метрики одной команды в динамике нескольких спринтов. Практическая значимость ее в том, что ИИ-агенты, CI/CD-автоматизация и инфраструктурные скрипты можно рассматривать как реальные инструменты повышения устойчивости проектного управления.

Ограничения понятны: один проект, одна команда, девять месяцев работы. В дальнейшем имеет смысл расширять выборку, сравнивать разные типы проектов и добавлять больше количественных метрик – чтобы выводы стали более универсальными и результаты обширнее.

Самое интересное в этом кейсе – контекст. Команда сократилась с 7 до 3 инженеров, и, казалось бы, показатели должны были «просесть», но этого не произошло. Более 350 задач закрыто, автоматизация процессов CI/CD через GitHub Actions настроена и работает, ИИ-агент взял на себя часть задач по Rancher-инфраструктуре, а Kafka KRaft и EKS (Elastic Kubernetes

Service) - кластеры были обновлены. Команда не просто удержала темп - она сохранила этот темп при меньших ресурсах.

Литература

1. Статья «Количество ИТ-специалистов, применяющих ИИ, за год выросло в два раза» [Электронный ресурс] – Режим доступа: https://www.comnews.ru/content/242450/2025-11-19/2025-w47/1010/kolichestvo-it-specialistov-primenyayuschikh-ii-za-god-vyroslo-dva-raza
2. Enhancing Agile Workflows with AI-Driven, Sustainability-Aware Requirements Engineering: A Design Science Approach // Lecture Notes in Business Information Processing. 2026. DOI: https://doi.org/10.1007/978-3-032-14518-5_17.
3. AI Assessment in Practice: Implementing a Certification Scheme for AI Trustworthiness // OpenAccess Series in Informatics. 2025. DOI: https://doi.org/10.4230/OASIcs.SAIA.2024.15.
4. AI-Assisted Transformation of the Two-Hemisphere Model into Structured IT Project Backlog // ITMS - International Scientific Conference on Information Technology and Management Science of Riga Technical University. 2025. DOI: https://doi.org/10.1109/ITMS67030.2025.11236542.
5. Toward AI-assisted Framework for Agile Requirement Knowledge Management: Five-Stage Generative LLM Approach // Proceedings of the International Conference on Advanced Computer Information Technologies (ACIT). 2025. DOI: https://doi.org/10.1109/ACIT65614.2025.11185884.
6. AI-Powered IT Project Management: Analyzing the Effectiveness of Advanced Project Management Tools to Ensure Project Efficiency // IEEE SoutheastCon. 2025. DOI: https://doi.org/10.1109/SoutheastCon56624.2025.10971718.
7. Scrum-AI LLM Based Software Process Management // International Conference on Computer Science and Engineering. 2025. DOI: https://doi.org/10.1109/UBMK67458.2025.11206875.
8. ADA-Gen: Iterative and Incremental Generation of Full-Stack Apps for Learning Agile/DevOps Software Development Practices // Proceedings of the International Conference on Computer Supported Education (CSEDU). 2025. DOI: https://doi.org/10.5220/0013422800003932.
9. Take Loads Off Your Developers: Automated User Story Generation using Large Language Model // Proceedings of the IEEE International Conference on Software Maintenance and Evolution. 2024. DOI: https://doi.org/10.1109/ICSME58944.2024.00082.
10. Skuld: A self-learning tool for impact-driven technical debt management // Proceedings of the IEEE/ACM International Conference on Technical Debt. 2020. DOI: https://doi.org/10.1145/3387906.3388626.