DORA

Модель возможностей ИИ



2025

Google Cloud

Содержание

02
Краткое резюме

06
ИИ как усилитель

12
Семь возможностей для эффективного применения ИИ

14
Чёткая и доведённая до сотрудников позиция по ИИ

20
Зрелая экосистема данных

26
Внутренние данные, доступные для ИИ

36
Надёжные практики контроля версий

44
Работа небольшими порциями

52
Ориентация на потребности пользователей

58
Качественные внутренние платформы

66
Оценка и приоритизация возможностей

67
Определение профиля команды

72
Углублённый анализ с помощью карты потока создания ценности

81
Командный семинар по расстановке приоритетов

88
Заключение. Дорожная карта внедрения ИИ

94
Благодарности

Если не указано иное, источники проверены в октябре 2025 года. Отчёт «Модель возможностей ИИ DORA» (Google LLC) и эта русская адаптация распространяются по лицензии CC BY-NC-SA 4.0.

Краткое резюме





Главный вывод: ИИ усиливает существующие возможности

В отчёте DORA о разработке ПО с использованием ИИ[1] показано, кто применяет ИИ и как именно. ИИ используют почти 90 % опрошенных специалистов.

Мы проанализировали более 100 часов качественных данных и ответы почти 5 000 специалистов в сфере технологий. Основная роль ИИ в разработке ПО — усиливать уже существующие особенности работы. В результативных организациях он укрепляет сильные стороны, а там, где процессы дают сбои, усугубляет проблемы.

Наибольшую отдачу приносят вложения в процессы и системы, от которых зависит успешное применение ИИ.

Модель возможностей ИИ DORA

В отчёте DORA о разработке ПО с использованием ИИ представлена модель возможностей ИИ DORA. Она описывает семь базовых возможностей, которые усиливают положительное влияние ИИ на результативность. Они охватывают как технические практики, так и культуру работы.

Модель дополняет базовую модель DORA[2]. Базовая модель описывает возможности, от которых зависят поставка программного обеспечения и результативность организации. Модель возможностей ИИ выделяет то, что стоит развивать для успешного внедрения ИИ и получения максимальной отдачи.

Семь ключевых возможностей

Чёткая и доведённая до сотрудников позиция по ИИ

Неопределённость создаёт риски. Ясные правила обеспечивают психологическую безопасность, необходимую для экспериментов.

Зрелая экосистема данных

Качественные, доступные и согласованные внутренние данные повышают пользу от ИИ.

Внутренние данные, доступные для ИИ

Доступ к внутренней документации и репозиториям превращает универсального ИИ-помощника в специалиста по вашим системам.

Надёжные практики контроля версий

Чем быстрее вносятся изменения с помощью ИИ, тем важнее контроль версий. Он позволяет безопасно экспериментировать и при необходимости откатывать изменения.

Работа небольшими порциями

Небольшие порции изменений снижают риск того, что ИИ внесёт крупные и нестабильные изменения. Это помогает превратить ускорение разработки в улучшение продукта.

Ориентация на потребности пользователей

Потребности пользователей помогают командам направить скорость, которую даёт ИИ, на решение нужных задач.

Качественные внутренние платформы

Платформа предлагает безопасные автоматизированные способы работы, которые помогают распространить преимущества ИИ на всю организацию.

Как пользоваться отчётом

Этот отчёт — практическое руководство по модели возможностей ИИ DORA. Для каждой из семи возможностей он описывает подходы к внедрению, первые шаги для команд и способы отслеживать прогресс и непрерывно улучшать работу. Руководство адресовано руководителям и специалистам, которые хотят создать условия для эффективного применения ИИ при разработке ПО.

Читайте это руководство вместе с отчётом DORA за 2025 год о разработке ПО с использованием ИИ[3].

У каждой организации свои условия. Результаты исследования помогут выбрать стратегию и направить работу команд. Формулируйте гипотезы, проводите эксперименты и измеряйте результаты, чтобы узнать, какие подходы лучше всего работают в ваших условиях.



1. Отчёт о разработке ПО с использованием ИИ. https://dora.dev/dora-report-2025
2. Исследовательская программа DORA. https://dora.dev/research
3. Отчёт о разработке ПО с использованием ИИ. https://dora.dev/dora-report-2025



ИИ как усилитель

Нейтен Харви

Руководитель DORA, Google Cloud

Для того чтобы помочь им ответить на этот вопрос, DORA провела исследование в 2025 году, которое включало более 100 часов качественных данных и ответов на опросы почти 5000 специалистов по технологиям со всего мира.1

Исследование раскрывает критическую истину: основная роль ИИ в разработке программного обеспечения заключается в усилении. ИИ увеличивает сильные стороны высокоэффективных организаций и дисфункции борющихся.



Успех внедрения ИИ зависит от культуры и практик работы

Мы определили семь основополагающих возможностей, включая четкую политику ИИ, здоровую экосистему данных, качественную внутреннюю платформу и ориентированную на пользователя направленность, которые, как доказано, усиливают положительное влияние ИИ на производительность.

Относитесь к внедрению ИИ как к организационной трансформации. Наибольшая отдача будет получена от инвестиций в основополагающие системы, которые усиливают преимущества ИИ: вашу внутреннюю платформу, вашу экосистему данных и основные инженерные дисциплины ваших команд.

Эти элементы являются необходимыми предпосылками для превращения потенциала ИИ в измеримую организационную эффективность.

Действительно, многие из них являются одними и теми же основными возможностями2, которые уже давно доказали, что позволяют высокопроизводительным, технологически управляемым командам. ИИ является последним значительным изменением в этом ландшафте, принося как захватывающие возможности, так и новые вопросы.

DORA уже давно стремится помочь организациям использовать решения, основанные на данных, чтобы ориентироваться именно в этих видах эволюции и соответствовать моменту. В этом году мы вышли за рамки вопросов о том, кто принимает ИИ (90% наших респондентов используют ИИ в рамках своей работы)3 и как они его используют, чтобы исследовать условия, в которых разработчики программного обеспечения с помощью ИИ наблюдают лучшие результаты.

Мы представляем эти результаты в качестве нашей первой модели возможностей DORA ИИ. Семь возможностей ИИ в этой инаугурационной модели:

Наше исследование охватывает как технические, так и культурные аспекты организации и предполагает, что инвестиции в развитие этих областей могут помочь раскрыть потенциал инструментов ИИ.

Модель возможностей DORA ИИ разработана, чтобы дополнять модель DORA Core Model.4 Она не заменяет ее. В то время как модель Core определяет основополагающие драйверы производительности доставки программного обеспечения, эта новая модель исследует конкретные условия, при которых помощь ИИ обеспечивает лучшие результаты для команд и организаций, управляемых технологиями.

Как и в случае с моделью DORA Core Model 5, мы продолжим проверку, пересмотр и доработку модели возможностей DORA ИИ с дальнейшими исследованиями.

Модель возможностей ИИ DORA

На рисунке 1 показано, какие возможности усиливают влияние внедрения ИИ на конкретные результаты. Это модель возможностей DORA ИИ.

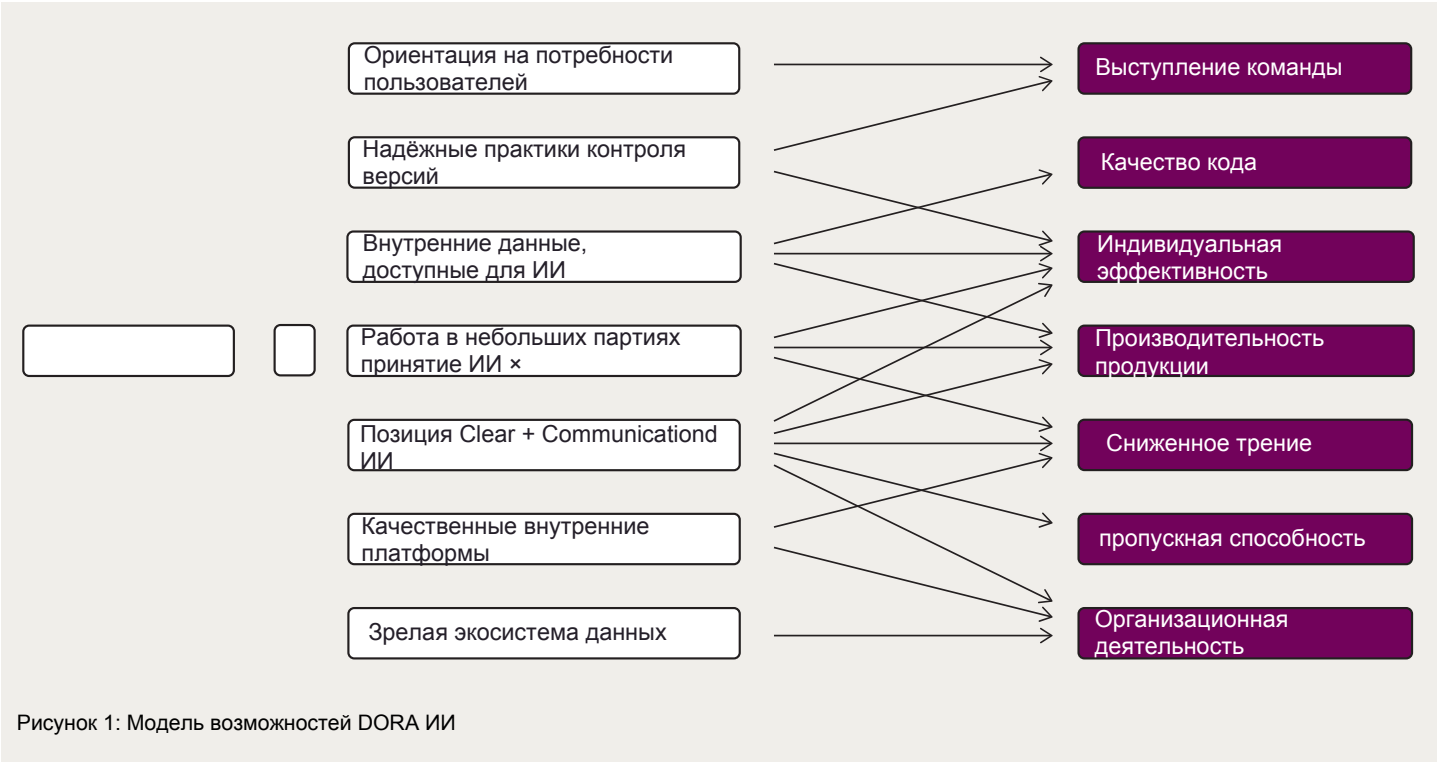


Рисунок 1: Модель возможностей DORA ИИ

Успешное использование ИИ в разработке программного обеспечения не так просто, как внедрение новых инструментов. Скорее, организации должны культивировать конкретную техническую и культурную среду, чтобы пожинать наибольшие выгоды.



Как применение ИИ связано с основными результатами

Наше исследование направлено на определение того, предсказывают ли различные уровни внедрения ИИ различия в ключевых результатах:

Организационная деятельность

Общий успех организации, основанный на таких характеристиках, как прибыльность, доля рынка и удовлетворенность клиентов.

Выступление команды

Воспринимаемая эффективность и сила совместной работы непосредственной команды человека.

Производительность продукции

Успех и качество продуктов или услуг, которые команда создает, основаны на таких характеристиках, как помощь пользователям в выполнении важных задач и обеспечении безопасности информации, а также показателях производительности, таких как задержка.

Пропускная способность программного обеспечения

Скорость и эффективность процесса доставки программного обеспечения.

Нестабильность доставки программного обеспечения

Качество и надежность процесса доставки программного обеспечения.

Качество кода

Индивидуальная оценка качества кода, лежащего в основе основного приложения или услуги, над которой они работают.

Индивидуальная эффективность

Самооценка индивидуальной эффективности и чувство выполненного долга на работе.

Ценная работа

Самооцененное количество времени, которое человек тратит на работу, которую он считает ценной и стоящей.

трение

Степень, в которой трение препятствует работе человека. Более низкие количества трения обычно считаются положительным результатом.

выгорание

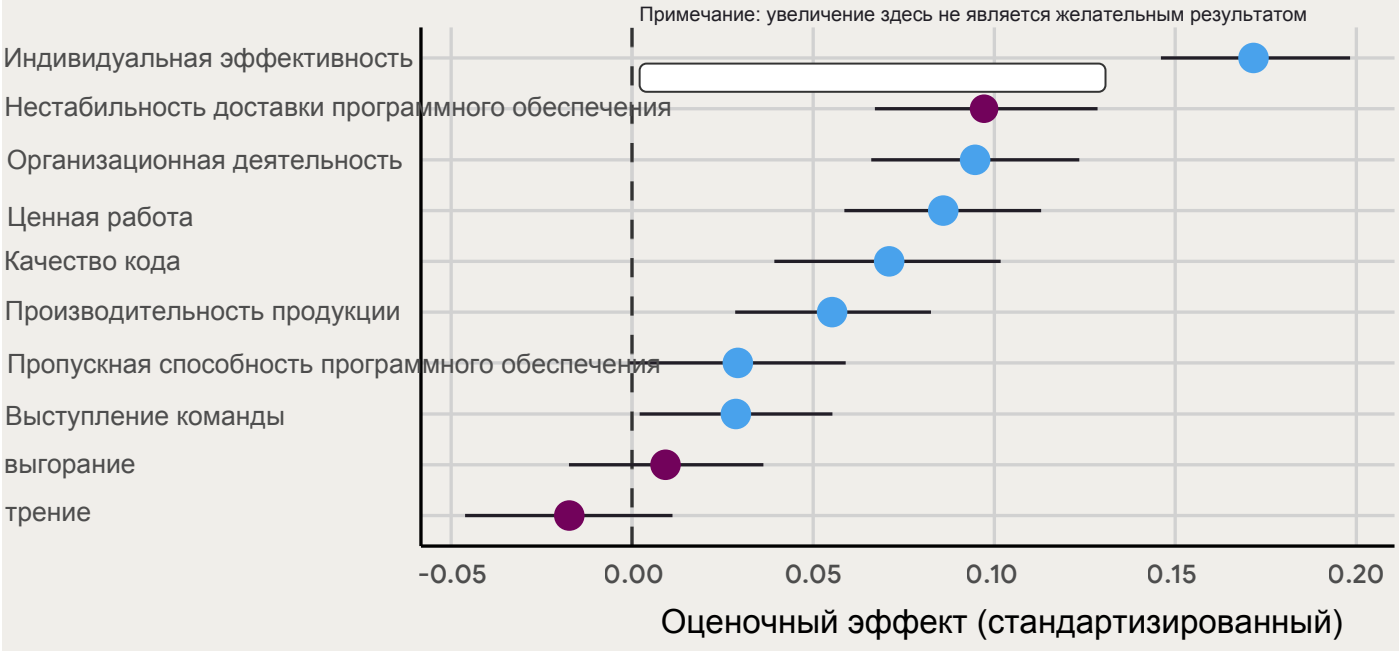
Чувства истощения и цинизма, связанные с работой. Более низкие уровни выгорания, как правило, считаются положительным результатом.

Рисунок 2 визуализирует отношения, которые ИИ имеет с этими результатами.

Подробнее о том, как мы измеряем принятие ИИ и как оно связано с ключевыми результатами в главе «Исследование отношения ИИ к ключевым результатам» отчета о состоянии разработки программного обеспечения с помощью ИИ.6



Оценка влияния внедрения ИИ на ключевые результаты с 89% достоверными интервалами



Для результатов в фиолетовом, таких как выгорание, желательно отрицательное влияние. Рисунок 2: Ландшафт воздействия ИИ

1.Дополнительная информация о том, кто принимал участие в исследовании этого года, содержится в главе «Демография и фирмография» доклада о состоянии разработки программного обеспечения с использованием ИИ. https://dora.dev/dora-report-2025
2. «Исследовательская программа DORA». https://dora.dev/research/?view=detail
3. Состояние разработки программного обеспечения с помощью ИИ. 24. https://dora.dev/dora-report-2025
4. «Исследовательская программа ДОРА». https://dora.dev/research
5 Там же.
6. Состояние разработки программного обеспечения с помощью ИИ. 33-48. https://dora.dev/dora-report-2025

DORA COMMUNITY

Сообщество DORA1 предоставляет профессионалам платформу для участия в этом исследовании и применения его для улучшения их собственных организационных показателей.



Почему стоит присоединиться к сообществу DORA?

Есть несколько причин, по которым вы должны быть частью сообщества:

Учитесь у экспертов и коллег: сообщество предлагает возможность учиться у приглашенных докладчиков и других членов посредством презентаций и дискуссий.

Будьте в курсе исследований: будьте первыми, кто узнает о новой информации и публикациях от DORA.

Сотрудничайте и обсуждайте: Сообщество DORA Google Group2 предоставляет форум для асинхронных разговоров, объявлений и приглашений на мероприятия. Это позволяет участникам обсуждать темы и делиться своим опытом с другими в этой области.

Участие в мероприятиях сообщества: Календарь событий, как виртуальных, так и личных, доступен на сайте DORA.community.

Содействие разговору: Содействие разговору путем прослушивания, разговора и участия в чатах.Сообщество ценит вклад своих членов и предоставляет пространство для текущих дискуссий по таким темам, как лидерство, расширение возможностей команды и эволюция технологических практик.

1. Сообщество Дора. https://dora.community
2. Сообщество DORA Google Group. https://groups.google.com/g/dora-community/about

Семь возможностей для эффективного применения ИИ

Мы хотели понять конкретные возможности и условия, которые позволяют командам достигать положительных результатов.

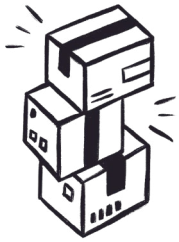
Нейтен Харви

Руководитель DORA, Google Cloud

Эти семь возможностей составляют основу нашей новой модели:



Чёткая и доведённая до сотрудников позиция по ИИ



Работа небольшими порциями



Зрелая экосистема данных



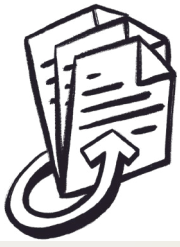
Ориентация на потребности пользователей



Внутренние данные, доступные для ИИ



Качественные внутренние платформы



Надёжные практики контроля версий

В этом отчете приведены практические советы, основанные на семи возможностях DORA ИИ. Вот краткое резюме, которое поможет вам начать работу.

Уточните и социализируйте свою политику ИИ

Неоднозначность вокруг ИИ подавляет принятие и создает риск. Установить и социализировать четкую политику в отношении разрешенных инструментов и использования для создания доверия разработчиков. Эта ясность обеспечивает психологическую безопасность, необходимую для эффективного экспериментирования, снижения трений и усиления положительного влияния ИИ на индивидуальную эффективность и организационную эффективность.

Относитесь к своим данным как к стратегическому активу

Преимущества ИИ в организационной эффективности значительно усиливаются здоровой экосистемой данных. Инвестируйте в качество, доступность и унификацию ваших внутренних источников данных. Когда ваши инструменты ИИ могут учиться на высококачественных внутренних данных, их ценность для вашей организации возрастает.

Подключите ИИ к своему внутреннему контексту

Подключите свои инструменты ИИ к внутренним системам, чтобы выйти за рамки общей помощи и разблокировать повышение индивидуальной эффективности и качества кода. Это означает выход за рамки простого приобретения лицензий и инвестирования инженерных усилий, чтобы дать вашим инструментам ИИ безопасный доступ к внутренней документации, кодовым базам и другим источникам данных. Это обеспечивает специфический для компании контекст, необходимый для того, чтобы инструменты были максимально эффективными.

Обнимите и укрепите свои защитные сетки

Кодирование с помощью ИИ может увеличить объем и скорость изменений, что также может привести к большей нестабильности. Ваша система управления версиями является критической сетью безопасности. Поощряйте команды стать высококвалифицированными в использовании функций отката и возврата, поскольку эта практика связана с лучшей производительностью команды в среде с помощью ИИ.

Уменьшите размер рабочих предметов

В то время как ИИ может повысить восприятие индивидуальной эффективности, генерируя большое количество кода, наши результаты показывают, что это не обязательно самый важный показатель. Вместо этого сосредоточьтесь на результатах. Обеспечьте дисциплину работы небольшими партиями, что улучшает производительность продукта и уменьшает трение для команд, работающих с ИИ.

Потребности пользователей в стратегии продукта

Профессионалы могут испытывать значительное повышение своей личной эффективности, когда они принимают ИИ. Однако, если потребности людей, для которых они строят, не являются их фокусом, они могут быстро двигаться в неправильном направлении. Мы обнаружили, что принятие инструментов разработки с помощью ИИ может нанести вред командам, которые не имеют ориентированного на пользователя фокуса. И наоборот, сохранение потребностей пользователей в качестве продукта North Star может направлять разработчиков с помощью ИИ к соответствующим целям и оказывает исключительно сильное положительное влияние на производительность команд, использующих ИИ.

Инвестируйте в свою внутреннюю платформу

Качественная внутренняя платформа является ключевым фактором для увеличения положительного влияния ИИ на организационную эффективность. Эти платформы обеспечивают необходимые ограждения и общие возможности, которые позволяют преимуществам ИИ эффективно и безопасно масштабироваться в организации.



Чёткая и доведённая до сотрудников позиция по ИИ

Нейтен Харви

Руководитель DORA, Google Cloud

Вивиан Ху

10X Lead, дельта Google Cloud Consulting, Google Cloud

Четкая и коммуникативная позиция усиливает положительное влияние внедрения ИИ на:

- Индивидуальная эффективность
- Организационная деятельность
- Пропускная способность программного обеспечения

Четкая и коммуникативная позиция по внедрению ИИ уменьшает трение.

Зачем нужна ясная позиция по использованию ИИ

Четкая позиция обеспечивает психологическую безопасность, позволяя разработчикам эффективно экспериментировать и использовать инструменты ИИ с уверенностью, зная, что они работают в утвержденных границах.



Ясность также может уменьшить трение и усилить положительное влияние ИИ на индивидуальную эффективность и эффективность организации.

Четкая и коммуникативная позиция ИИ является официальной позицией организации по использованию разработчиками инструментов разработки с помощью ИИ.

Успешная позиция заключается не в том, чтобы быть «разрешительным» или «ограничительным», а в том, чтобы быть:

Понятный: он хорошо определен, легко понятен и обеспечивает практические ограждения.

Общение: широко известен, доступен и социализирован в командах разработчиков.

Эта ясность должна применяться к четырем ключевым областям:

1. Ожидание использования: Как вы думаете, ожидается ли использование ИИ на работе?

2.Поддержка: поддерживает ли организация разработчиков в экспериментах с ИИ?

3. Разрешение: Ясно ли, какие инструменты ИИ разрешены для использования?

4.Применимость: Понимает ли политика организации в области ИИ, что она непосредственно относится к вашей роли?

В ходе углубленных интервью, которые мы провели в этом году, разработчики последовательно выявили отсутствие ясности и осведомленности о позиции своей организации в отношении ИИ.

1. Разработчики действуют слишком консервативно, опасаясь, что они преодолеют границы.
2. Разработчики действуют слишком разрешительно, используя ИИ неодобренными способами.

Ни один из этих случаев не является оптимальным.

Организации, имеющие четкую и четкую позицию в отношении ожиданий и приемлемости ИИ в разработке программного обеспечения, могут способствовать укреплению доверия разработчиков1, смягчать проблемы конфиденциальности данных2 и масштабировать внедрение ИИ.3

Четкая и коммуникативная позиция ИИ смягчает влияние ИИ на индивидуальную эффективность

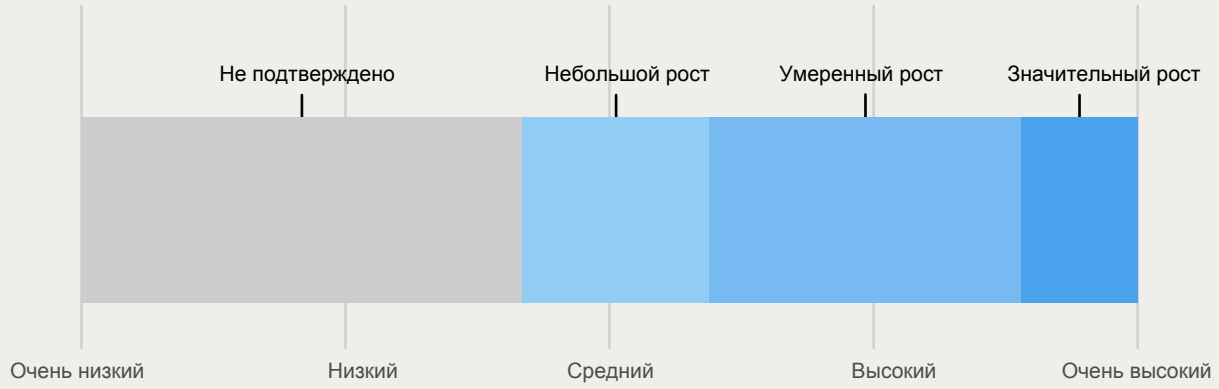


Рисунок 3: Четкая и коммуникабельная позиция ИИ смягчает влияние ИИ на индивидуальную эффективность

Четкая и коммуникативная позиция ИИ определяет влияние ИИ на организационную эффективность

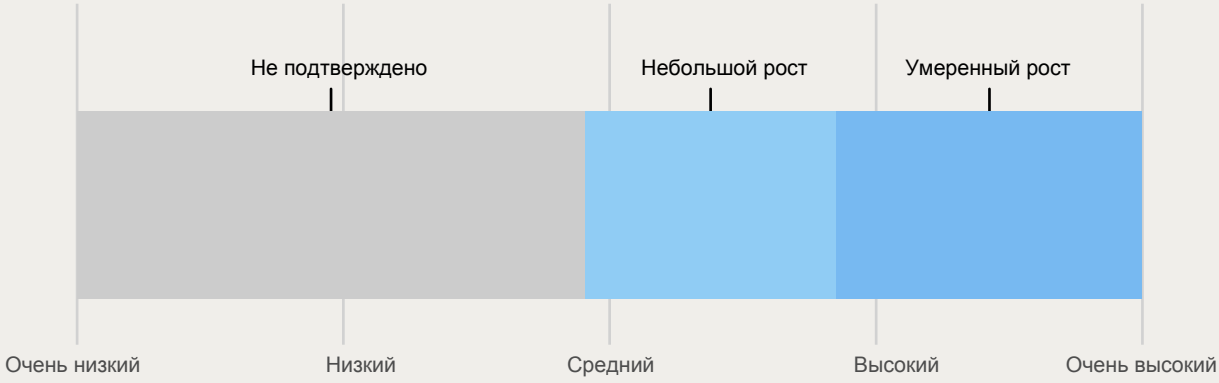


Рисунок 4: Четкая и коммуникативная позиция ИИ определяет влияние ИИ на организационную эффективность

Четкая и коммуникативная позиция ИИ смягчает влияние ИИ на пропускную способность



Рисунок 5: Четкая и коммуникативная позиция ИИ смягчает влияние ИИ на пропускную способность

Четкая и коммуникабельная позиция ИИ смягчает влияние ИИ на трение

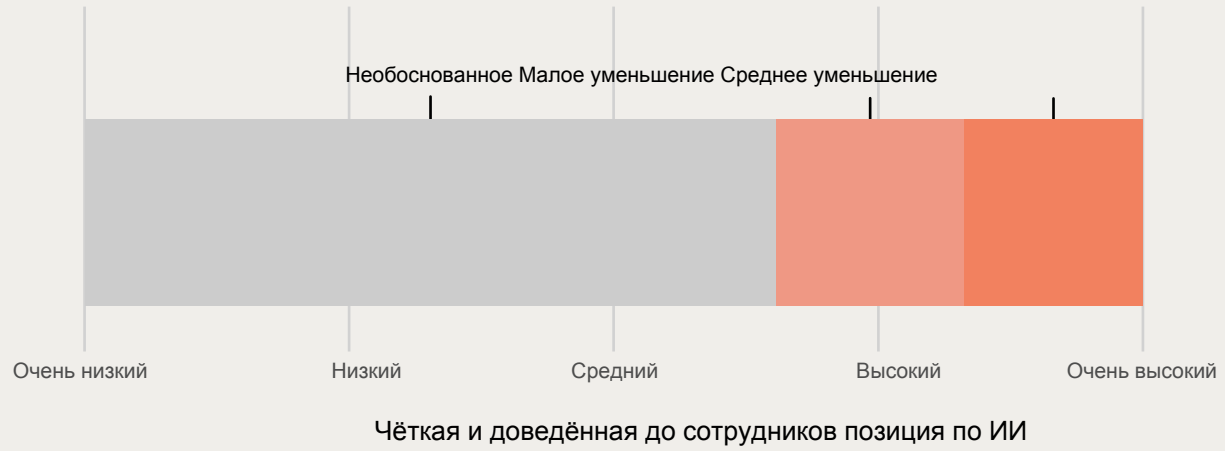


Рисунок 6: Четкая и коммуникабельная позиция ИИ смягчает влияние ИИ на трение

Как применять правила использования ИИ в повседневной работе

Построение четкой и сообщаемой позиции - это преднамеренное, кросс-функциональное усилие, которое переходит от исполнительного видения к ежедневному использованию. Это путешествие, а не одноразовый контрольный список.

Руководство должно определить и отстаивать четкую миссию, цели и план внедрения ИИ. Это сигнализирует о стратегической важности ИИ, выделяет ему необходимые ресурсы и предоставляет полномочия для принятия всеобъемлющей политики.

Однако это видение лидерства не может быть реализовано из одного блока. Политика, созданная только законом или безопасностью, вряд ли будет работать в реальности.

Позиция должна быть написана межфункциональной рабочей группой с представителями из инженерных, юридических, безопасности, ИТ и лидерства продукта. Группа такого рода уникально позиционируется, чтобы сбалансировать управление рисками с практическими реалиями рабочих процессов разработчиков.

Эта группа должна также определить четких, долгосрочных владельцев политики, которые будут отвечать за проверку изменений и сортировки предложений из цикла обратной связи. Для обеспечения понятности результатов эта группа должна принять простое руководство по стилю для самой политики. Это руководство должно подчеркивать использование простого языка, предоставлять конкретные примеры «делай» и «не делай» и максимально избегать внутреннего жаргона и законности.

Основной вывод этой группы должен быть не простым документом «да/нет», а тонкой, основанной на риске структурой. В качестве отправной точки это руководство по разработке приемлемой политики использования для генеративного AI4 предлагает практические советы о том, как построить структуру и сбалансировать потребности заинтересованных сторон.

Эффективная модель должна служить документированным набором критических вариантов использования ИИ и использовать подход «три корзины» для категоризации инструментов и данных:

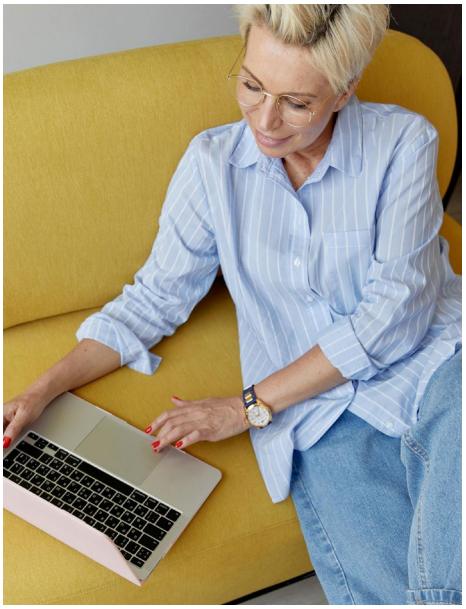
Запрещено: Высокорисковое, неприемлемое использование (например, ввод личной информации или коммерческой тайны любого клиента в общедоступную модель ИИ).

Разрешено с ограждением: Допускается только с определенными элементами управления (например, использование проприетарного исходного кода только с утвержденным инструментом искусственного интеллекта корпоративного уровня; требуется обзор человека в петле для всего кода, созданного ИИ).

Допускается: деятельность с низким уровнем риска и высокой ценностью, которая активно поощряется (например, генерация кода на основе шаблонов, мозговой штурм идей без использования собственных данных).

Этот фреймворк должен быть опубликован как «живой документ» в центральном, легкодоступном для поиска хабе, таком как внутренний портал разработчика или вики. В этом хабе должны размещаться политика, официальный список утвержденных инструментов и развивающиеся часто задаваемые вопросы.

Наконец, политика полезна только в том случае, если она известна. Ее запуск должен быть социализирован через «городские залы», встречи команд и формальное обучение. Самое главное, это сообщение должно установить четкую петлю обратной связи для разработчиков, чтобы задавать вопросы и предлагать обновления, позволяя политике развиваться по мере изменения потребностей технологий и бизнеса.



Что мешает сформулировать и донести позицию по ИИ

Рассматривать политику как статический документ

Технологии и рабочие процессы, которые позволяют ИИ оказывать влияние на важные результаты, быстро меняются. Организации должны использовать уроки, извлеченные из экспериментов с ИИ, для регулярного обновления своей позиции в отношении ИИ и приемлемой политики использования.

Слишком часто меняться

Изменения в политике требуют времени для полного включения команд. Политика, которая всегда меняется, может быть хуже, чем отсутствие политики вообще. Убедитесь, что у команд есть время и стимул адаптироваться к изменениям в позиции ИИ.

Миопический взгляд на политику

Позиция ИИ должна уравновешивать потребности всех заинтересованных сторон, а не только одной группы.

Как оценить ясность позиции по ИИ

Самый прямой способ измерить эту способность - опрос ваших команд, чтобы оценить восприятие и ясность разработчиков.



Вопросы опроса DORA

Исследование DORA измерило эту способность на основе ответов на эти вопросы: 5

• В какой степени использование ИИ на работе является обязательным?

В какой степени ваша организация поддерживает вас в экспериментировании с ИИ?

В какой степени ясно, как вам разрешено и не разрешено использовать ИИ на работе?

В какой степени политика ИИ вашей организации непосредственно относится к вашей работе?
В какой степени ясно, какие инструменты ИИ вам разрешено и не разрешено использовать на работе?

Дополнительные меры

Вы также можете измерить осведомленность о политике, отслеживая просмотры страниц для политики и количество уточняющих вопросов о политике ИИ в общедоступных каналах.

Выводы

Четкая и коммуникабельная позиция ИИ является важной основой для успешной интеграции ИИ. Его эффективность зависит от ясности, которая уменьшает трения и дает разработчикам уверенность в безопасном внедрении инноваций. Эта способность работает в соответствии с другими шестью возможностями в модели возможностей ИИ DORA для достижения полного, положительного воздействия ИИ.

Следующая возможность сосредоточена на создании высококачественного, надежного источника информации для потребления ИИ, что требует создания здоровых экосистем данных.

1. «Повышение доверия разработчиков к генеративному искусственному интеллекту». https://dora.dev/research/ai/trust-in-ai
2. «Обеспокоенность превыше точности вывода ИИ». https://dora.dev/research/ai/concerns-beyond-accuracy-of-ai-output
3. «Помощь разработчикам в принятии генеративного ИИ: четыре практических стратегии для организаций». https://dora.dev/research/ai/adopt-gen-ai
4. «Как разработать политику приемлемого использования для ИИ поколения (и выглядеть умным, делая это)». https://cloud.google.com/transform/how-to-craft-an-acceptable-use-policy-for-gen-ai-and-look-smart-doing-it
5. «Опросы: политика в области ИИ». https://dora.dev/ai/capabilities-model/questions/#clear-and-communicated-ai-stance



Зрелая экосистема данных

Амир Аббас

Менеджер по продуктам, Google Cloud

Люсия Субатин

Разработчик Advocate, Google Cloud

Здоровая экосистема данных усиливает положительное влияние внедрения ИИ на организационную эффективность.

Зачем ИИ нужна зрелая экосистема данных

Когда организации инвестируют в создание и поддержание высококачественных, доступных, унифицированных экосистем данных, они могут принести еще более высокие выгоды для их организационной эффективности, чем только с внедрением ИИ.



Здоровая экосистема данных относится к общему качеству и полезности внутренних систем данных организации. Наше исследование в области развития с помощью ИИ определяет здоровые экосистемы как основополагающую способность, определяемую степенью, в которой внутренние данные являются высококачественными, легко доступными и унифицированными.

В организации со здоровой экосистемой данных:

Внутренние данные надежны и надежны, что обеспечивает их пригодность для использования в качестве контекста для инструментов ИИ.
Данные очень доступны для всех необходимых команд и систем, включая разработчиков и инструменты ИИ.

Данные не разделены и не разделены, что способствует целостному, унифицированному взгляду на всю организацию.

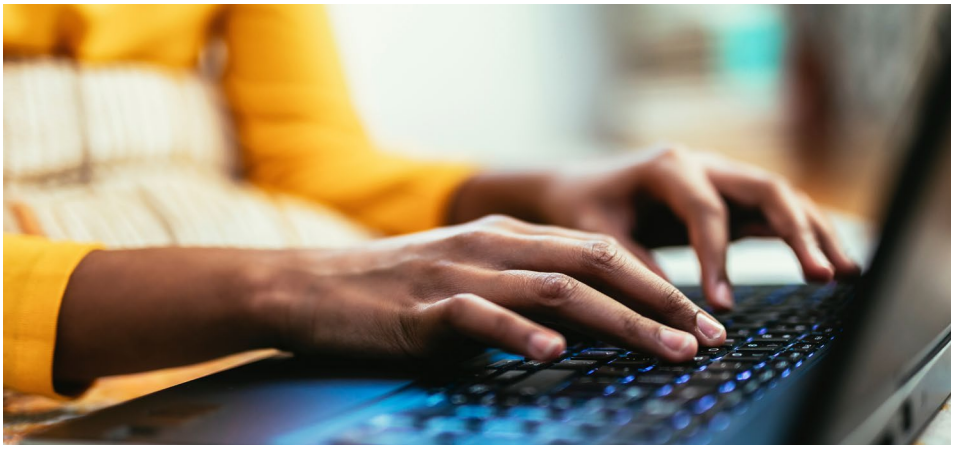
Системы созданы для обеспечения целостности и непрерывной доступности данных, что делает их надежным ресурсом как для лиц, принимающих решения, так и для передовых систем с искусственным интеллектом.

Ценность внедрения ИИ обусловлена; он разблокирован не только самими инструментами, но и окружающей технической и культурной средой. Наше исследование подтверждает, что ИИ функционирует как усилитель,1 увеличивая сильные стороны или дисфункции существующих организационных систем.

Здоровая экосистема данных является важной предпосылкой, которая определяет общий успех внедрения ИИ, особенно в максимизации отдачи от инвестиций.

Повышение организационной эффективности

Когда организации инвестируют в создание и поддержание высококачественных, доступных и унифицированных экосистем данных, положительное влияние внедрения ИИ на эффективность организации значительно усиливается.



Здоровье экосистемы данных смягчает влияние ИИ на организационную эффективность

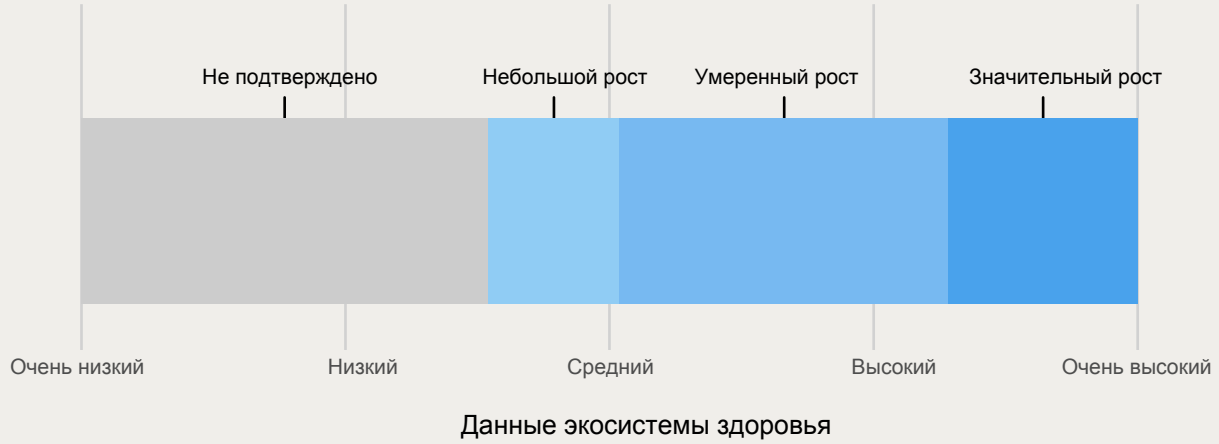


Рисунок 7: Здоровье экосистемы данных смягчает влияние ИИ на организационную эффективность

Предоставление контекстуально значимой помощи

Здоровая экосистема данных имеет важное значение для разработки сложных, контекстно-осведомленных ИИ-помощи. Это необходимо для таких задач, как точная настройка основополагающих моделей на проприетарных знаниях или внедрение эффективных систем с расширенным поиском (RAG), которые требуют высококачественного поиска данных.

Уменьшение хаоса вниз по течению

Когда среда данных низкого качества, ИИ рискует создать локализованный прирост производительности, который «поглощается узкими местами вниз по течению», что приводит к небольшой реальной ценности для организации в целом. Инвестирование в здоровье данных гарантирует, что индивидуальный прирост производительности преобразуется в системные организационные преимущества.

Общие препятствия для здоровой экосистемы данных

Культурные сдвиги необходимы для создания здоровых экосистем данных. Организационные лидеры должны спонсировать переход от обработки данных в качестве побочного продукта. Это позволяет рассматривать его как продукт с потребителями, которые нуждаются в согласованной долгосрочной стратегии.

Инструмент представляет собой силос: зрелая платформа данных сможет управлять различными источниками и их конкретными подходами к хранению и моделированию данных, включая различные инструменты для приема пищи и схемы. В то время как некоторые инструменты являются более самоуверенными, чем другие, часть взлома бункеров данных нарушает самоуверенный подход от конкретных инструментов, когда это необходимо для достижения общей точки зрения.

Приоритетность правильного инструмента для работы и платформы, а также наличие гибких интеграций и правильно документированных схем становится особенно актуальной, когда внутренние данные могут дополняться внешними источниками в качестве подготовки к RAG.

Как улучшить здоровье экосистемы данных

Улучшение экосистем данных требует стратегических организационных инвестиций, поддерживаемых принципами непрерывного совершенствования, такими как непрерывная интеграция и непрерывная доставка.

Спонсорство со стороны лидеров Создание и постоянное совершенствование основополагающих возможностей требует вовлеченных лидеров, которые обладают полномочиями и бюджетом для осуществления крупномасштабных изменений и обеспечения видимой поддержки инициативы.2 Лидеры играют ключевую роль в содействии необходимым культурным и техническим сдвигам.

Например, руководители должны назначить конкретную команду для владения «данными учетной записи клиента» и нести ответственность за ее точность, метаданные и политику доступа.

Ясность в отношении владения и процессов имеет основополагающее значение для поддержания качества документации3 и, в более широком смысле, целостности данных. Четкое управление обеспечивает подотчетность и определяет обязанности по поддержанию, обновлению и проверке данных.

Цель состоит в том, чтобы создать «проложенный путь» безопасных, хорошо задокументированных и надежных путей передачи данных, уравновешивая средства управления безопасностью с необходимостью сделать данные доступными для обнаружения.

Многие организации хоронят информацию в разрозненных системах и неформальных каналах знаний. Команды могут решать проблему изолированных и разделенных данных путем выявления ключевых источников данных и работы по их консолидации или федерации для создания единого представления, доступного бизнес-пользователям.

Создание единого представления может потребовать создания рабочих процессов, которые автоматически помечают и отображают контекст данных во время разработки и обзора, сокращая время, которое разработчики тратят на поиск и улучшение качества вывода ИИ. Эти рабочие процессы уменьшают трение для разработчиков и систем ИИ, пытающихся получить доступ к необходимому контексту, и позволяют бизнес-пользователям предоставлять обратную связь по одним и тем же данным, чтобы избежать расхождений вниз по течению.

Внедрение рамок качества данных В соответствии с принципом построения качества в,4 мы рекомендуем ввести автоматизированные проверки и мониторинг для обеспечения точности, полноты и своевременности данных. Автоматизированные методы проверки отражают необходимость автоматического, непрерывного тестирования на протяжении всего жизненного цикла разработки программного обеспечения, обеспечивая надежность и пригодность данных для целей.

Для демократизации доступа к данным используйте инструменты и платформы, которые позволяют командам легко находить и получать доступ к необходимым данным в сочетании с средствами контроля безопасности. Повышение доступности напрямую поддерживает создание внутренних данных, доступных для ИИ, позволяя инструментам ИИ выходить за рамки общей помощи, предоставляя специфический для компании контекст, необходимый для улучшения качества кода и индивидуальной эффективности.

Документировать и обмениваться данными локально команды могут начать с документирования ключевых наборов данных, которые они производят и потребляют. Например, команда может согласиться добавить и поддерживать раздел «данные» в README.md приложения, который включает информацию о том, какие данные создаются или используются этим приложением.

В идеале эта документация и метаданные (например, описания таблиц и столбцов) рассматриваются как артефакт кода. Они редактируются, живут вместе с данными и одинаково доступны для всех типов потребляющих приложений и в разных средах.

Как оценить зрелость экосистемы данных

Прогресс в создании здоровых экосистем данных должен оцениваться с помощью комбинации мер восприятия и объективных показателей, направляющих организацию к постоянному совершенствованию.



Вопросы опроса DORA

Исследование DORA измерило эту способность, задав следующие вопросы: 5

Насколько легко вы можете получить доступ и проанализировать внутренние источники данных, необходимые для выполнения вашей работы?

Как часто вы не можете использовать важные внутренние данные, потому что данные изолированы?

Как бы вы оценили общее качество данных, на которые вы обычно полагаетесь для своей работы?

Если вам нужна конкретная точка данных, насколько вероятно, что вы можете получить окончательный ответ в течение 1 часа после поиска?



Дополнительные меры

Дополнительные сигналы, которые могут помочь измерить здоровье экосистемы данных, включают:

Что проверить	Что измерить
Своевременность данных	Прошедшее время, необходимое разработчику для получения доступа к необходимому набору данных для нового проекта Эта метрика служит ключевым индикатором потока, подобно измерению времени выполнения изменений, помогая выявлять и устранять ограничения в предоставлении данных и доступе к рабочим процессам.
Инциденты данных	Количество ошибок, производственных инцидентов или дефектов, о которых сообщают клиенты, которые окончательно связаны с плохим качеством данных Это может выступать в качестве прокси для незапланированной работы и переработки, демонстрируя прямое влияние дисфункции данных на нестабильность доставки программного обеспечения и операционную производительность.
Качество данных	Скорость прохождения автоматизированных проверок, включая точность, полноту и своевременность
Свежесть данных	Метаданные из источника данных (например, «последние обновленные» метки времени)

Выводы

В конечном счете, здоровая экосистема данных — это не дополнительная инвестиция, а фундаментальный усилитель организационной эффективности. Успех внедрения ИИ зависит от качества, доступности и единства этих данных.

Здоровые экосистемы данных обеспечивают надежную, контекстно-осознанную основу для всей Модели возможностей ИИ DORA. Однако высококачественные данные полезны только в том случае, если инструменты ИИ могут безопасно подключаться к ней. Для этого требуется следующая возможность: сделать внутренние данные ИИ доступными.

1. Состояние разработки программного обеспечения с помощью ИИ. https://dora.dev/dora-report-2025
2. 2017 State of DevOps Report. https://dora.dev/dora-report-2017
3. «Возможности DORA: качество документации». https://dora.dev/capabilities/documentation-quality
4. 2016 Состояние отчета DevOps. https://dora.dev/dora-report-2016
5. «Опросы: экосистема данных». https://dora.dev/ai/capabilities-model/questions/#healthy-data-ecosystems



Внутренние данные, доступные для ИИ

Роб Эдвардс

Доставка приложений Lead, Google Cloud

Наличие доступа к внутренним данным усиливает положительное влияние внедрения ИИ на:

- Индивидуальная эффективность
- Качество кода

Зачем ИИ нужна зрелая экосистема данных

Чтобы инструменты ИИ были полезны, они должны быть не просто помощниками: они должны стать специалистами, которые понимают вашу организацию. Чтобы сделать этот скачок, они должны быть надежно связаны с вашими внутренними данными.

Наше исследование подтверждает, что предоставление командам инструментов ИИ, которые могут получить доступ к этим внутренним данным, напрямую повышает их эффективность и качество кода.

Думайте об этом как о полном контексте, который нужен разработчику для достижения цели, которая включает в себя запатентованные данные компании, такие как кодовые базы, вики, архитектурные диаграммы, документацию и руководства по стилю, все из которых могут управляться с помощью ручной и автоматизированной контекстной инженерии.

СОВЕТ: Хотя это не строго внутренние данные, важно сделать доступной текущую внешнюю документацию для внутренних инструментов ИИ. Учебные данные ИИ - это статичный снимок; предоставление современных документов для фреймворков и языков вашей команды имеет важное значение для предотвращения создания устаревшего кода.

Системный подход к контексту для ИИ

Многие люди знакомы с быстрой инженерией, которая является искусством написания конкретной команды для ИИ. Чтобы разблокировать реальную ценность, мы должны перейти к контекстной инженерии, чтобы гарантировать, что модель производит желательные, надежные и контролируемые результаты.1

Простой подсказка - это как дать своему помощнику ИИ один заказ. С другой стороны, контекстная инженерия - это как дать ему полный пакет брифингов, прежде чем они начнут работать.

Этот пакет должен содержать весь контекст, необходимый для правильного выполнения работы, включая:

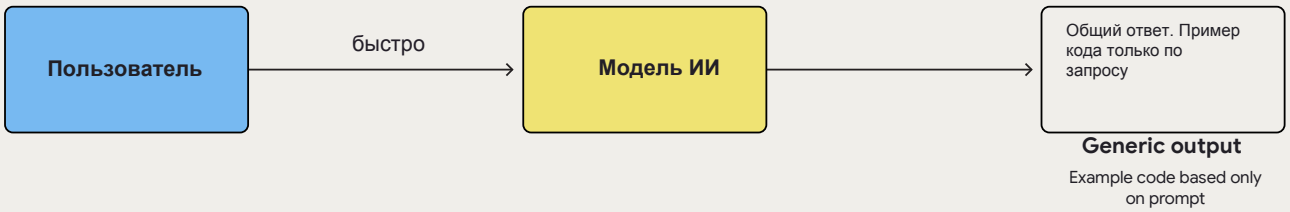
Данные компании: «Вот наша внутренняя кодовая база и наше руководство по стилю кодирования».

«Вот самые современные документы интерфейса прикладного программирования (API) для библиотеки, которую вы используете».

Инструменты и правила: «Вот правила, которым вы должны следовать, и инструменты, которые вы можете использовать для достижения или проверки вашей работы»2.

Вместо одноразовой строки контекстная инженерия - это система, которая автоматически собирает эту соответствующую информацию, а затем использует ее для управления интеллектуальным рабочим процессом. Эта система позволяет итеративным циклам уточнения, где ИИ поручается генерировать первый черновик, критиковать свою собственную работу против контекста и улучшать свой ответ.3 Она позволяет ИИ перейти от общего «чат-бота» к специализированному, экспертному помощнику, который понимает вашу компанию.

Быстрая инженерия



Контекстная инженерия

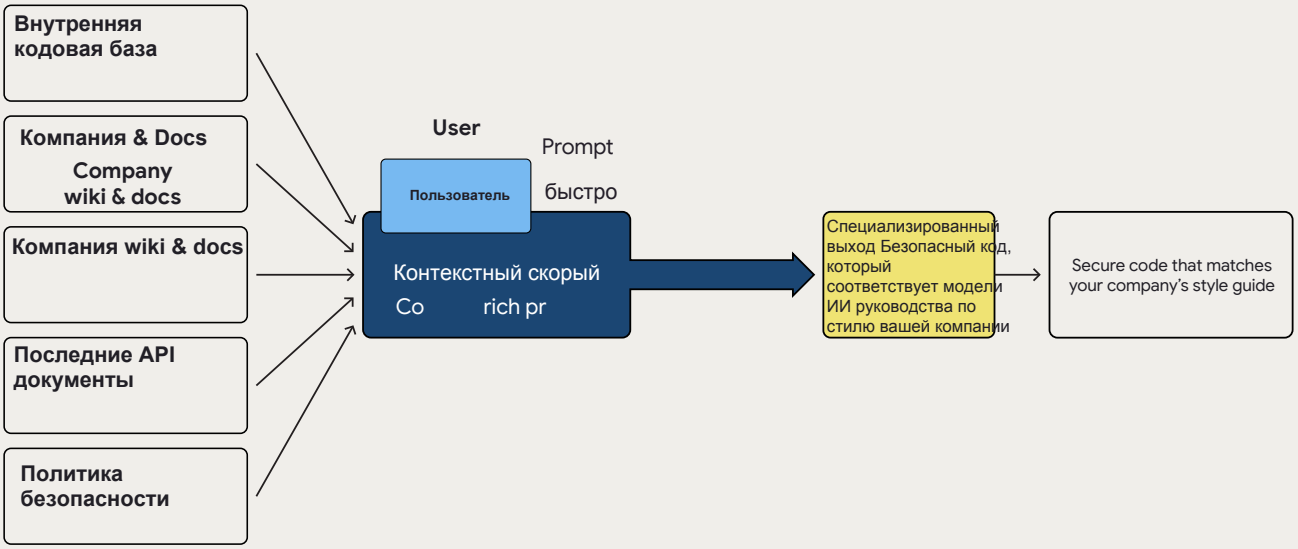


Рисунок 8: Пример сравнения быстрой инженерии с контекстной инженерией

Зачем ИИ доступ к внутренним данным

Подключение ИИ к полному контексту может поднять ИИ от помощника по кодированию общего назначения до специализированного эксперта с глубоким знанием конкретной архитектуры, шаблонов и бизнес-логики вашей организации.

Повышение эффективности разработчика: сокращение времени, которое разработчики тратят на сбор контекста, включая чтение документации, расшифровку существующего кода и прерывание работы старших инженеров, путем предоставления мгновенных ответов, учитывающих контекст.

Улучшение качества и согласованности кода: предложения ИИ начинают согласовываться с внутренними конвенциями и передовыми практиками кодирования, что приводит к созданию кода, который с большей вероятностью будет правильным и эффективным в ваших конкретных системах.

Смягчить технический долг: работая с новейшей рамочной и языковой документацией, ИИ избегает предлагать устаревшие функции и устаревшие шаблоны, помогая командам создавать более удобный и безопасный код с самого начала.

Ускорить абордаж и обмен знаниями: сделать институциональные знания более доступными.Новые разработчики могут быстрее набирать скорость, задавая вопросы ИИ о кодовой базе, сокращая перерывы для штатных членов команды.

Эти возможности являются мощными усилителями преимуществ внедрения ИИ. Исследование DORA показывает с высокой степенью уверенности, что внутренние данные, доступные ИИ, увеличивают положительное влияние ИИ на ключевые результаты.

В частности, он усиливает индивидуальную эффективность и качество кода. Способность ИИ предоставлять ответы, адаптированные к уникальному контексту вашей организации, является ключевым отличием между командами, которые видят маргинальную выгоду от ИИ, и теми, которые достигают поэтапного изменения производительности.

Внутренние данные, доступные для ИИ, смягчают влияние ИИ на индивидуальную эффективность

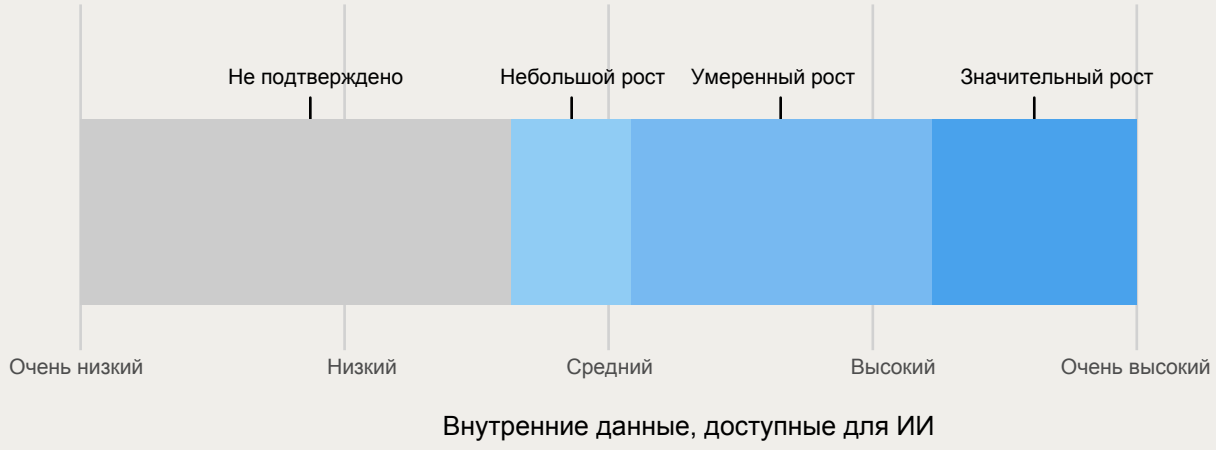


Рисунок 9: Внутренние данные, доступные для ИИ, смягчают влияние ИИ на индивидуальную эффективность

Внутренние данные, доступные для ИИ, смягчают влияние ИИ на качество кода

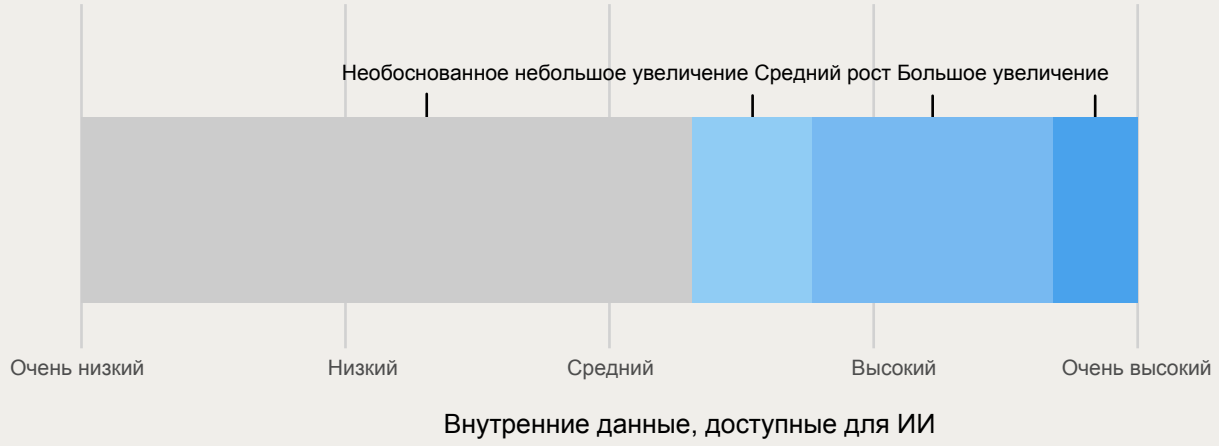


Рисунок 10: Внутренние данные, доступные для ИИ, смягчают влияние ИИ на качество кода

Как улучшить доступ к внутренним данным

Улучшение доступа к внутренним данным предполагает поэтапный подход, начиная с ручных методов и заканчивая более систематическим, автоматизированным решением.

Этап 1: Основополагающий и ручной

Начните с развития ваших навыков быстрой инженерии в ручной контекстной инженерии. Команды должны быть наделены полномочиями выходить за рамки простого написания запроса и собирать контекст, в котором ИИ должен ответить на него.

Обеспечить пространство для команд, чтобы узнать, как вручную найти и прикрепить наиболее релевантную информацию (например, конкретные API-документы, руководства по стилю и фрагменты кода) к их запросам. Создать общую библиотеку этих многоразовых шаблонов контекста или пакеты брифингов для общих задач.

Самое главное, надежный контекст полностью зависит от доступных и точных внутренних данных. Вы не можете подключить модель или приложение ИИ к информации, которой не существует, и вы не должны подсоединять его к информации, которая неверна. Работайте с командами, чтобы сначала определить и улучшить наиболее важную внутреннюю документацию.

Фаза 2: Автоматизированный пилот

Определить, следует ли создавать индивидуальное решение или закупать коммерческий инструмент, который может быть надежно интегрирован. После того, как инструмент выбран, реализовать автоматизированное решение поиска для одного случая использования с высокой отдачей.

Этот шаг подчеркивает важность контекстной инженерии. В то время как подача больших, сырых документов в контекстное окно ИИ заманчива, это часто контрпродуктивно. Этот подход часто приводит к галлюцинациям: выходы, которые звучат уверенно, но фактически ложны.

Вместо этого эффективные решения сосредоточены на предоставлении только наиболее конкретного контекста для текущего запроса. Для этого возникают две основные модели:

• Дополненная генерация (RAG):4 Более сложный, но точный метод поиска и предоставления только самой актуальной, актуальной информации, которая может быть централизованной.

Сервер типового контекстного протокола (MCP): 5 Подход, который разумно выбирает, структурирует и передает только наиболее релевантный контекст для ИИ, а не целые исходные документы.

Эта фаза перемещает иглу от ручного ввода к автоматическому поиску, что является ключевым шагом в созревании этой способности.

Фаза 3: Получение спонсорства, масштабирование и реализация

Используйте успех пилота для получения спонсорства от лидеров для более широких стратегических инвестиций. Эти инвестиции имеют решающее значение для масштабирования возможностей, которые часто включают в себя решение фундаментальных проблем данных и создание надежных, безопасных внутренних API, чтобы сделать больше данных доступными для ваших систем ИИ. Для документации это может включать интеграцию с услугой, которая обеспечивает современную документацию программно.

Оригинальное название: Onboarding a New Engineer

Представьте себе нового инженера, который присоединился к команде, отвечающей за комплексный биллинговый микросервис.

Без данных, доступных для ИИ, их погружение включает в себя дни чтения устаревшей документации, прерывания старших инженеров вопросами и медленного объединения мысленной модели системы.

С помощью данных, доступных для ИИ, они могут напрямую спросить помощника ИИ: «Каков процесс выдачи возврата и какие услуги задействованы?» ИИ, используя RAG для доступа к последним спецификациям API, архитектурным диаграммам и самой кодовой базе, предоставляет пошаговый ответ со ссылками на точные функции и документы.

Это не только ускоряет процесс адаптации, но и дает возможность новому инженеру вносить свой вклад быстрее и увереннее.

Пример: ИИ-ассистированный обзор кода

Подумайте о том, чтобы разработчик отправил запрос на тягу (PR) для новой функции.

Без данных, доступных для ИИ, рецензент должен вручную проверить приверженность руководству по стилю команды. Они также могут иметь смутное воспоминание о подобных дебатах по предыдущему PR, но не могут найти ссылку, что приводит к непоследовательным отзывам или повторным дискуссиям.

С помощью данных, доступных для ИИ, агент для анализа кода может быть настроен на использование внутреннего руководства по стилю и базы данных прошлых обсуждений обзора кода в качестве контекста. Он может автоматически комментировать:

«Это отличное начало. Я заметил, что схема обработки ошибок здесь отличается от той, которую мы установили для этой службы в PR #1234. Чтобы сохранить последовательность, давайте придерживаться этого подхода. Вы можете увидеть обсуждение и рассуждение здесь: [ссылка на PR]».

Доступные для ИИ данные могут помочь разгрузить утомительные проверки согласованности от рецензентов-людей. Он также может адаптировать обратную связь ИИ к установленным нормам организации, позволяя команде сосредоточиться на архитектурной и логической обоснованности изменений.

Общие препятствия для доступности данных

Препятствие: плохое качество или «беспорядочные» внутренние данные

Многие организации борются с внутренними данными, которые являются устаревшими, неточными и распространяются по нескольким отключенным системам. Одна из наиболее распространенных проблем, которую мы видим, заключается в том, что искусственный интеллект, подключенный к плохим данным, будет давать плохие ответы.

Как отметил один из участников исследования, многие компании «даже не на той стадии, когда у них должным образом организованы данные». ИИ, подключенный к плохим данным, будет давать только плохие ответы.

Решение: не кипятите океан. Первым важным шагом в вашем путешествии «сначала качество данных» является назначение команды для пилотного проекта, связанного с конкретным приложением или услугой. Команда должна стремиться повысить точность и фокусировку доступной информации.

«Месси» данные включают в себя документацию, которая просто слишком длинная. Например, 20-страничный стандартный документ часто является слишком большим для модели, чтобы обращать внимание, а также пытаться решить запрос пользователя.

• Идентификация единственного наиболее ценного и достаточно хорошо поддерживаемого источника данных (например, документации API для одной критической службы).

• Используйте ИИ в качестве отправной точки для перегонки этой информации, затем попросите эксперта по человеческой тематике проверить, отредактировать и одобрить это дистиллированное резюме (например, GEMINI.md).6

Сосредоточьтесь на том, чтобы сделать этот высококачественный дистиллированный источник доступным для вашего пилота ИИ.

• Измерить улучшение качества отклика ИИ для этой конкретной услуги, создав бизнес-кейс для масштабирования усилий.

• Приоритетность декларативных подходов и рамок, которые позволяют проводить итерации, автоматизированные конвейеры развертывания и тестирование, включая утверждения о качестве данных.

Препятствие: риск загрязнения ИИ плохими примерами

Общим импульсом является индексация всех репозиториев кода для максимизации доступного контекста. Однако это может быть контрпродуктивно.

Если ваши индексы включают устаревшие проекты, экспериментальные филиалы и код, который нарушает существующие передовые методы, ИИ будет учиться на этом «плохом» коде так же легко, как и на «хорошем» коде.

Решение: Изучите контекст кода. Будьте избирательны и стратегически настроены в отношении того, что вы индексируете. Начните с индексации только репозиториев и филиалов (например, основных), которые представляют собой золотой стандарт вашей организации для стиля кода, архитектуры и текущих шаблонов.

Курирование контекста кода гарантирует, что ИИ учится на ваших лучших примерах, усиливая хорошие практики, а не плохие. Вы можете постепенно расширять индекс, поскольку вы проверяете качество других источников кода, рассматривая свой индексированный код как «учебную программу» для вашего помощника по ИИ.

Препятствие: контекстная гниль

Многие команды предполагают, что «больше данных лучше» и стараются предоставить как можно больше контекста. Однако большие языковые модели имеют конечные окна контекста.

Подавление модели чрезмерной, нерелевантной или низкой плотностью информации может быть столь же вредным, как и предоставление слишком малого количества. Ключ, соответствующий сигнал может быть «потерян», что приводит к не сфокусированным, неправильным и общим ответам, которые игнорируют важные детали.

Решение: Сосредоточьтесь на релевантности, а не на объеме. Именно здесь точность и понимание опыта могут помочь с контекстной инженерией. Вместо больших свалок данных реализуйте точные механизмы поиска, такие как RAG.

RAG предназначен для поиска обширного корпуса (вся ваша вики, весь ваш код), но извлекает только конкретные «куски» информации, которые наиболее актуальны для немедленного запроса пользователя. Он гарантирует, что подсказка заполнена высокосигнальными, релевантными данными, а не раздута шумом. Этот акцент на релевантности приводит к появлению дисциплины сбора контекста, активного процесса фильтрации для релевантности, а не объема.

Препятствие: Проблемы безопасности и контроля доступа

Основной проблемой является обеспечение того, чтобы инструмент ИИ уважал существующие разрешения и контроль доступа к конфиденциальным данным.Связанным с этим основным препятствием является контроль того, к каким службам ИИ (например, утвержденному серверу MCP) пользователь может подключиться, и обеспечение безопасности самих этих служб.

Решение: Внедрить многоуровневую модель безопасности с наименьшими привилегиями. Этот стратегический подход согласуется с принципами, изложенными в Secure ИИ Framework (SAIF),7 - отраслевой модели, которая обеспечивает всеобъемлющее руководство по управлению рисками ИИ.

• Для доступа к данным: при использовании автоматизированных систем, таких как RAG, убедитесь, что механизм поиска работает с собственными учетными данными пользователя. Это гарантирует, что система может только «видеть» и извлекать документы, к которым пользователь уже имеет право доступа.

• Для доступа к услугам: не допускайте специальных подключений к непроверенным внешним службам ИИ. Вместо этого направляйте запросы через утвержденный централизованный прокси-сервер или MCP-сервер, который проверяется, контролируется и управляется ИТ. Это может помочь смягчить эксфильтрацию данных к несанкционированным моделям и обеспечить, чтобы все взаимодействия ИИ соответствовали политике безопасности компании.



Как оценить доступность внутренних данных



Вопросы опроса DORA

Исследование DORA измерило эту способность, задав эти вопросы: 8

Как часто вы вводили или предоставляли внутреннюю информацию или данные компании (например, документы, код, электронные таблицы, текст) в инструмент ИИ?

Как часто вы использовали ИИ в своей организации, чтобы получить и использовать информацию из внутренних источников данных (например, для ответа на вопросы, создания отчетов или улучшения рабочих процессов)?

В какой степени модели имеют доступ к внутренним источникам данных (например, базам данных или кодовым базам) и приложениям (например, вики или системам управления работой)?

• Как часто при использовании инструмента искусственного интеллекта для вашей работы вам кажется, что ваши ответы или результаты использовали внутреннюю информацию или данные компании в качестве контекста (например, ссылаясь на конкретные внутренние проекты, данные или терминологию в своих объяснениях)?

Когда вы задавали инструменту искусственного интеллекта вопросы о внутренней компании (например, конкретные детали проекта, внутренние отчеты), как часто его ответ приводил вас к мысли, что он получил доступ к внутренней информации компании или данным для получения этого ответа?

Дополнительные меры

Могут быть и другие факторы, которые помогут вам оценить, насколько внутренние данные доступны для ИИ и как это помогает повысить производительность.

Что проверить	Что измерить
Частота событий поиска	Сколько раз система ИИ успешно выполняет запрос RAG
Скорость доступа к источникам данных	Какие источники данных доступны
Латентность поиска ИИ (time-to-context)	Время, прошедшее с момента, когда система ИИ идентифицирует необходимость контекста, до момента, когда она успешно извлекает эти данные.
Контекстные подсказки	Доля подсказок, которые «богаты контекстом» против «простых подсказок»
Новый разработчик на борту	Это может быть первое, десятое или другое изменение разработчика, которое поставляется в производство.

Разумно ожидать, что доступ к внутренним данным также увеличивает влияние ИИ на другие факторы, такие как когнитивная нагрузка. Для оценки этого вы можете включить вопросы опроса, которые используют шкалу с ответами, начиная от «очень не согласен» до «сильно согласен».

Ответы ИИ, учитывающие контекст, сократили мои усилия по поиску информации.

Информированные в контексте ответы ИИ улучшили мою способность понимать нашу кодовую базу.
• Информативные ответы ИИ требуют меньше усилий для проверки.
Использование контекстно-осведомленных ИИ-помощников сокращает количество времени, которое я трачу на поиск информации.

Выводы

Подключение ИИ к внутреннему коду и документации, или «контекстная инженерия», превращает ИИ из универсального помощника в специализированного эксперта.

Внутренние данные, доступные для ИИ, напрямую повышают индивидуальную эффективность и качество кода, обеспечивая соответствующий внутренний контекст.

По мере того, как этот улучшенный ИИ ускоряет объем и скорость генерации кода, он создает новые риски.

Чтобы безопасно управлять этой повышенной скоростью, команды должны полагаться на основную систему безопасности, обеспечиваемую следующей возможностью: сильные методы контроля версий.

1. «Возникновение «контекстной инженерии»». https://blog.langchain.com/the-rise-of-context-engineering
2. «Правила файлов для более безопасного кодирования вибрации». https://www.wiz.io/blog/safer-vibe-coding-rules-files
3. «Продвижение методов генерации безопасного кода: систематическое исследование». https://arxiv.org/abs/2407.07064
4. «Что такое генерация с расширенным поиском (RAG)?» https://cloud.google.com/use-cases/retrieval-augmented-generation
5.«Что такое МЦП и как она работает?» https://cloud.google.com/discover/what-is-model-context-protocol?e=48754805&hl=en
6. "Предоставить контекст с файлами GEMINI.md". https://geminicli.com/docs/cli/gemini-md/
7. «Перспективы облачного CISO: ИИ как стратегический императив для управления рисками» https://cloud.google.com/blog/products/identity-security/cloud-ciso-perspectives-ai-strategic-imperative-to-manage-risk
8. "Опросы: внутренние данные". https://dora.dev/ai/capabilities-model/questions/#ai-accessible-internal-data



Надёжные практики контроля версий

Роб Эдвардс

Доставка приложений Lead, Google Cloud

Дэйв Стэнк

Специалист по работе с клиентами Google Cloud

Сильные методы контроля версий усиливают положительное влияние внедрения ИИ на:

- Индивидуальная эффективность
- Эффективность команды

Основополагающие возможности для современной разработки программного обеспечения

Системы управления версиями являются основой современной разработки программного обеспечения, обеспечивая систематический способ управления и отслеживания изменений в файлах с течением времени. Такие системы, как Git,1 Apache Subversion,2 Mercurial,3 и Jujutsu4, позволяют командам разработчиков координировать работу, отслеживать историю проекта и поддерживать окончательный источник истины для всех активов проекта.

В качестве краеугольного камня высокопроизводительных технологических организаций управление версиями — это комплексное использование системы управления версиями для всех производственных артефактов.

Еще в 2014 году в отчете State of DevOps Report 5 компания DORA подчеркнула, что комплексное использование включает в себя не только код приложения, но и конфигурации системы, сценарии тестирования и развертывания, а также определения инфраструктуры.

В эпоху генеративного ИИ, когда объем и скорость генерации кода резко возрастают, наши исследования показывают, что зрелые привычки контроля версий более важны, чем когда-либо, для максимизации преимуществ ИИ при одновременном снижении его рисков.



Контроль версий обеспечивает несколько ключевых преимуществ:

Восстановление и стабильность: помогает сократить время восстановления после неудачного развертывания.Когда происходит ошибка, управление версиями облегчает определение причины сбоя и быстрое и надежное возвращение к последнему известному хорошему состоянию.

Скорость и пропускная способность: Это позволяет легко воссоздать среду для тестирования и устранения неполадок, что повышает пропускную способность.

Восстановление после стихийных бедствий и аудит: это гарантирует, что команды могут быстро и предсказуемо воспроизводить свои производственные услуги, обеспечивая полную прослеживаемость каждого изменения для целей соответствия.

Частые проверки и опора на возможности отката создают критическую психологическую сеть безопасности. Эта сеть безопасности позволяет командам разработчиков экспериментировать и внедрять инновации с уверенностью, зная, что они могут легко вернуться в стабильное состояние, если будет допущена ошибка.

Зачем нужен контроль версий при работе с ИИ

Растущее использование генеративного ИИ усиливает важность сильных практик контроля версий. Кодирование с помощью ИИ может увеличить пропускную способность, но наше исследование показывает, что оно также связано с повышенной нестабильностью доставки программного обеспечения. Сильные практики контроля версий являются необходимой системой безопасности, которая позволяет командам уверенно экспериментировать с кодом, созданным ИИ.

ИИ трансформирует принцип частого совершения небольших изменений из лучшей практики в критическую гарантию. По мере того, как развитие продвигается дальше в агентный мир, где помощники и агенты ИИ генерируют и применяют изменения кода автономно, эта дисциплина становится первостепенной. Каждое изменение, независимо от того, сделано ли оно человеком или ИИ, должно быть совершено как отдельная, атомная единица. Это создает четкий проверяемый след и позволяет немедленно и точно откатываться по мере необходимости.

Гигиена контроля версий (постоянная практика совершения небольших, частых и хорошо организованных обязательств) имеет решающее значение как для «внутреннего цикла» (местная работа разработчика по разработке предложения об изменении), так и для «внешнего цикла» (процесс обзора и слияния команды, чтобы в конечном итоге обеспечить его).

Практическое наблюдение показывает, что разработчики часто относятся к внутренней петле с меньшей строгостью, ожидая создания чистых обязательств перед открытием запроса на вытягивание. Более дисциплинированные методы контроля версий обычно появляются во внешней петле.

Кодирование с помощью ИИ коренным образом меняет то, как команды могут внедрять лучшие практики управления версиями. Инструменты ИИ могут дополнять разработчика, записывая код новыми способами, вводя мощный недетерминизм непосредственно в рабочий процесс разработчика.

Предложения ИИ по документированию контроля версий могут быть восхитительными прорывами или ошибочными обходными путями. Этот цикл «проб и ошибок» делает старую модель «обязанности, когда она готова» недостаточной.

Чтобы эффективно использовать ИИ, разработчики должны теперь привнести дисциплину внешней петли во внутреннюю петлю. Это означает создание небольших, частых обязательств в каждом чистом, функциональном состоянии. Когда изменение, генерируемое ИИ, нежелательно, его легко отбросить и вернуться в известное хорошее состояние.

Во внешней петле управление версиями обеспечивает основу для обзора изменений до их объединения. Предложение об изменениях может отражать работу любого количества агентов человека и ИИ, каждый из которых обладает уникальным набором знаний и предпочтений.

В то время как расхождение здесь может быть активом, как вектором для инноваций, конвергентный процесс анализа кода и автоматизированного тестирования необходим для обеспечения того, чтобы изменения соответствовали организационным стандартам и чтобы они соответствовали целям организации.

С высокой степенью уверенности мы обнаружили, что положительные преимущества внедрения ИИ зависят от частоты выполнения респондентами задач контроля версий. В частности, при наличии частых обязательств положительное влияние ИИ на индивидуальную эффективность усиливается.

Управление версиями фиксирует частоту смягчения влияния ИИ на индивидуальную эффективность

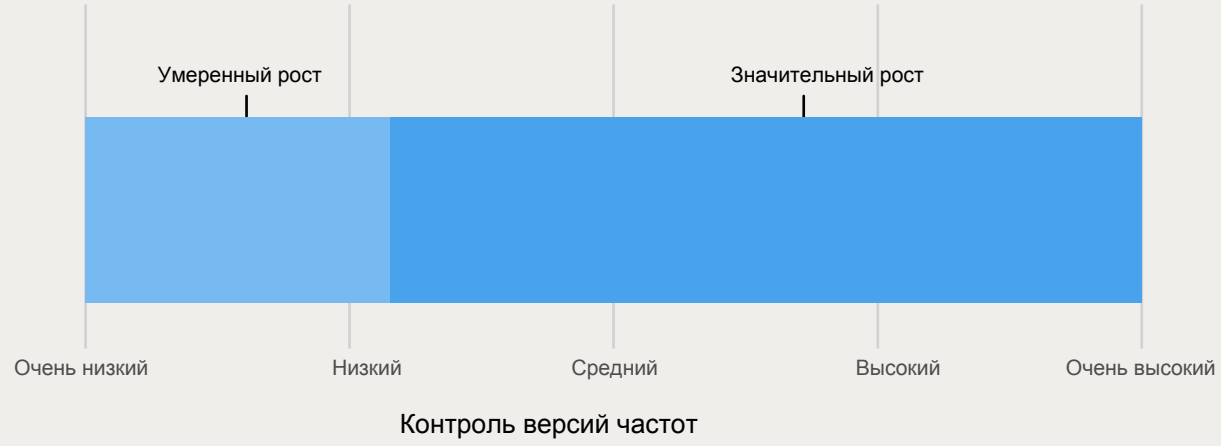


Рисунок 11: Управление версиями ограничивает влияние ИИ на индивидуальную эффективность

Кроме того, мы обнаружили, что положительные преимущества внедрения ИИ зависят от частоты использования респондентами функций «отката» своих систем управления версиями для отмены или возврата изменений. В частности, при наличии более частых откатов положительное влияние ИИ на производительность команды усиливается.

Эта зависимость от откатов для производительности команды, вероятно, связана с проблемой управления кодом, генерируемым ИИ.

Как показывают наши исследования, ИИ может увеличить нестабильность доставки программного обеспечения, часто создавая большие, труднопроверяемые пакеты кода. Способность быстро возвращать эти изменения становится критическим фактором в поддержании производительности команды.

Мы обнаружили, что около 21% респондентов нашего опроса начинают хранить подсказки ИИ в контроле версий. Срочные предложения содержат ценный контекст о задаче. Версируя их, команды могут делиться знаниями, совершенствовать свою способность получать ценность от ИИ и создавать аудиторский след для агентных рабочих процессов.

Способность отката смягчает влияние ИИ на производительность команды



Рисунок 12: Способность отката смягчает влияние ИИ на производительность команды

Как улучшить практики контроля версий

Как команды используют контроль версий так же важно, как и используют ли они его. Мы обнаружили, что несколько ключевых практик помогают командам разблокировать большую ценность из своих систем контроля версий:

Версия все: Хранить все производственные артефакты в управлении версиями. Это включает в себя код приложения, конфигурации системы, сценарии тестирования, сценарии развертывания, файлы оркестровки контейнеров и файлы конфигурации облака. Цель состоит в том, чтобы иметь возможность воспроизводить среды тестирования и производства полностью автоматизированным способом, используя только информацию из управления версиями.

Содействовать развитию на базе магистральных систем:6 Поощрять стратегию разветвления, которая сводит к минимуму долгосрочные отрасли и способствует частой интеграции. Это помогает снизить риск частых или сложных конфликтов слияния.

Совершать небольшие, частые обязательства: частое совершение небольших, связанных с ними изменений, независимо от того, являются ли они сгенерированными человеком или ИИ, облегчает понимание прогресса и отслеживание истории функции. Эта практика снижает риск сложных конфликтов слияния и упрощает возвращение к известному хорошему состоянию, если изменение создает проблему.

Напишите четкие сообщения об обязательствах: сообщения об обязательствах рассказывают историю кодовой базы. Четко объясняя «почему» изменения, а не только «что», команды создают бесценную историческую запись. Принятие стандарта, такого как Conventional Commits7, может еще больше улучшить ясность и обеспечить автоматизацию на основе истории обязательств.

Сохраняйте подсказки ИИ: Срочные предложения содержат ценный контекст о задаче. Путем их версирования команды могут делиться знаниями, совершенствовать свою способность получать ценность от ИИ и создавать аудиторский след для агентных рабочих процессов.

Файлы конфигурации агентов: файлы конфигурации агентов, такие как GEMINI.md или CLAUDE.md, предоставляют контекст и инструкции для агентов ИИ. Эти файлы могут быть важным источником командных норм и практик и могут обеспечивать важные ограждения для помощников ИИ.

Внедрение ИИ в команду означает, что вам нужно переосмыслить свой учебник по управлению версиями. Это идеальное время, чтобы попасть на ту же страницу и установить некоторые новые основные правила. Первое, что вам нужно решить, это самое большое:

Основная философия: Во-первых, согласитесь как команда: мы позволяем ИИ совершать изменения напрямую или нет. Человек всегда должен проверять и выполнять код локально.

Как только у вас есть ответ на этот вопрос, другие вопросы начинают вставать на свои места:

Обязательная история: разработка с помощью ИИ может резко увеличить объем и скорость выполнения обязательств. Как команда будет управлять этой расширенной историей? Это хорошее время, чтобы определить и выровнять четкие, общекомандные правила использования перебазы, сквоша и слияния.

Обязательные сообщения: Что является стандартом команды для «хорошего» сообщения об обязательствах? Чтобы все изменения были понятными, согласуйте согласованный формат. Например, команды должны решить, хотят ли они включить подсказку при описании причины изменения.

Практика работы в небольших партиях является гарантией. Команды должны рассмотреть, как они будут гарантировать, что все взносы являются небольшими, логическими изменениями вместо одного гигантского, трудно пересматриваемого сброса кода.

Написание этих новых правил имеет решающее значение. Это дает вашей команде четкое руководство и даже создает базу знаний, которая затем может использоваться внутренними возможностями данных, доступными для ИИ.

Как улучшить практики контроля версий

По мере того, как команды разрабатывают свои методы контроля версий, часто возникают два существенных препятствия:

Ограниченное использование: Одной из наиболее распространенных ошибок является хранение только исходного кода приложения в управлении версиями.Команды должны иметь возможность воспроизводить среды тестирования и производства полностью автоматизированным способом с необходимой инфраструктурой и данными вместе со скриптами, исходным кодом и информацией о конфигурации из управления версиями.

Конфликты возникают, когда система контроля версий не может автоматически разрешать конкурирующие изменения в одних и тех же строках кода. Хотя иногда неизбежные, частые или сложные конфликты слияния часто являются признаком проблем процесса. Команды могут смягчить этот риск, гарантируя, что ветви разработки недолговечны и путем принятия основополагающих практик, таких как непрерывная интеграция8 и разработка на основе магистралей.9



Как оценить практики контроля версий



Вопросы опроса DORA

Исследование DORA измерило эту способность, задав эти вопросы:10

Для основного приложения или службы, над которой вы работаете, какой из следующих активов управляется в системе управления версиями?

○ Код приложения

• Конфигурации приложений

Код для автоматизации сборки и конфигурации

○ Потребности в системах ИИ

- Системные конфигурации

• Подумайте о последних изменениях, которые были внесены в основное приложение или услугу, над которой вы работаете, примерно сколько строк было изменено (добавлено, отредактировано или удалено) в рамках этого обязательства?

При активном изменении кода или конфигурации основного приложения или службы, над которой вы работаете, примерно как часто вы совершаете изменения в контроле версий?

При активном изменении кода или конфигурации для основного приложения или службы, над которой вы работаете, примерно насколько вы полагаетесь на возможность отмены, возврата или отката изменений?

Дополнительные меры

Дополнительные сигналы, которые могут помочь измерить методы контроля версий команды, включают:

Что проверить	Что измерить
Время восстанавливать окружающую среду	Практика восстановления среды с использованием только необходимой инфраструктуры, данных и системы управления версиями
Активные ветви в репозитории кода приложения	Измерьте, сколько активных ветвей у вас есть в системах управления версиями репозиториев приложений, и сделайте это число видимым для всех команд.
Частота слияния ветвей и вилок с стволом	Измерьте двоичное значение (да / нет) для каждой ветви, которая слита, или измерьте процент ветвей и вилок, которые сливаются каждый день.
Слияние конфликтных ситуаций	Отслеживайте количество или процент операций слияния (например, объединение ветви функций с основной), которые приводят к конфликту, требующему ручного разрешения.

Выводы

Управление версиями — это больше, чем просто инструмент для хранения кода; это основополагающая возможность, которая лежит в основе многих других передовых практик. Это необходимое условие для непрерывной доставки11 и для создания гибкой инфраструктуры12, которая позволяет организациям адаптироваться и процветать.

DORA уже давно заявляет о необходимости сильных практик контроля версий. Сегодня эта потребность сильнее, чем когда-либо. По мере интеграции ИИ во все аспекты разработки программного обеспечения код создается быстрее и менее предсказуемо, чем когда-либо прежде.

Этот динамизм открывает огромные творческие возможности, но также приносит новые формы риска. Контроль версий является важным механизмом безопасности, который облегчает эксперименты: в программном обеспечении, которое создают команды, а также в способах его создания.

По мере того, как развитие продвигается дальше в агентный мир, где агенты ИИ генерируют и применяют изменения автономно, эта дисциплина становится первостепенной. Каждое изменение, будь то человек, ИИ или сотрудничество между несколькими сторонами, должно быть отдельной атомной единицей с четким сообщением, создавая проверяемый след, который позволяет немедленно и точно откатываться.

Данная система безопасности наиболее эффективна в сочетании со спецификой работы небольшими партиями.

1.Git. https://git-scm.com

2. Apache® Subversion®. https://subversion.apache.org

3. Меркуриальный. https://www.mercurial-scm.org

4. Jujutsu — система контроля версий. https://github.com/jj-vcs/jj

5.2014 State of DevOps Report. https://dora.dev/research/2014

6. «Развитие на основе мусора». https://dora.dev/capabilities/trunk-based-development

7. «Обычные обязательства». https://www.conventionalcommits.org

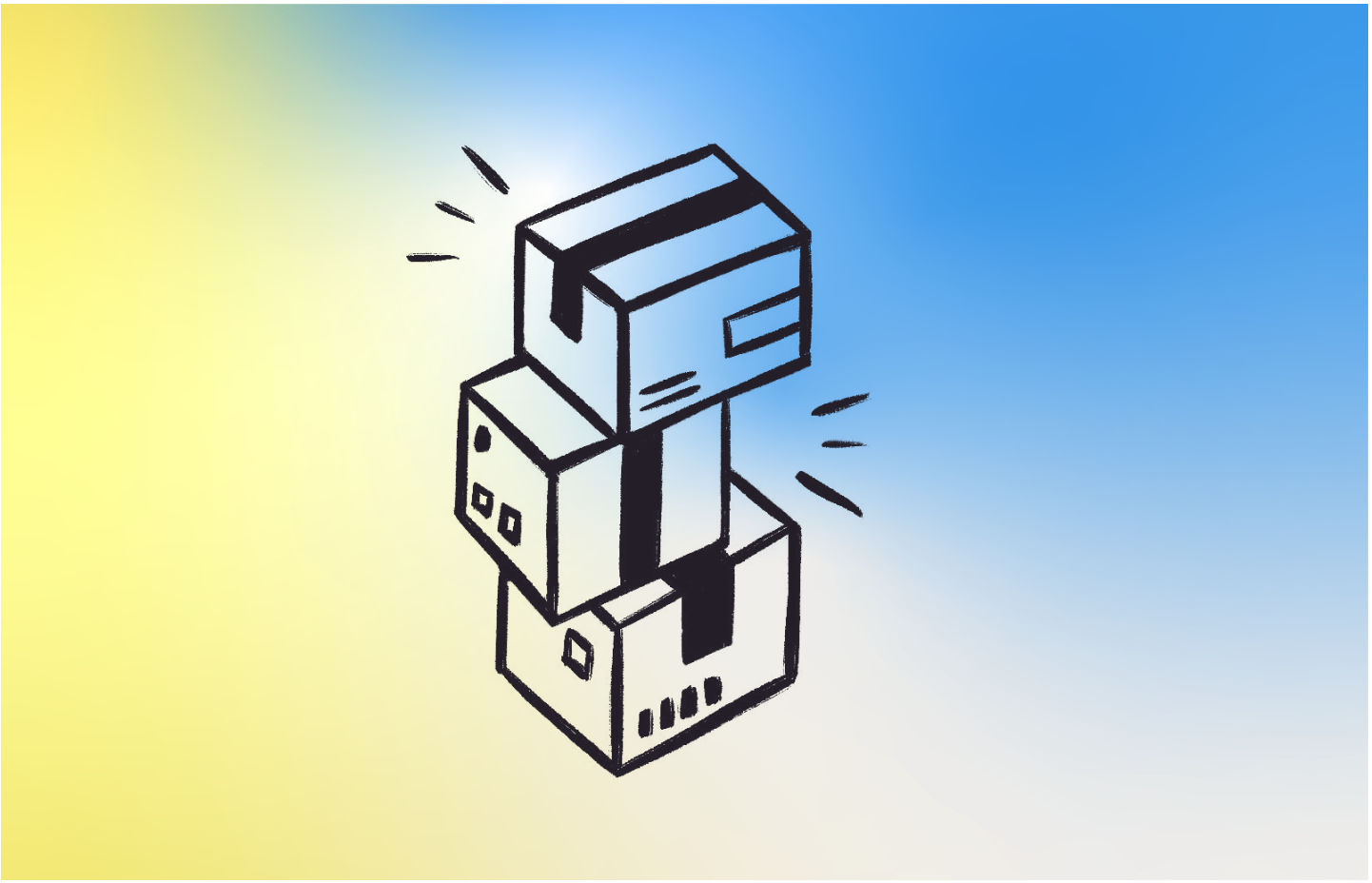
8 «Непрерывная интеграция». https://dora.dev/capabilities/continuous-integration

9. «Развитие на основе мусора». https://dora.dev/capabilities/trunk-based-development

10. «Опросы: контроль версий». https://dora.dev/ai/capabilities-model/questions/#strong-version-control-practices

11. "Непрерывная доставка". https://dora.dev/capabilities/continuous-delivery

12 «Гибкая инфраструктура». https://dora.dev/capabilities/flexible-infrastructure



Работа небольшими порциями

Роб Эдвардс

Доставка приложений Lead, Google Cloud

Работа в небольших партиях усиливает положительное влияние внедрения ИИ на производительность продукта.

Работа в небольших партиях уменьшается

- Трение
- Индивидуальная эффективность

Как небольшие порции изменений повышают результативность

Создание успешного приложения — это игра по улучшению обратной связи по единицам работы, которую выполняет команда.Команде необходимо знать, имеет ли изменение правильный синтаксис, интегрируется ли оно с остальной кодовой базой, соответствует ли оно операционным требованиям, достигает ли оно своей цели, и, в конечном итоге, довольны ли пользователи изменением.

Работа в небольших партиях - это степень, в которой команды разбивают свою работу на самые маленькие возможные куски, которые могут быть завершены и доставлены независимо. Каждая партия работы должна следовать принципу INVEST1, чтобы обеспечить ее высокое качество и ценность:

Независимый: он может быть разработан и развернут без зависимости от других частей.

Это не строгий контракт; детали могут быть уточнены с течением времени.

Ценность: это обеспечивает реальную ценность для пользователя или заинтересованных сторон.

Оценка: Вы можете получить приблизительное представление о том, сколько времени потребуется для строительства.

Маленький: он может быть завершен в короткие сроки, в идеале часы, а не недели.

Тестируемый: вы можете убедиться, что он работает так, как ожидалось.

Возможность работать небольшими партиями является краеугольным камнем как бережливого управления продуктами, так и непрерывной доставки.3 Это меняет экономику процесса разработки программного обеспечения, делая его дешевым и легким для получения изменений в двери и в руки пользователей.

Работа в больших, нечастых партиях является рискованной, медленной и демотивирующей. Она представляет значительный риск для процесса развертывания, затрудняет восстановление после неудачи и создает длительные, болезненные фазы интеграции, которые убивают производительность.

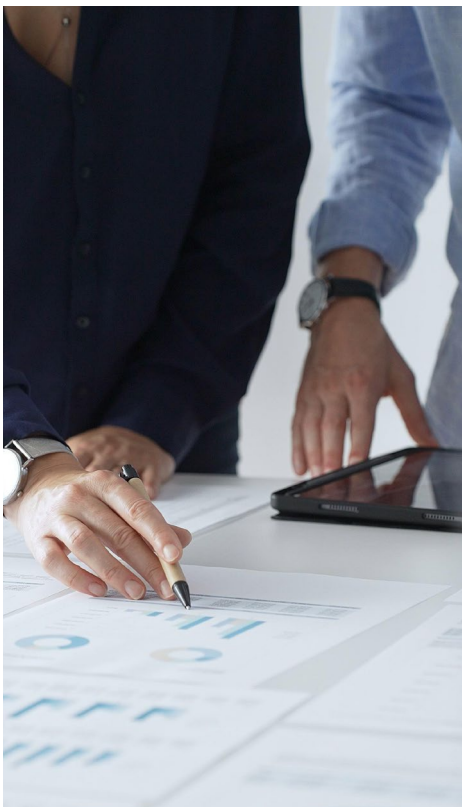
Работа в небольших партиях является центральным принципом непрерывной интеграции4 и непрерывной доставки5, и это значительно улучшает производительность и благополучие за счет:

Создавая более быстрые петли обратной связи, когда вы выпускаете небольшое изменение, вы почти сразу же узнаете, работает ли оно. Это облегчает сортировку проблем, исправление ошибок и обучение на основе ваших решений.

Качество строительства в практике помогает встроить качество в процесс и минимизирует переделку, связанную с исправлением ошибок, обнаруженных в конце цикла. Это усиливается в сочетании с сильной автоматизацией испытаний.

Повышение эффективности и мотивации Команды видят, что их работа быстро идет, что является огромным мотиватором. Это также мешает организации поддаться заблуждению о потопленных издержках.

Практика позволяет быстро высвобождать выпуск минимально жизнеспособных продуктов и часто и часто собирать отзывы пользователей с помощью таких методов, как A / B-тестирование.



Зачем вносить изменения небольшими порциями при работе с ИИ

Работа в небольших партиях является критически важной контрмерой рискам разработки с помощью ИИ. Хотя ИИ может быстро генерировать большие объемы кода, большие изменения трудно просматривать, тестировать и безопасно интегрировать.

Мелкосерийная работа может помочь предотвратить ускоренное развитие ИИ, приводящее к повышенной нестабильности. Она вынуждает к дисциплинированному подходу, при котором фокус смещается от генерации исходного кода к продуманному разложению, подсказке и проверке.

Исследования DORA показывают, что в сочетании с внедрением ИИ, работая небольшими партиями:

Улучшает производительность продукта, оказывает четкое, положительное влияние на производительность продукта.

Это превращает нейтральный эффект ИИ на организационные трения в чистый позитив, помогая командам преодолевать хаос вниз по течению и избегать разрушительных развертываний.

Дисциплина небольших партий ставит естественный предел этому, что немного снижает воспринимаемые индивидуальные выгоды от эффективности ИИ, но направляет эту производительность к более стабильному и устойчивому процессу доставки.

Размер партии смягчает влияние ИИ на производительность продукта

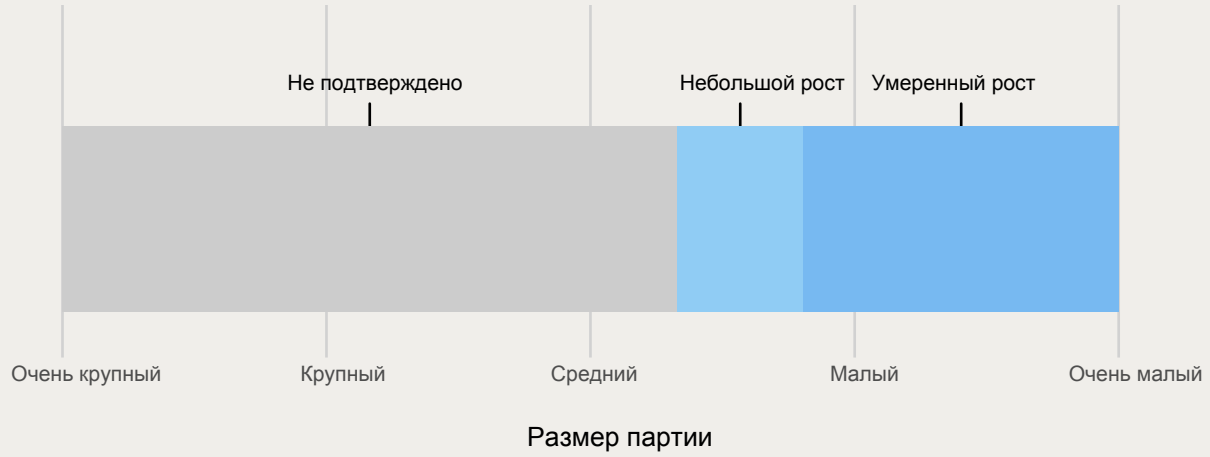


Рисунок 13: Размер партии смягчает влияние ИИ на производительность продукта

Размер партии смягчает влияние ИИ на трение

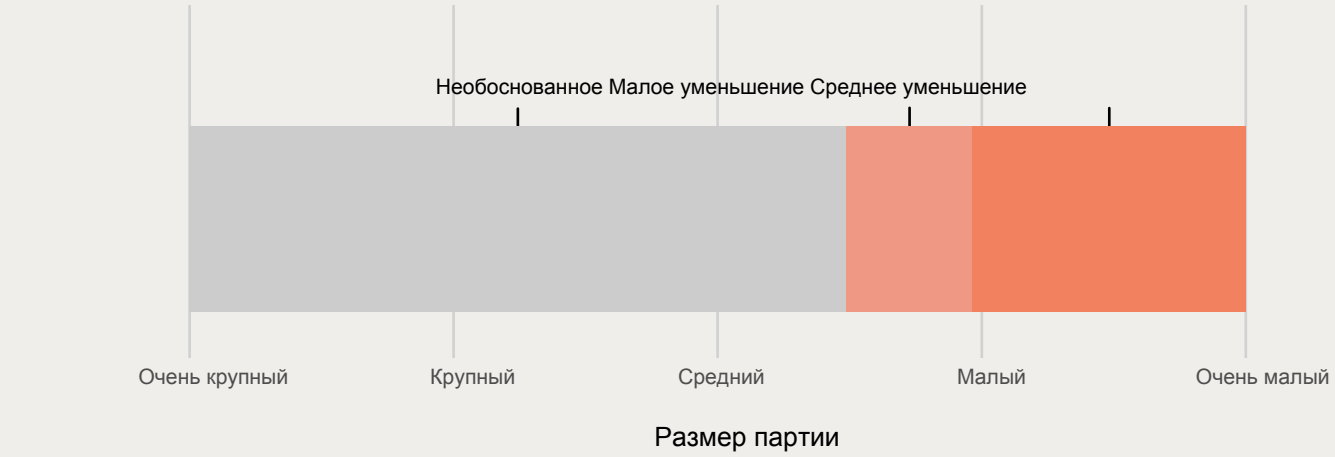


Рисунок 14: Размер партии смягчает влияние ИИ на трение

Размер партии смягчает влияние ИИ на индивидуальную эффективность

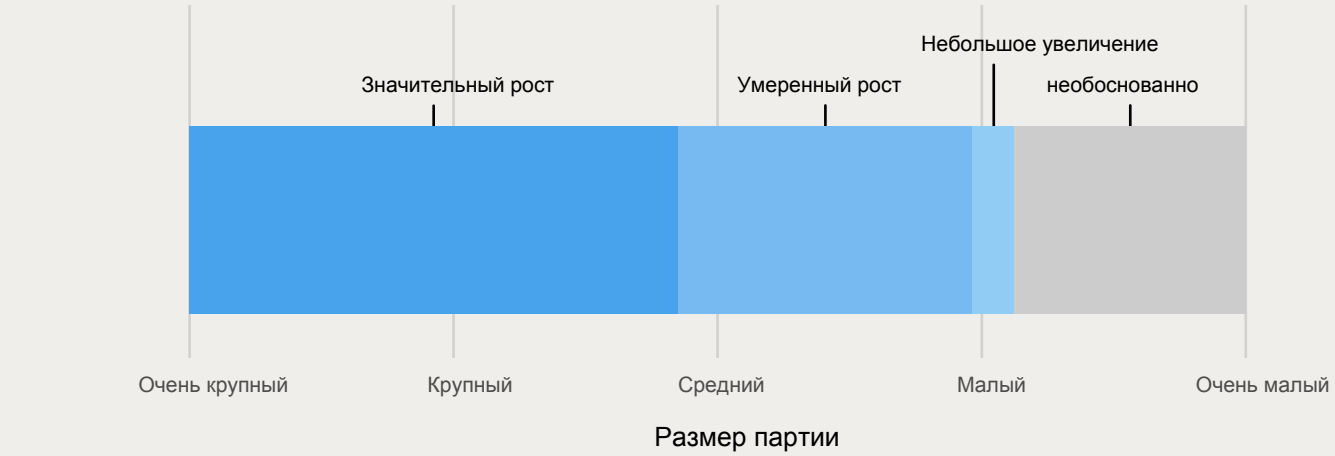


Рисунок 15: Размер партии смягчает влияние ИИ на индивидуальную эффективность

Наше исследование выявило любопытный вывод: хотя ИИ оказывает положительное влияние на индивидуальную эффективность, наши данные свидетельствуют о том, что эти преимущества немного уменьшаются в командах, которые работают небольшими партиями.

Накладная гипотеза: ИИ-помощники могут быть очень эффективными при создании больших блоков кода для четко определенной задачи. Человеческие усилия, необходимые для разложения более крупной проблемы на множество мелких, дискретных и

Хорошо сформированные подсказки могут компенсировать некоторую скорость генерации исходного кода. Работа разработчика переходит от написания кода к разложению, подсказке и проверке.

Гипотеза трения обзора: когнитивная нагрузка для человека, чтобы рассмотреть и понять небольшой кусок незнакомого, машинного кода, может быть выше на строку, чем просмотр написанного человеком кода от доверенного коллеги.

Общий процесс, отрицающий некоторые выгоды от быстрой генерации.

Гипотеза несоответствия Tooling: сегодняшние инструменты разработки ИИ могут быть лучше оптимизированы для создания более крупных, более полных функций или файлов. Рабочий процесс использования этих инструментов для создания многих небольших, дискретных запросов на вытягивание может быть громоздким, добавляя слой ручных накладных расходов, который подрывает рост производительности.

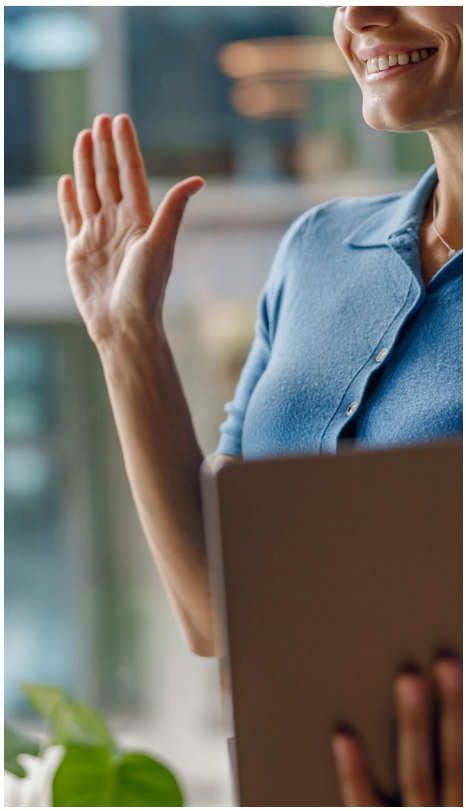
Что еще более важно, мы утверждаем, что индивидуальная эффективность не обязательно должна преследоваться как цель сама по себе. Скорее, индивидуальная эффективность является средством для достижения большей организационной, командной и производительности продукта и улучшения благосостояния разработчиков.

Хотя этот необходимый переход от генерации исходного кода к продуманному разложению и проверке может показаться потерей индивидуальной скорости, именно эта дисциплина может разблокировать устойчивую производительность на уровне команды и помочь предотвратить хаос вниз по течению.

В этом случае работа в небольших партиях увеличивает заявленную производительность продукта, а также уменьшает воспринимаемые трения для команд, работающих с ИИ. Мы считаем, что эти преимущества перевешивают любой потенциальный вред для индивидуальной эффективности от работы в небольших партиях - в дополнение к тем преимуществам работы в небольших партиях, которые были давно доказаны как часть нашей базовой модели DORA 6.

Эти преимущества значительны, но они полагаются на освоение основ. Независимо от того, использует ли ваша команда инструменты с помощью ИИ или нет, основные методы сокращения размера партии остаются прежними.

Как уменьшить размер порций изменений



Цель состоит в том, чтобы рабочие блоки были завершены в более короткие сроки, в идеале продолжительностью всего от нескольких часов до нескольких дней. Любая партия кода, которая занимает больше недели, слишком велика.

Уточните навыки срезки историй Поезд владельцев продуктов и команд о том, как разбить большие истории пользователей на более мелкие, вертикальные срезы стоимости.

Обязательство по ежедневному хоботу Разработчики должны совершать и объединять свою работу в магистраль (транк) не реже одного раза в день. Это необходимое условие как для непрерывной интеграции7, так и для разработки на основе хобота.8

Использование флагов функций (или других прогрессивных методов доставки) для отделения развертывания от выпуска. Это критическая техническая практика, которая позволяет объединять неполные функции в основную ветвь и развертывать их в производство, не будучи видимыми для пользователей.

Установите ограничения на работу в процессе (WIP) Используйте доски в стиле Kanban с явными ограничениями WIP, чтобы препятствовать многозадачности и поощрять завершение работы.

Это делает узкие места видимыми и заставляет команду сосредоточиться на том, чтобы «сделать» работу.

Пример: нарезка функции «сохранить на потом»

Менеджер продукта просит добавить кнопку «сохранить для более поздних» к элементам на странице продукта. Подход большой партии будет заключаться в том, чтобы построить всю функцию на ветке и выпустить ее, когда все будет сделано. Подход небольшой партии выглядит следующим образом:

Бэтч 1 (Back-end): Создание новой конечной точки API (POST/api/ v1/items/{id}/save) и изменение базы данных для хранения сохраненного элемента.

Но ни один клиент пока не звонит.

Batch 2 (Back-end): Создание конечных точек GET и DELETE для управления сохраненными элементами.

Батч 3 (Front-end): Добавьте кнопку «сохранить для последующего использования» в пользовательский интерфейс, но спрячьте ее за флагом функции. Обработчик кнопки onClick вызывает новый API. Он развернут, но функция отключена для всех пользователей.

4-я серия (тестирование): команда включает флаг функции в

Испытательная среда для проверки сквозного потока.

Бэтч 5 (выпуск): Флаг функции позволяет небольшому проценту пользователей проверить его производительность и полезность перед полным развертыванием.

Каждая партия представляет собой небольшую, независимую и развертываемую единицу работы, которая может быть завершена за день или меньше.

Что мешает вносить небольшие изменения

Поскольку команды работают над уменьшением размеров партий, они должны избегать ловушки локальной оптимизации. Распространенной ловушкой для команды разработчиков является оптимизация для быстрого завершения собственной работы и передачи ее следующей команде в процессе, например, обеспечение качества (QA) или операции. Это может привести к накоплению работы, создавая узкое место, которое замедляет всю систему.

Лучшим подходом является полное представление о системе и определение приоритетности потока небольших партий вплоть до завершения. Это часто включает в себя реализацию ограничений WIP9 и видимость работы и узких мест10, которые могут помочь команде в решении системных проблем.

Как говорится, «Прекратите начинать, начните заканчивать». Это способствует выполнению небольшого количества задач, а не запуску многих и одновременному жонглированию ими.

Препятствие: «Эта особенность слишком велика, чтобы ее можно было разбить» Это распространенное чувство, но это редко бывает правдой. Ключ заключается в том, чтобы перейти от мышления о горизонтальных слоях (например, «создать весь пользовательский интерфейс, а затем весь бэкэнд») к вертикальным срезам. Начните с наименьшего возможного фрагмента сквозного значения, даже если он сначала не обращен к пользователю, как конечная точка API. Техника «отделение по абстракции»11 является мощным шаблоном для постепенного создания крупномасштабных рефакторов.

Препятствие: «Мой менеджер по продуктам хочет увидеть всю функцию, прежде чем мы ее выпустим» Это проблема связи и доверия. Одно из решений заключается в использовании флагов функций. Объясните, что код выпускается в производство непрерывно, что снижает риск, но функция останется невидимой для пользователей, пока она не будет завершена и готова к обзору. Это отделяет техническое развертывание от бизнес-релиза, давая управлению продуктом полный контроль над тем, когда функция будет запущена.

Как измерять размер порций изменений



Вопросы опроса DORA

Исследование DORA измерило эту способность, задав следующие вопросы:12

Для основного приложения или услуги, над которой вы работаете, примерно сколько времени требуется разработчику для выполнения работы, назначенной в одной задаче (например, карточка, билет или история)?

Для основного приложения или службы, над которой вы работаете, примерно сколько изменений (например, запросы на вытягивание, запросы на слияние или списки изменений) объединены в один выпуск или развертывание?

• Подумайте о последних изменениях, которые были внесены в основное приложение или услугу, над которой вы работаете, примерно сколько строк было изменено (добавлено, отредактировано или удалено) в рамках этого обязательства?

Вопросы для групповой оценки

Обсуждение размера партии с вашей командой может выявить возможности для улучшения вашего рабочего процесса и архитектуры.

Как мы можем разбить нашу работу на более мелкие части, которые могут быть выпущены независимо?
• Каков самый большой источник задержки в нашем трубопроводе развертывания?

Мы устраняем эти препятствия или просто ослабляем ограничения?

Когда в производстве обнаруживается ошибка, сколько отдельных изменений мы обычно должны исследовать, чтобы найти причину?

Дополнительные меры

Дополнительные сигналы, которые могут помочь измерить работу небольшими партиями, включают:

Что проверить	Что измерить
Ведущее время для изменений	Время, необходимое для успешного внедрения кода или изменения в производство
Частота слияния ветвей и вилок с стволом	Измерьте двоичное значение (да / нет) для каждой ветви, которая слита, или измерьте процент ветвей и вилок, которые сливаются каждый день.
Выпуски разъединения	Измерить количество или процент изменений, развернутых в производстве, но не сразу доступных для всех пользователей.

Выводы

Работа в небольших партиях является важной дисциплиной, которая направляет генеративную скорость ИИ в устойчивую производительность, заставляя большие, генерируемые ИИ изменения в небольшие, поддающиеся обзору, и тестируемых единиц, мелкосерийная работа выступает в качестве критического усилителя. Это переводит потенциальное повышение индивидуальной эффективности в реальную производительность продукта и снижение трения. Когда мелкосерийная работа сочетается с управлением версиями, команды могут двигаться быстро и безопасно.Следующим шагом является обеспечение их движения в правильном направлении, что требует сильной ориентированности на пользователя.

1. «Инвестировать (мнемонично)». https://en.wikipedia.org/wiki/INVEST_(mnemonic)
2.2017 State of DevOps Report. https://dora.dev/dora-report-2017
3. «Непрерывная доставка». https://dora.dev/capabilities/continuous-delivery
4 «Непрерывная интеграция». https://dora.dev/capabilities/continuous-integration
5. «Непрерывная доставка». https://dora.dev/capabilities/continuous-delivery
6. «Исследовательская программа DORA». https://dora.dev/research
7 «Непрерывная интеграция». https://dora.dev/capabilities/continuous-integration
8. «Развитие на основе мусора». https://dora.dev/capabilities/trunk-based-development
9. «Работа в рамках процесса». https://dora.dev/capabilities/wip-limits
10. «Видимость работы в потоке ценностей». https://dora.dev/capabilities/work-visibility-in-value-stream
11. «Развитие на основе мусора: отделение посредством абстракции». https://trunkbaseddevelopment.com/branch-by-abstraction
12. "Опросы: Работа небольшими партиями". https://dora.dev/ai/capabilities-model/questions/#working-in-small-batches



Ориентация на потребности пользователей

Дэйв Стэнк

Специалист по работе с клиентами Google Cloud

Аманда Льюис

Инженер по связям с разработчиками, Google Cloud

Ориентированный на пользователя фокус смягчает влияние внедрения ИИ на производительность команды.

Низкий уровень ориентированности на пользователя приводит к снижению производительности команды

Высокий уровень ориентированности на пользователя приводит к повышению производительности команды

Все программное обеспечение предназначено для пользователей-людей. Истинная ценность любой технологии, от хорошо видимого интерфейса интерфейса до невидимого процесса бэкэнда, определяется его полезностью для человека в конце цепочки создания стоимости. Однако сложные технологические стеки и сложные поддерживающие технологии, используемые для их создания и эксплуатации, могут скрыть существенную связь между актом создания программного обеспечения и выгодой, которую оно в конечном итоге обеспечивает.

Слишком легко зацикливаться на внутренних механизмах разработки программного обеспечения — обсуждении вариантов рамок, оптимизации для угловых случаев, борьбе с организационной бюрократией — и упускать из виду всеобъемлющую цель этого программного обеспечения.

В то время как исследования DORA продемонстрировали, что внимание к промежуточным результатам (таким как показатели доставки программного обеспечения) в жизненном цикле разработки программного обеспечения имеет важное значение для общей производительности, команды также должны постоянно согласовывать свои приоритеты в обслуживании конечного пользователя.

В дополнение к улучшению качества продукта1 и прогнозированию 40% улучшения организационной производительности2 ориентированная на пользователя направленность также может улучшить качество жизни разработчиков, повышая удовлетворенность работой и производительность при одновременном снижении выгорания.



Зачем учитывать потребности пользователей при работе с ИИ

До появления автономных систем кодирования человек (разработчик) был бы тесно вовлечен в создание каждой строки кода, опираясь на свою человеческую интуицию и опыт. Теперь большая часть этого кода генерируется ИИ. Поэтому разработчик должен быть прокси-сервером для пользователя — постоянно переориентировать инструменты ИИ на ориентированные на пользователя цели. Инженеры должны стремиться улучшить свое понимание потребностей пользователей и расставить приоритеты для тщательного описания этих потребностей в контексте, предоставляемом моделям ИИ.

Исследование DORA 2025 года показало, что по мере того, как команды с сильной ориентацией на пользователя внедряют инструменты ИИ, их эффективность растет. Но и обратное: команда с плохой ориентацией на пользователя, которая принимает ИИ, вероятно, увидит снижение производительности своей команды. Если приоритеты команды не соответствуют потребностям пользователей, усиление с ИИ может продвинуть их еще дальше в неправильном направлении.

Ориентированный на пользователя фокус смягчает влияние ИИ на производительность команды

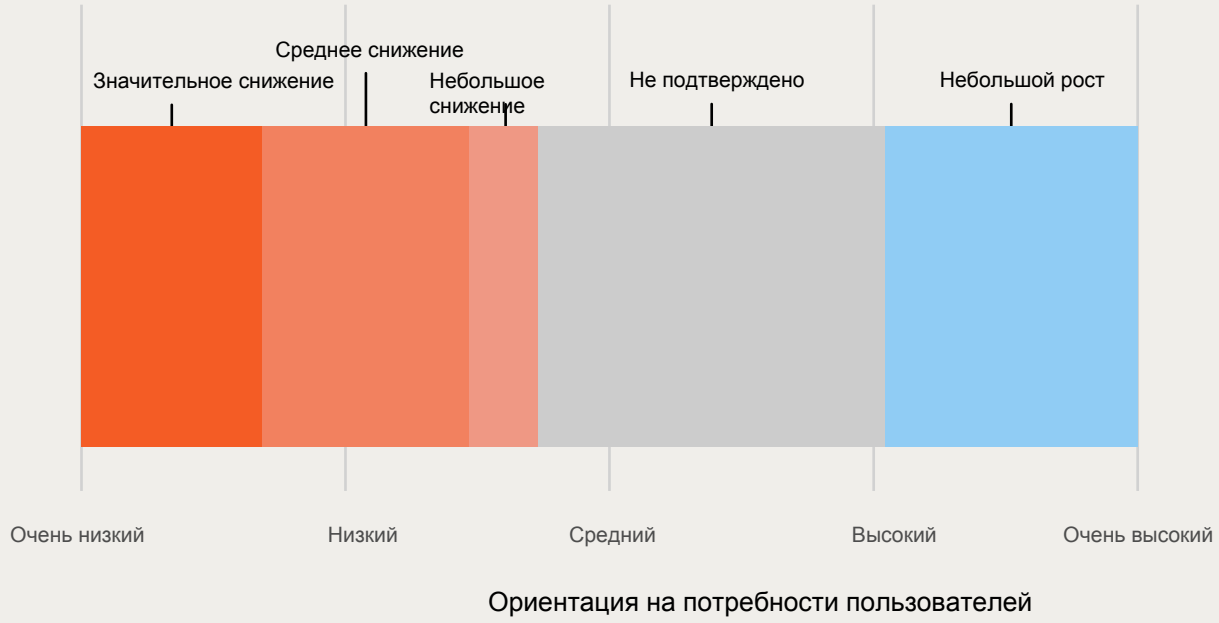


Рисунок 16: Ориентация на пользователя смягчает влияние ИИ на производительность команды

Как лучше учитывать потребности пользователей

Поскольку модели ИИ интегрированы в наборы инструментов команд, эти модели должны быть обеспечены точным, текущим, детализированным контекстом конечного пользователя, таким как поведение, болевые точки и желаемые результаты. Необходим культурный сдвиг, который делает истории пользователей, обратную связь и аналитику необходимыми. Некоторые методы для разработки ориентированного на пользователя фокуса команды включают:

Интеграция циклов обратной связи пользователей

Активно запрашивайте и используйте прямую обратную связь с пользователем. Создавайте каналы для обратной связи с пользователем, такие как интеграция в приложение, опросы по одному вопросу сразу после того, как пользователь завершит критическую задачу. Дополняйте количественные данные качественными выводами из интервью с пользователем и структурированными программами бета-тестирования. Установите непрерывный цикл с низкой задержкой: обратная связь поступает, она немедленно доступна для команды, а идеи используются для уточнения подсказок ИИ и разработки функций.

Сделайте пользовательские метрики видимыми

Если панели приборов показывают только такие показатели, как скорость и частота развертывания, пользователь легко забывается.

Чтобы перейти к ориентированному на пользователя фокусу, отображайте пользовательский опыт и инженерные метрики на видном месте на встречах в команде и непосредственно в инструментах разработчика. Сосредоточьтесь на показателях успеха и удовлетворенности, например, измерении удовлетворенности клиентов (CSAT), показателях выполнения задач для функций или времени выполнения задач для критических поездок пользователей. Рассмотрим показатели успеха продукта, такие как в рамках H.E.A.R.T..3

Делая «почему» и «для кого» работы видимыми наряду с техническими «что» и «как», команды и инструменты искусственного интеллекта, которые они используют, основаны на влиянии, которое они оказывают на пользователя.

Вовлекайте инженерию непосредственно в пользовательские исследования

В то время как менеджеры по продуктам, исследователи пользовательского опыта и другие члены команды играют жизненно важную роль в синтезе и выявлении потребностей пользователей, важно, чтобы эти роли служили для облегчения соединения, а не для обеспечения расстояния между инженерами и пользователями. Пригласите разработчиков наблюдать за сессиями тестирования пользователей и исследованиями юзабилити напрямую. Результаты дистиллированных исследований пользователей могут убрать нюансы. Видя, что пользователь из первых рук пытается выполнить задачу, создает устойчивую связь с проблемой. Прямое воздействие создает более сильное сочувствие пользователей и обеспечивает лучшее понимание болевых точек пользователя.

Этот опыт из первых рук информирует о том, как они проектируют решения, пишут тесты и взаимодействуют с ИИ, обеспечивая его удовлетворение потребностей пользователя.

Рассмотрим развитие, основанное на спецификациях

Новая парадигма развития направлена на создание структуры, вокруг которой крупные языковые модели (LLM) могут быть ориентированы на потребности пользователей: разработка на основе спецификаций (SDD).4 SDD разбивает кодирование на фазы, генерируя выходы, доступные для версий. На начальном этапе разработчики, возможно, с ИИ, уточняют потребности пользователей и ограничения в документации, «источнике истины».

Только после блокировки в спецификациях (которые, необязательно, могут быть подвергнуты экспертной оценке), ИИ поручено писать фактический код. Повышая спецификации до позиции первенства, SDD побуждает разработчиков активно утверждать потребности пользователей как ограничения. Это обещает снизить вероятность того, что код, генерируемый ИИ, будет смещен в сторону пользовательской ценности.

Для глубокого погружения в разработку, основанную на спецификациях, см. презентацию Бетсалеля (Саула) Уильямсона сообществу DORA: разблокируйте потенциал LLM с помощью разработки на основе спектра.5

Общие препятствия для ориентированного на пользователя фокуса

По мере того, как команды внедряют ИИ, несколько общих препятствий могут ослабить необходимую связь с конечным пользователем, подрывая ценность работы, ускоряемой ИИ:

Функциональная фабрика6 мышления Команды фокусируются на измерении выходных данных (поставляемых характеристик, скорости), а не на результатах (ценности пользователя).

ИИ может усилить эту дисфункцию, что приводит к тому, что команды быстро создают функции, которые не решают реальные проблемы пользователей, что приводит к высокой активности, но низкому воздействию.

Технологически ориентированный подход Команды попадают в ловушку «решительности» или «разработки, основанной на резюме», внедряя новые технологии (включая модели ИИ) ради себя, а не для решения конкретной проблемы пользователя.

Организационные и процедурные бункеры разработчики часто систематически отключаются от своих конечных пользователей политиками, процессами или структурами команды. Это создает модель «привратника», где пользовательская обратная связь фильтруется, лишая разработчиков глубокого контекста и сочувствия, необходимых для создания ценных решений.

Как оценить ориентацию на потребности пользователей



Вопросы опроса DORA

Конечная мера ориентированного на пользователя внимания находится в глазах пользователя, что отражается в таких показателях продукта, как принятие, удержание или удовлетворенность клиентов.

Команда со здоровой культурой, внимательно относящаяся к видимым показателям пользователей, будет определять приоритеты этого конечного результата. Помимо таких внешних мер, вы также можете оценить отношение и практику команды, согласно той же рубрике, используемой в опросе DORA 2025 года.

Исследование DORA измерило эту способность на основе ответов на эти вопросы:7

Создание ценности для наших пользователей - это наша цель.

Опыт наших пользователей является нашим главным приоритетом.
Мы считаем, что сосредоточение внимания на пользователе является ключом к успеху бизнеса.
У нас есть четкое понимание того, что наши пользователи хотят достичь с помощью нашего приложения или сервиса.
• Мы используем обратную связь с пользователем для постоянного просмотра и переориентации функций.

Дополнительные меры

Дополнительные сигналы, которые могут помочь измерить ориентированность команды на пользователя, включают:

Что проверить	Что измерить
Успех продукта	Метрики продуктов, такие как: • Уровень усыновления • Показатель удержания • удовлетворенность клиентов
Выравнивание команды	Включение целей пользователя или пользователя в такие задачи, как этапы проекта
Представление пользователей в артефактах	Спецификации дизайна, которые включают пользователя
Ориентация пользователя на ad-hoc дизайн	Как часто пользователь изображается на доске или чертежах дизайна
Ценности команды и признание	Ориентированные на пользователя достижения, которые получают самые громкие похвалы от сверстников и руководства

Выводы

Инструменты генного ИИ ценятся за их способность обрабатывать огромное количество разнородных данных и быстро производить единый когерентный выход. Они делают это, делая прогнозы на основе контекста.

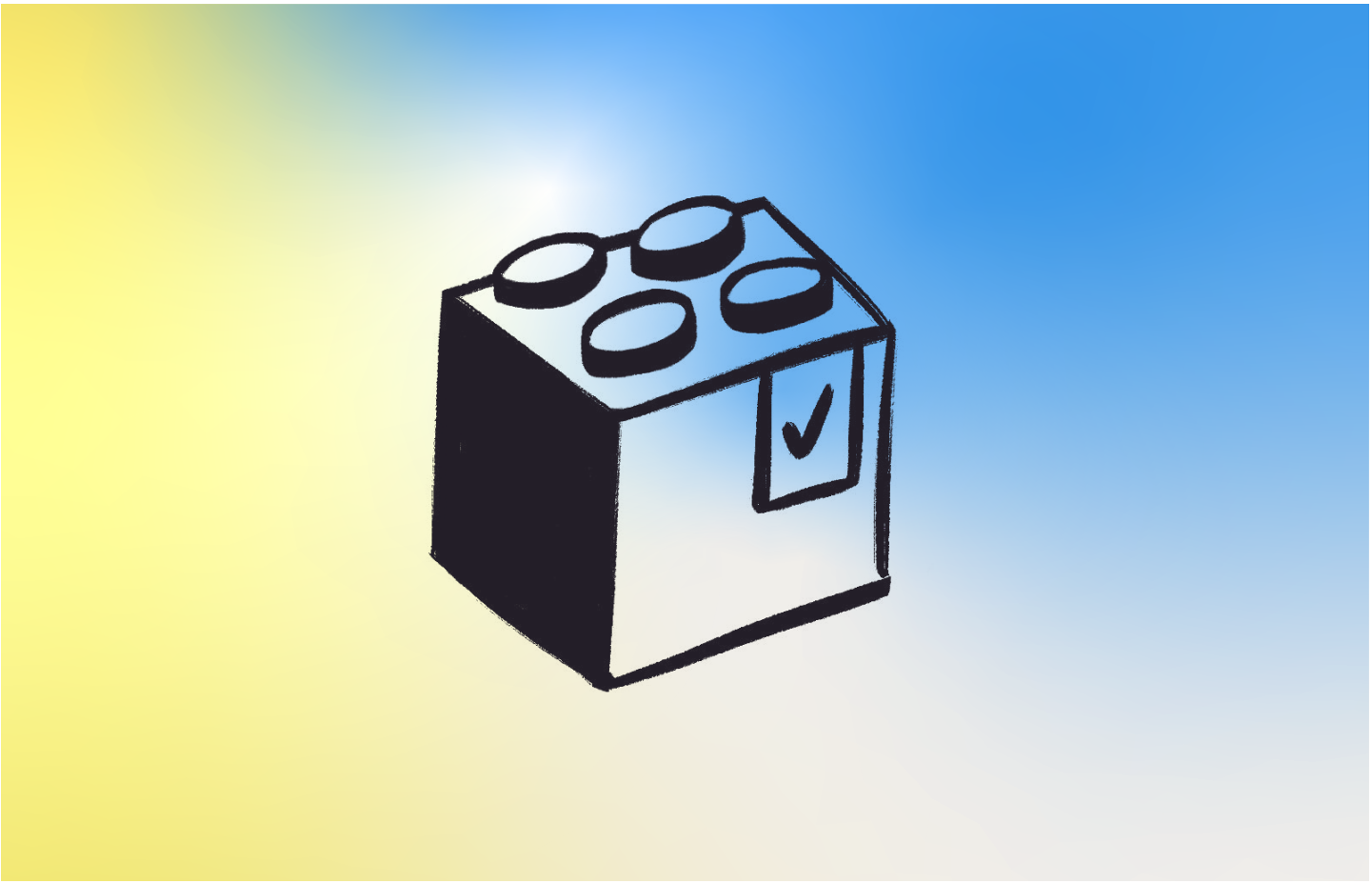
Разработчику поручено формировать этот контекст, направляя инструменты ИИ к предпочтительным результатам.

Мощные инструменты, такие как ИИ, и внимание к таким практикам, как непрерывная доставка8, позволяют командам быстро производить высококачественное программное обеспечение.

Сосредоточенность сделает программное обеспечение достойным.

Как только команды выровняются в этом ориентированном на пользователя направлении, им нужен масштабируемый, безопасный и автоматизированный способ доставки этой ценности, которая является ролью качественной внутренней платформы.

1.2024 Ускорить состояние DevOps. 59-60. https://dora.dev/dora-report-2024
2. Ускорить состояние DevOps Report 2023. 17. https://dora.dev/dora-report-2023
3. «Разблокирование успеха продукта путем объединения DORA и H.E.A.R.T.» https://cloud.google.com/transform/unlocking-product-success-by-combining-dora-and-heart
4. «Понимание разработки, основанной на спецификациях: Kiro, spec-kit и Tessl». https://martinfowler.com/articles/exploring-gen-ai/sdd-3-tools.html
5. «DORA Community Demo & Discussion: Unlock LLM potential with spec-driven development». https://www.youtube.com/watch?v=9Goq80lgxSY
6.12 признаков того, что вы работаете на фабрике. https://medium.com/@johnpcutler/12-signs-youre-working-in-a-feature-factory-44a5b938d6a2
7. "Опросы: Ориентированность пользователей". https://dora.dev/ai/capabilities-model/questions/#user-centric-focus
8. "Непрерывная доставка". https://dora.dev/capabilities/continuous-delivery



Качественные внутренние платформы

Эрик Максвелл

10-кратный технологический лидер, Google Cloud

Качественные внутренние платформы усиливают положительное влияние внедрения ИИ на организационную эффективность.

Центральная тема нашего исследования 2025 года заключается в том, что ИИ является усилителем. Он увеличивает существующие сильные стороны вашей организации, а также ее дисфункции. В то время как многие организации стремятся оснастить отдельных разработчиков инструментами ИИ, наши данные показывают решающий пробел: эти индивидуальные повышения производительности часто теряются из-за беспорядка вниз по течению, проглоченного узкими местами в тестировании, обзорах безопасности и сложных процессах развертывания.

Наше исследование показало, что 90% организаций приняли внутренние платформы, а 76% теперь имеют специализированные команды платформ для управления ими, что делает платформы важной основой для современной доставки программного обеспечения.

Высококачественная внутренняя платформа обеспечивает автоматизированные, стандартизированные и безопасные пути, необходимые для превращения потенциала ИИ в системные, организационные улучшения. Это основополагающая способность, которая позволяет использовать скорость ИИ, не жертвуя стабильностью, усиливая ее положительное влияние на организационную производительность.

Признаки качественной внутренней платформы

Качественная внутренняя платформа - это набор общих высококачественных инструментов, услуг, документации и «мощеных дорог» (или «золотых путей»), которые позволяют командам разработчиков безопасно, надежно и с соблюдением требований внедрять свои приложения.

Эта платформа лучше всего понимается как внутренний продукт, разработанный для ваших разработчиков. Она абстрагирует основную сложность, позволяя командам сосредоточиться на предоставлении пользовательской ценности, а не на навигации по инфраструктуре, безопасности и эксплуатационным препятствиям.

Наше исследование показывает, что успех платформы заключается не только в ее технических особенностях, но и в целостном опыте разработчиков.



Зачем нужны внутренние платформы при внедрении ИИ

Внутренняя платформа выступает в качестве основного уровня распределения и управления для ИИ.

Наше исследование 2025 года показало, что положительное влияние внедрения ИИ на организационную эффективность зависит от качества внутренней платформы.

Когда качество платформы низкое, влияние внедрения ИИ на организационную производительность незначительно, но когда качество платформы высокое, этот эффект становится сильным и позитивным.

Высококачественная платформа обеспечивает стандартизированные, автоматизированные пути, необходимые для управления увеличенным объемом и скоростью разработки с помощью ИИ. Она гарантирует, что код, генерируемый ИИ, последовательно тестируется, защищен и развертывается, превращая индивидуальные повышения производительности в системные, организационные улучшения. Она также служит критическим фактором снижения риска, создавая безопасное пространство для экспериментов, делая отказ дешевым и быстрое восстановление.



Внутренние платформы смягчают влияние ИИ на производительность организации

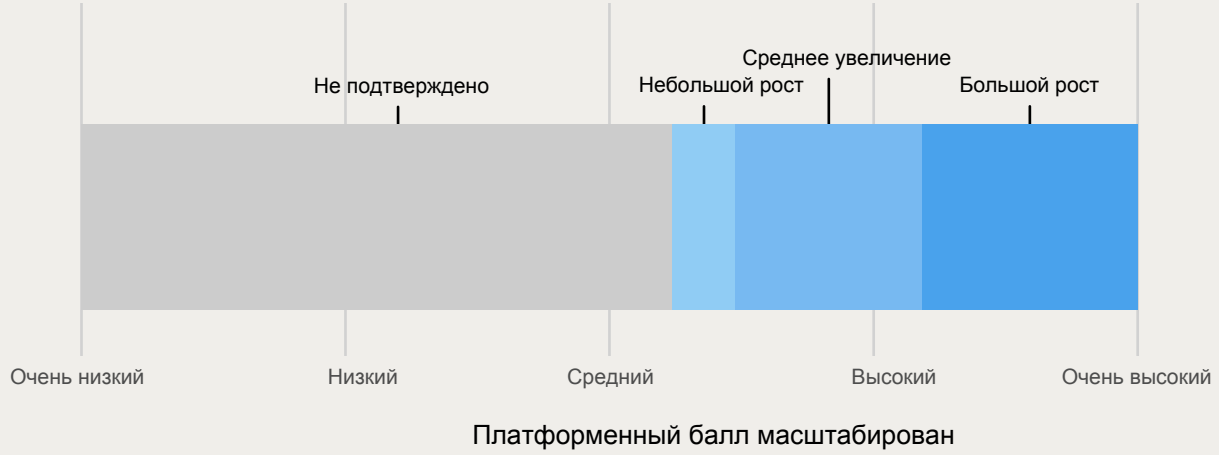


Рисунок 17: Внутренние платформы смягчают влияние ИИ на производительность организации

Как улучшить внутреннюю платформу

Наше исследование 2025 года показывает, что ценность платформы связана с ее общим целостным опытом, а не только с ее индивидуальными особенностями. Разработчики воспринимают платформу как единое целое, и весь их путь отражает их восприятие ее успеха.

Улучшение вашей платформы означает выход за рамки просто технологий и принятие ориентированного на продукт и пользователя подхода.

Принять мышление управления продуктом

Платформа - это внутренний продукт, а ваши разработчики - ваши клиенты. Для этого требуется назначить менеджера по продуктам, чтобы сосредоточиться на DevEx, глубоко понять потребности разработчиков и создать стратегическую дорожную карту. Эта работа должна быть обусловлена отображением критических поездок пользователей (например, «восстановление новой услуги» или «отладка производственной проблемы») для выявления и устранения наиболее значительных точек трения.

Проактивно «сдвиньте вниз» когнитивную нагрузку

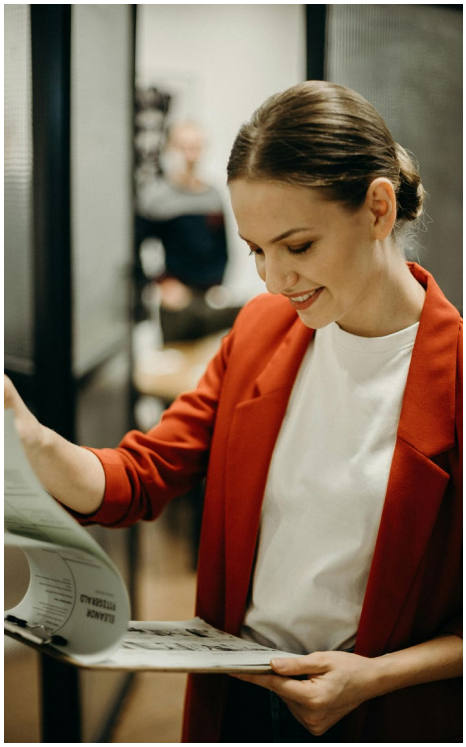
Основная цель платформы - уменьшить когнитивную нагрузку на разработчиков, абстрагируя основную сложность. Вместо того, чтобы заставлять разработчиков становиться экспертами в Kubernetes, облачных сетях и политике безопасности, вы «сдвинете» эту сложность в платформу. Предоставляя простые рабочие процессы самообслуживания и золотые пути, платформа позволяет разработчикам сосредоточиться на том, что они делают лучше всего: создание функций и предоставление ценности пользователям. Эта концепция является ключевым принципом успешной стратегии разработки платформы (подробнее см. «Как Google делает это: ваше руководство по разработке платформы» 1 и решение Google Cloud для разработки платформы »).

Начните с «минимально жизнеспособной платформы».

Вместо того, чтобы пытаться построить комплексную платформу сразу, сосредоточьтесь на решении одной задачи с высокой ценностью для определенного набора пользователей. Определите золотой путь для наиболее распространенного рабочего процесса и создайте достаточно, чтобы сделать это путешествие очевидным образом лучше. Это позволяет быстро доставлять ценность, получать критически важную обратную связь и создавать импульс для будущей итерации.

Дизайн для расширения и вклада

Ваша команда центральной платформы не может — и не должна — строить все. Успешная платформа расширяема, позволяя другим командам вносить свои собственные инструменты и услуги стандартизированным способом. Создайте свою платформу с четкими API, хорошей документацией и четко определенной моделью вклада. Этот «открытый» подход масштабирует ценность платформы, использует опыт домена от других команд и препятствует тому, чтобы команда платформы стала узким местом.



Приоритет четкой обратной связи и наблюдаемости

Как вы знаете, что ваша платформа работает хорошо? Наши данные за 2025 год показали, что возможности платформы, наиболее коррелирующие с положительным пользовательским опытом, «дают мне четкую обратную связь о результатах моих задач». Разработчики должны легко понимать, что происходит, когда они используют платформу, особенно когда что-то не удается. Обеспечить четкую, действенную обратную связь, журналы и диагностику, чтобы дать разработчикам возможность самостоятельно обслуживать и устранять неполадки.

Безопасные текущие инвестиции, демонстрирующие ценность

Платформа — это не проект «один-и-сделано»; это живой продукт, который требует постоянных инвестиций для развития с потребностями разработчиков и бизнеса. Используйте стратегии измерения в следующем разделе (например, показатели DORA, H.E.A.R.T. и платформенные табло), чтобы связать улучшения вашей платформы непосредственно с производительностью разработчиков и, в конечном итоге, с организационной производительностью. Это создает добродетельный цикл инвестиций и улучшений.

Общие препятствия и антипаттерны

По мере того, как организации инвестируют в разработку платформ, несколько распространенных подводных камней могут подорвать ценность платформы, расстроить разработчиков и помешать организации реализовать преимущества своих инвестиций.

«Строим, и придут» антипаттерн

Это основной симптом отношения к платформе как к техническому проекту, а не продукту. Команда создает платформу, основанную на том, что, по их мнению, нужно разработчикам, не проводя никаких исследований пользователей, интервью или проверки. Они полностью сосредоточены на технологии и инженерии, предполагая, что ее ценность будет самоочевидной. Эта платформа в конечном итоге становится городом-призраком, потому что она либо не решает реальные, болезненные проблемы для разработчиков, либо не соответствует их существующим рабочим процессам.

Платформа «башня из слоновой кости»

Это препятствие возникает, когда команда центральной платформы диктует архитектуру и инструменты с высоких, обеспечивающих соблюдение жестких стандартов без сотрудничества или цикла обратной связи. Они выступают в качестве привратников технологии, а не воспитателей разработчиков. Этот подход оставляет разработчиков чувствовать себя бесправными и часто создает «теневые ИТ» или неофициальные обходные пути, чтобы обойти ограничения платформы, побеждая ее цель.

«Ловушка для билетов»

Здесь команда платформы работает как торговый автомат для инфраструктуры, а не как активатор самообслуживания. Их работа полностью реактивна и обусловлена бесконечной очередью билетов от разработчиков (например, «предоставьте мне базу данных», «настройте конвейер CI/CD»).

Это создает узкое место и добавляет работу как для команды платформы, так и для разработчиков. Команда тратит все свое время на одноразовые запросы и никогда не имеет возможности создавать сплоченные возможности самообслуживания, которые обеспечивают реальную ценность платформы.

Подход «большого взрыва»

Некоторые организации пытаются создать всеобъемлющую, всеохватывающую платформу, которая решает каждую мыслимую проблему, прежде чем выпустить какую-либо ее часть. Эта стратегия является высокорискованной, дорогостоящей и задерживает ценную обратную связь. К тому времени, когда «идеальная» платформа наконец-то запущена, потребности разработчиков часто менялись. Подход, ориентированный на продукт, напротив, обеспечивает ценность постепенно и итерирует на основе реального использования.



Мышление «один размер подходит всем»

В попытке максимизировать стандартизацию команды платформ могут создать единую, жесткую «золотую клетку», которая не учитывает разнообразные потребности разных команд разработчиков.

Успешная платформа обеспечивает ограничения и золотые пути, предлагая гибкость для команд, чтобы использовать правильные инструменты для своей конкретной работы.

Неспособность обеспечить спонсорство исполнительной власти

Инициатива платформы — это значительные организационные и технические инвестиции, которые требуют преодоления существующих бункеров.Без четкого, видимого и долгосрочного исполнительного спонсорства команда платформы может быть лишена ресурсов, изо всех сил пытается получить бай-ин от других отделов и не имеет полномочий для проведения необходимых кросс-функциональных изменений.

Как измерить качество внутренней платформы



Вопросы опроса DORA

Исследование DORA измерило эту способность, задав следующие вопросы: 3

— В последние три месяца вы пользовались платформой?

• В какой степени ваша платформа демонстрирует следующие характеристики?

Платформа ведет себя так, как я ожидаю.

Платформа эффективно абстрагирует сложность базовой инфраструктуры.

Платформа дает мне четкую обратную связь о результатах моих задач.

Платформа помогает мне создавать и запускать надежные приложения и услуги.

Платформа помогает мне создавать и запускать безопасные приложения и услуги.

Платформа помогает мне следить за требуемыми процессами (например, обзоры кода, выписки из системы безопасности).
Платформа проста в использовании.

Платформа предоставляет инструменты и информацию, необходимые для самостоятельной работы.

Команда платформы действует на обратную связь, которую я предоставляю.

Пользовательский интерфейс платформы прост и чист.

Задачи, которые я выполняю на платформе, хорошо автоматизированы.

• Есть ли у вашей организации специальная команда?

Рамочные меры E.E.A.P.T

Рамки H.E.A.R.T. предоставляют дополнительные сигналы, которые могут помочь измерить инженерную практику платформы.

Что проверить	Что измерить
Счастье (H.E.A.R.T.)	Настроения разработчиков и удовлетворенность платформой
Вовлечение (H.E.A.R.T.)	Как часто и насколько глубоко разработчики используют функции платформы
Усыновление (H.E.A.R.T.)	Скорость новых команд и сервисов на борту платформы
Удержание (H.E.A.R.T.)	Скорость, с которой команды продолжают использовать платформу
Успешная работа (H.E.A.R.T.)	Эффективность и результативность выполнения разработчиками ключевых рабочих процессов

Дополнительные меры

Кроме того, вы можете измерить влияние ваших усилий по разработке платформы с помощью показателей производительности программного обеспечения и удовлетворенности разработчиков.

Используйте показатели производительности программного обеспечения DORA (внутренне и внешне)

Пять показателей производительности программного обеспечения DORA (первое время для изменений, частота развертывания, время восстановления после неудачного развертывания, частота сбоев и скорость переделки развертывания) приносят пользу командам платформы и приложений.

1.Для команды платформы: Ваша команда платформы должна использовать эти показатели для измерения собственной производительности, помогая вам улучшить скорость и стабильность изменений на платформе.

2. Для команд приложений: Платформа должна предоставлять эти показатели своим пользователям. Путем автоматического инструментария и раскрытия этих показателей для каждой службы вы даете разработчикам возможность получить представление высокого уровня, которое им необходимо для понимания их собственной производительности доставки и увидеть прямое влияние платформы на их работу.

Отслеживание удовлетворенности разработчиков

Регулярно опрашивайте разработчиков (клиентов платформы) на предмет их удовлетворенности платформой. Используйте простые меры, такие как удовлетворенность клиентов (CSAT) или оценка чистого промоутера (NPS), в сочетании с качественной обратной связью, чтобы отслеживать настроения с течением времени и определять области для улучшения.

Выводы

Качественная внутренняя платформа является конечным усилителем в модели возможностей DORA ИИ. Она действует как «мощеный путь», который надежно и надежно масштабирует преимущества всех других возможностей - от рабочих процессов с помощью ИИ до функций, ориентированных на пользователя.

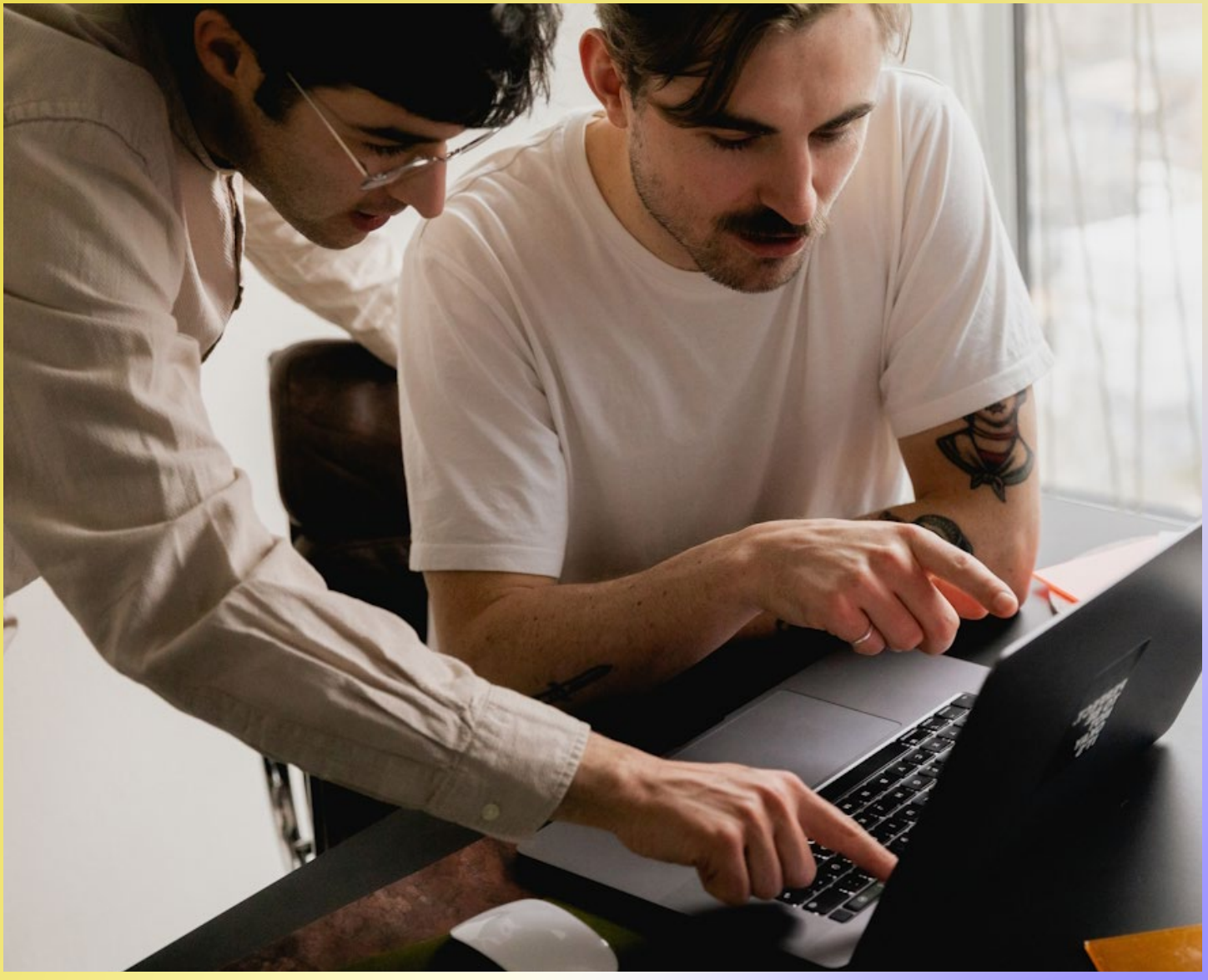
Абстрагируя сложность и автоматизируя доставку, платформа гарантирует, что индивидуальный прирост производительности от ИИ не будет потерян из-за беспорядка вниз по течению. Это основополагающая система, которая переводит потенциал ИИ в измеримую, системную и устойчивую организационную производительность.

1. «Как Google это делает: ваше руководство по разработке платформ». https://cloud.google.com/blog/products/application-modernization/a-guide-to-platform-engineering
2. «Перейти к разработке платформы в Google Cloud». https://cloud.google.com/solutions/platform-engineering
3. «Опросы: разработка платформы». https://dora.dev/ai/capabilities-model/questions/#quality-internal-platform

Оценка и приоритизация возможностей

Нейтен Харви

Руководитель DORA, Google Cloud



Определение профиля команды

Используя данные, собранные почти 5000 специалистов по технологиям со всего мира, мы провели кластерный анализ, чтобы понять человеческие и системные факторы, которые влияют на производительность. Мы определили общие закономерности, архетипы команд и профили команд. Наш подход к статистической кластеризации включал следующие факторы:

Выступление команды

Этот фактор измеряет воспринимаемую эффективность и силу сотрудничества непосредственной команды человека.

Производительность продукции

Этот фактор измеряет успех и качество продуктов или услуг, которые команда создает, основываясь на таких характеристиках, как помощь пользователям в выполнении важных задач и обеспечение безопасности информации, а также показатели производительности, такие как задержка.

Пропускная способность программного обеспечения

Это отражает скорость и эффективность процесса доставки программного обеспечения.

Нестабильность доставки программного обеспечения

Это отражает качество и надежность процесса доставки программного обеспечения.

Индивидуальная эффективность

Этот фактор отражает самооцененную эффективность человека и чувство выполненного долга на работе.

Ценная работа

Это измеряет количество времени, которое человек тратит на выполнение работы, которую он считает ценной и стоящей.

трение

Это измеряет степень, в которой трение препятствует работе человека. Более низкие количества трения обычно считаются положительным результатом.

выгорание

Это измеряет чувство истощения и цинизма, связанное с работой. Более низкое количество выгорания обычно считается положительным результатом.

Организации, команды и отдельные лица обычно стремятся повысить производительность команды, производительность продукта, пропускную способность программного обеспечения, индивидуальную эффективность и ценную работу.Они стремятся уменьшить нестабильность, трение и выгорание доставки программного обеспечения.

Наш анализ выявил семь различных групповых архетипов, начиная от тех, которые преуспевают в здоровой, устойчивой среде (гармоничные высокодостижимые), и заканчивая теми, кто попал в ловушку технического долга (устаревшее узкое место) или неэффективных процессов (сдержанных процессом).

На рисунке 18 показаны уровни производительности семи командных архетипов. Пунктирная линия представляет собой среднее число всех респондентов, в то время как цветная линия показывает производительность каждого кластера. Помните, что команды обычно стремятся увеличить свой балл по некоторым факторам, таким как индивидуальная эффективность и пропускная способность программного обеспечения, при одновременном снижении балла по другим факторам, таким как выгорание и трение.

Уровень производительности семи командных архетипов

Кластер 1: Основополагающие задачи



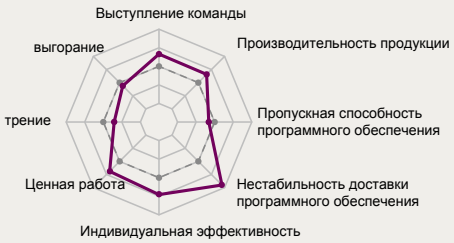
Кластер 2: Унаследованное узкое место



Кластер 3: ограниченный процессом



Кластер 4: Высокое воздействие, низкая каденция



Кластер 5: Стабильный и методичный



Кластер 6: Прагматичные исполнители



Кластер 7: Гармоничные высококлассники



Имена и описания для каждого из этих кластеров являются интерпретацией данных. Ваша команда может видеть аналогичные уровни производительности в качестве данного кластера, но может не чувствовать, что название кластера или описание хорошо описывают вашу команду. Рисунок 18: Уровни производительности семи архетипов команды

Кластер 1: Основополагающие задачи

Эти команды застряли в режиме выживания, столкнувшись со значительными проблемами с фундаментальными пробелами в своих процессах, окружающей среде и результатах.

Процент респондентов: 10% респондентов входят в группу 1.

Показатели производительности: выход команды, поставка продукции и создание стоимости неизменно низки.

Благополучие команды: данные показывают высокий уровень выгорания и значительные трения.

Стабильность системы: существуют значительные проблемы со стабильностью программного обеспечения и операционной среды.

Кластер 1: Основополагающие задачи



Рисунок 19: Кластер 1: Основные проблемы

Кластер 2: Унаследованное узкое место

Команды в этом кластере находятся в постоянном состоянии реакции, где нестабильные системы диктуют свою работу и подрывают их моральный дух.

Процент респондентов: 11% респондентов входят в группу 2.

Показатели производительности: Показатели производительности продукта низкие. В то время как команда регулярно обновляется, реализованная ценность уменьшается из-за текущих проблем с качеством.

Благополучие команды: Данные указывают на сложную рабочую среду. Члены команды сообщают о повышенном уровне трения и выгорания.

Стабильность системы: существуют значительные и частые проблемы со стабильностью программного обеспечения и его операционной среды, что приводит к большому объему незапланированной, реактивной работы.



Рисунок 20: Кластер 2: Унаследованное узкое место

Кластер 3: ограниченный процессом

Эти команды работают на беговой дорожке. Несмотря на работу над стабильными системами, их усилия расходуются неэффективными процессами, приводящими к высокому выгоранию и низкому воздействию.

Процент респондентов: 17% респондентов входят в группу 3.

Показатели эффективности: эти команды испытывают низкую эффективность и создание ограниченной ценности для клиентов или бизнеса.

Благополучие команды: Данные показывают высокий уровень выгорания и трения. Это говорит о том, что текущие рабочие процессы и процессы создают сложную и неустойчивую рабочую среду для команды.

Стабильность системы: Программное обеспечение и операционная среда команды стабильны и надежны. Это указывает на то, что техническая нестабильность не является основным фактором, влияющим на проблемы в производительности и благополучии.



Рисунок 21: Кластер 3: Сдержанный процессом

Кластер 4: Высокое воздействие, низкая каденция

Эти команды производят высокопроизводительную работу, что отражается в высокой производительности продукта и высокой индивидуальной эффективности, однако это в сочетании с моделью доставки с низким уровнем каденции, характеризующейся низкой пропускной способностью программного обеспечения и высокой нестабильностью.

Процент респондентов: 7% респондентов входят в группу 4.

Показатели эффективности: Команда последовательно достигает уровней производительности высшего уровня. Показатели эффективности и производительности продукта сильны.

Благополучие команды: данные указывают на среду с низким коэффициентом трения, предполагая, что командные процессы эффективны и эффективны.

Стабильность систем. Рабочая среда отличается высокой нестабильностью. Это создаёт серьёзный риск для надёжности сервисов и устойчивости работы в долгосрочной перспективе.



Рис. 22. Кластер 4. Высокая отдача, редкие поставки

Кластер 5: Стабильный и методичный

Эти команды являются постоянными ремесленниками мира программного обеспечения, обеспечивая высококачественную, ценную работу в преднамеренном и устойчивом темпе.

Процент респондентов: 15% респондентов входят в группу 5.

Показатели производительности: показатели качества продукции и создания стоимости неизменно положительные. Однако пропускная способность программного обеспечения команды находится в более низком процентиле, что указывает на более целенаправленный темп работы.

Благополучие команды: данные показывают низкий уровень выгорания и трения, что указывает на здоровую и устойчивую командную среду.

Стабильность системы: Программное обеспечение и операционная среда команды характеризуются высокой стабильностью и надежностью.



Рисунок 23: Кластер 5: Стабильный и методический

Кластер 6: Прагматичные исполнители

Эти команды постоянно выполняют работу с впечатляющей скоростью и стабильностью, даже если их рабочая среда не достигла пика вовлеченности.

Процент респондентов: 20% респондентов входят в группу 6.

Показатели производительности: Производительность программного обеспечения сильна, с пропускной способностью выше среднего и низкой нестабильностью. Команда поддерживает устойчивую каденцию ценного продукта, надежно удовлетворяя ожидания.

Благополучие команды: данные показывают средний уровень зарегистрированного выгорания и трения. Это указывает на рабочую среду, которая является функциональной и устойчивой, но может не иметь сильных драйверов вовлеченности.

Стабильность системы: программное обеспечение и операционная среда команды стабильны и надежны, обеспечивая прочную основу, необходимую для их высокой производительности.



Рисунок 24: Кластер 6: Прагматичные исполнители

Кластер 7: Гармоничные высококлассники

Вот как выглядит превосходство — добродетельный цикл, в котором стабильная среда с низким коэффициентом трения позволяет командам выполнять высококачественную работу устойчиво и без выгорания.

Процент респондентов: 20% респондентов входят в группу 7.

Показатели эффективности: команда показывает положительные показатели по нескольким областям, включая благополучие команды, результаты продукта и доставку программного обеспечения.

Благополучие команды: рабочая среда характеризуется низким уровнем выгорания и трения.

Стабильность системы: команда работает на стабильной технической основе, которая поддерживает скорость и качество их работы.



Рисунок 25: Кластер 7: Гармоничные достижения

Где сегодня ваша команда?

Улучшение производительности начинается с понимания текущих условий для команды.

Тем не менее, точный ответ на вопрос, какой кластер лучше всего описывает команду сегодня, не имеет решающего значения. более важно определить область, которая больше всего выиграет от усилий по улучшению.

Есть несколько различных способов оценить текущую производительность команды, чтобы определить, какой кластер лучше всего описывает, как вы работаете сегодня.

беседа

Соберите кросс-функциональную команду, которая отвечает за одно приложение или услугу, и поговорите о том, как работает команда. Команда может использовать семь кластеров в качестве руководства и договориться о том, какой из них лучше всего описывает, как идут дела.

Вот один из способов структурирования разговора.

Просмотрите профили команды:

Просмотрите содержание этого раздела, охватывающего семь профилей команды.

1.Создание консенсуса:

Перечислите семь профилей команды на доске или экране.

Дайте каждому участнику три листка бумаги для молчаливого анонимного голосования.

Голосование: Каждый участник должен отдать голоса за профиль, который, по его мнению, лучше всего соответствует текущим условиям команды. Каждый участник должен отдать три голоса, но может распределить их по своему усмотрению. Например, если участник чувствует, что команда лучше всего отображает кластер из шести, этот участник отдаст все три голоса за этот профиль.

○ Практически результаты.

2. Раскройте голоса и обсудите: Раскройте результаты для команды и обратите внимание на то, где были сгруппированы голоса. Проведите краткую дискуссию в формате таймбокса. Цель состоит не в том, чтобы найти один «идеальный» ярлык, а в том, чтобы построить общий консенсус по текущим проблемам команды. Некоторые полезные вопросы, которые нужно задать:

Глядя на то, где сгруппированы наши голоса, какие проблемы из этого профиля кажутся наиболее реальными в вашей повседневной работе?

Вы ожидали, что другой профиль получит больше голосов, и если да, то почему?

Каковы общие темы или разочарования, которые менее популярные профили разделяют с нашей командой?

Оценка

Проведите опрос, который собирает индивидуальные ответы для каждого из восьми факторов, используемых в нашем кластерном анализе. Это даст вам представление о текущем опыте команды. Используя эту информацию, вы можете сопоставить свою команду с одним из семи кластеров.

Вопросы, использованные в исследовании DORA 2025 года, доступны на веб-сайте DORA.1.

Понимание того, как работает команда, может помочь выявить, какие области получат наибольшую выгоду от улучшения, а затем проинформировать возможности ИИ о приоритетах.

Например, команды, сталкивающиеся с «фундаментальными проблемами», могут захотеть сосредоточить свои усилия по улучшению на производительности команды, поддерживаемой возможностями ИИ, которые улучшают сотрудничество. Высокий эффект, команды с низкой каденцией могут захотеть сосредоточиться на снижении нестабильности доставки программного обеспечения, возможно, используя ИИ для автоматизированного тестирования и анализа.

1. «Опросы». https://dora.dev/research/2025/team-profiles/questions



Углублённый анализ с помощью карты потока создания ценности

Роб Эдвардс

Доставка приложений Lead, Google Cloud

Дэйв Стэнк

Специалист по работе с клиентами Google Cloud

Как неоднократно демонстрировали исследования DORA, организационная производительность зависит от быстрой доставки высококачественного программного обеспечения конечным пользователям. Но как это программное обеспечение действительно создается? Программная инженерия - это командная деятельность со многими участниками-людьми и автоматизированными процессами, каждый с уникальной перспективой.

Чтобы улучшить поток доставки программного обеспечения, нам сначала нужно общее, высокоуровневое понимание его текущего состояния. Картирование потоков ценности (VSM) предлагает подробный, основанный на данных взгляд на то, как работа на самом деле проходит через ваши системы. VSM - это практика визуализации и анализа каждого шага в вашем процессе - от идеи до клиента - для выявления реальных точек трения, узких мест и задержек.

Важно прояснить общую точку неразберихи: разницу между отображением потока ценностей и управлением потоком ценностей.

Картирование потока ценностей: это тактическое упражнение, которое мы описываем в этой главе, инструмент для визуализации и анализа конкретного рабочего процесса, часто на доске, для построения общего понимания.

Управление потоками ценности: это стратегическая, целостная дисциплина непрерывного управления и улучшения всего сквозного потока ценности, от первоначальной идеи до клиента, через взаимосвязанные команды и системы.

В этой главе мы намеренно сосредотачиваемся на картографическом упражнении и использовании процесса доставки программного обеспечения (от «обязательства к действию» в соответствии с показателями производительности программного обеспечения DORA) в качестве практической отправной точки, чтобы заставить команды задуматься о своем потоке.

Как карта потока помогает расставить приоритеты

Метрики и опыт могут рассказать вам, что вы испытываете (например, «Время ожидания изменения становится больше»), но VSM показывает вам, где и почему это происходит (например, «Наш процесс проверки кода имеет трехдневное среднее время ожидания, и наше сканирование безопасности занимает восемь часов для запуска»).

В эпоху быстрого внедрения ИИ наибольший риск заключается в том, чтобы вливать огромные инвестиции в хаотичную деятельность, которая не двигает иглу. Исследование DORA показывает, что ИИ действует как усилитель позитивного и негативного поведения и результатов, поэтому важно выявлять и устранять дисфункцию в потоке ценности. VSM - это то, что отделяет неорганизованную деятельность от целенаправленного улучшения, позволяя вам ориентироваться на наиболее эффективные возможности.

Как карта потока помогает повысить отдачу от ИИ

Исследование DORA подтвердило, что VSM является множителем силы, который превращает инвестиции в ИИ в истинное конкурентное преимущество. Результаты ясны: одно только внедрение ИИ оказывает лишь незначительное влияние на организационную эффективность. Однако этот эффект все больше усиливается в сочетании с сильными практиками VSM.

VSM смягчает влияние ИИ на организационную производительность

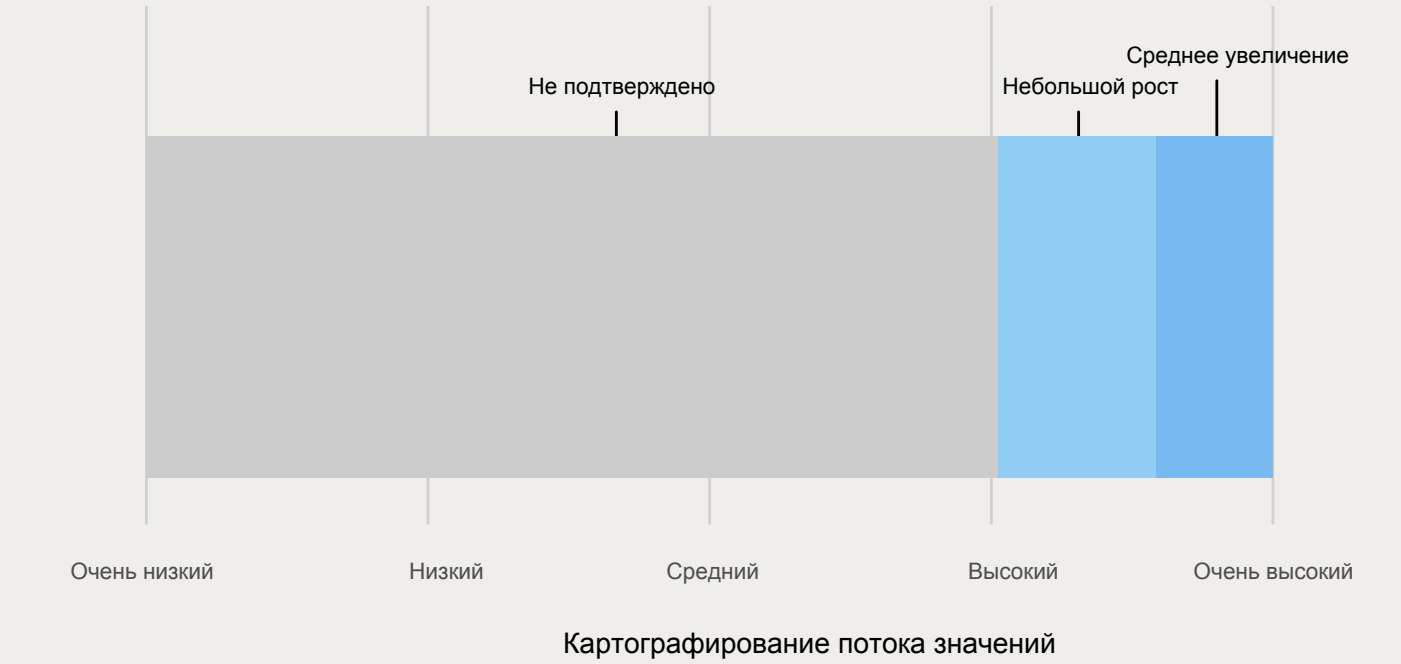


Рисунок 26: VSM смягчает влияние ИИ на эффективность организации

Успешное внедрение ИИ требует большего, чем просто принятие новых инструментов; это трансформация систем и процессов. VSM гарантирует, что повышение эффективности от ИИ направляется на решение системных ограничений. Без VSM ИИ рискует создать локальные карманы производительности, которые просто теряются из-за хаоса и узких мест.

Например, если ваше упражнение VSM показывает, что обзор кода является основным узким местом, VSM направляет вас исследовать, где улучшить, в том числе эксперименты по применению ИИ к процессу обзора. Это гораздо эффективнее, чем использование ИИ для простого создания большего количества кода, который будет только накапливать и ухудшать узкое место.

Команды, которые практикуют VSM, видят более высокую производительность команды и тратят значительно больше времени на ценную работу и достигают лучших результатов продукта.

Основные принципы картирования потока создания ценности

Для достижения этого целенаправленного улучшения у нас есть пять основных принципов.

Приоритетность действия картирования над артефактом:

VSM - это больше, чем просто карта. Наиболее ценным результатом VSM является совместный процесс его создания. Цель состоит в том, чтобы привлечь всех участников потока ценностей и запрашивать их различные перспективы. Обнаруженные общие истины и кросс-функциональное обязательство, построенное во время этого акта картирования, - это то, что действительно стимулирует улучшение.

Перейдите от ментального беспорядка к общей карте:

VSM - это упражнение в получении деталей процесса из головы каждого и в общее пространство, такое как доска. Эта общая карта позволяет команде развить коллективное понимание рабочего процесса и легко обнаружить скрытые узкие места и неэффективность. Ключевой тест - спросить: «Можете ли вы нарисовать поток ценности доставки программного обеспечения на доске?»

Сосредоточьтесь на потоке, а не только на скорости:

Основная цель состоит в том, чтобы сделать работу плавной и предсказуемой. Это требует перехода от оптимизации изолированных шагов к оптимизации общей системы. Для этого необходимо измерять ключевые показатели по каждому переходу между шагами.

Создание культуры непрерывного совершенствования:

Поскольку VSM является непрерывным циклом, команды должны регулярно пересматривать свою карту потока ценностей. Команды должны иметь возможность экспериментировать, учиться и адаптироваться, не опасаясь репрессий, а затем делиться этими уроками по всей организации.

Построить на основе технического совершенства:

Быстрый, плавный поток невозможен без технического совершенства, что, как правило, является одной из вещей, обеспечиваемых хорошо продуманной внутренней платформой. Эта платформа должна предлагать разработчикам «мощеные дороги» для таких возможностей, как тестирование и доставка, которые абстрагируют сложность и делают высокопроизводительную работу масштабируемой.

Легкое упражнение VSM

Вот простой способ начать:

Перед картографированием команда должна четко определить бизнес-результаты или результаты клиентов, которые они пытаются улучшить. Это гарантирует, что усилия VSM сосредоточены на решении реальной бизнес-задачи, а не просто на оптимизации процесса.

Этот шаг отвечает на вопросы: «Почему мы это делаем?» (например, «Снизить время выполнения заказа») и «Как выглядит «лучше»?» (например, «от семи дней до двух дней»).1

Соберите команду Получите всех, кто вносит свой вклад в продукт в комнате (виртуальной или физической). Вмешанная межфункциональная команда, вероятно, будет состоять из отдельных отдельных организационных элементов.

«Команды»; важно, чтобы все люди, вовлеченные в область, на которой вы сосредоточены, были вовлечены в упражнение.

Сопоставьте поток на доске, сопоставьте шаги высокого уровня.

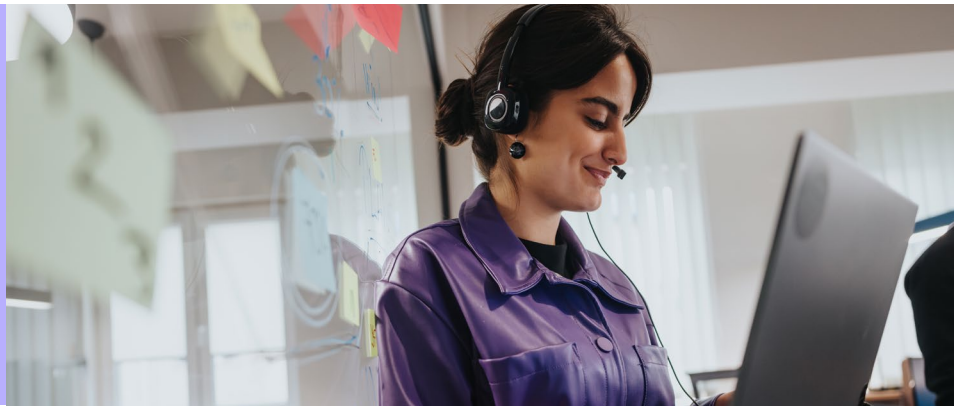
СОВЕТ:

Чтобы увидеть реальный поток, вы должны включить представителей ключевых команд вверх по течению (таких как продукт и дизайн) и вниз по течению (таких как безопасность, опционы и юридические). Самые болезненные узкие места и самые длинные состояния ожидания почти всегда встречаются в передачах между командами.



СОВЕТ:

Для свежей перспективы, решайте свою задачу в обратном направлении. Начав в конце и прокладывая путь назад к началу, вы нарушаете свои нормальные модели мышления, которые часто открывают новые идеи.



Крайне важно нанести на карту как «счастливый» путь (идеальный поток), так и процесс восстановления после инцидента («поток ценности восстановления»), поскольку это часто обнаруживает скрытые трения и задержки. В то время как полная карта переходит от идеи к клиенту, мощной отправной точкой часто является область от кода, предназначенного для работы в производстве, поскольку команды имеют здесь наибольшее количество агентств. После того, как поток текущего состояния отображается, команда должна проанализировать его, чтобы определить ограничения и собрать данные:

Определите состояния ожидания: Для каждого шага определите, где работа ждет кого-то или что-то еще. Это ваши узкие места.

Причины гипотезы: Обсудите потенциальные причины задержек в каждом состоянии ожидания. Это где вы свяжете узкое место с конкретным пробелом в возможностях.

Соберите метрики: Для каждого шага оцените или измерьте ключевые метрики:

Время обработки: время, необходимое для активной работы над задачей.

Время ожидания: время, которое задача проводит в очереди в ожидании следующего шага.

• Процент полной и точной (%C&A): Процент работы, которую может использовать нисходящий этап без необходимости исправлений. Например, если шесть из 10 запросов на вытягивание, отправленных в QA, принимаются без переделки, %C&A составляет 60%.

Когда вы впервые запустите это упражнение на доске, ваши показатели, скорее всего, будут оценками (например, «Обзор кода обычно занимает примерно два дня»).

В то время как современные VSM могут в конечном итоге извлекать реальные, живые данные из ваших инструментов, будьте осторожны, чтобы не сделать метрическую точность главной целью. Истинная ценность заключается в использовании этих чисел - будь то приблизительные оценки или точные наблюдения - для лучшего общения и поиска способов улучшить поток работы.

На рисунке 27 показан пример карты потока значений. Общее время активного процесса (PT) составляет всего около одного дня (~24,5 часа), в то время как общее время ожидания составляет почти четыре дня (~92,5 часа). Это приводит к общему сквозному времени выполнения примерно 117 часов и эффективности потока ~21%. Это подчеркивает, что работа находится в состоянии ожидания почти четыре пятых времени.

Картографируйте будущую статистику: После выявления ограничений и сбора показателей вашего текущего состояния спроектируйте идеальную карту «будущего состояния». Эта новая карта представляет вашу цель, показывая, как должен выглядеть процесс с устраненными узкими местами и временем ожидания. Пробелы между вашими текущими и будущими состояниями становятся вашими конкретными, действенными целями улучшения.

VSM: узкое место в обзоре кода

Process:	Commit Code is committed to repo	Pull request /Coding WT:24h	Code review	Deploy to QA env	CI build	Functional tests	Perf regression tests	Release review	Deploy to pre-prod	Deploy to prod
Responsible team:	Developer	Developer	Technical lead	QA Team	Perf regression tests	QA Team	Performance Eng	Change approval board (CAB)	Release eng	App ops
Metrics	PT: - WT: - %C&A: -	PT:12h WT: 24h %C&A: 80%	PT: 2h WT: 20h %C&A: 70%	PT: 0.5h WT: 1h %C&A: 95%	PT: 0.5h WT: 2h %C&A: 90%	PT: 4h WT: 16h %C&A: 85%	PT: 12h WT: 24h %C&A: 80%	PT: 1h WT: 4h %C&A: 100%	PT: 0.5h WT: 0.5h %C&A: 95%	PT: 1h WT: 1h %C&A: 99%

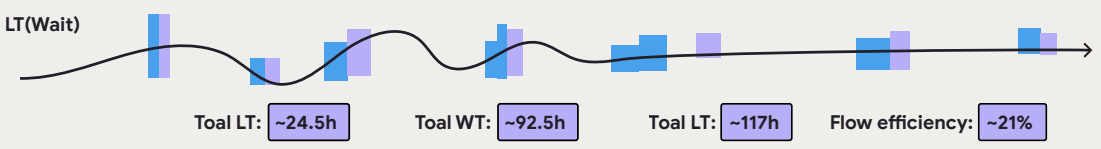


Рисунок 27: VSM: узкое место в обзоре кода

Как связать выводы из карты потока с возможностями для ИИ

Используйте результаты вашего упражнения VSM для нацеливания на конкретные возможности ИИ для максимального воздействия.

Если вы обнаружили…	В первую очередь…
Непоследовательная проверка кода	Откройте ИИ доступ к внутренним данным, чтобы агенты проверяли код с учётом контекста. Применяйте надёжные практики контроля версий, чтобы изменения оставались небольшими и удобными для проверки.
Разработчики тратят большое количество времени на поиск информации или понимание существующего кода.	«ИИ-доступные внутренние данные» для предоставления мгновенных, контекстно-ориентированных ответов
Много доработок и ошибок, которые обнаруживают на поздних этапах	Комплексная автоматизация тестирования. ИИ помогает создавать, запускать и поддерживать более полные наборы тестов.
Код копится в очереди на проверку	Используйте ИИ, чтобы ускорить проверку кода. У команд, которые быстрее проверяют код, результативность поставки ПО выше на 50 %².

Пример: использование VSM для разработки платформ

VSM-фреймворк не ограничивается конвейером доставки вашего приложения; это невероятно эффективный инструмент для инженерных команд платформы для улучшения своих собственных внутренних продуктов.

В этой модели вы относитесь к своей платформе как к продукту, а к разработчикам как к своим клиентам.

Ваш пользователь: внутренний разработчик или команда разработчиков.

Ваш поток ценностей: критические поездки разработчиков, необходимые для создания, тестирования и доставки программного обеспечения с использованием вашей платформы.

Ваша цель: картировать, измерять и улучшать опыт разработчиков, устраняя их узкие места, когнитивную нагрузку и время ожидания.

Вы можете использовать то же самое легкое упражнение VSM, описанное выше, но вместо этого вы нанесете на карту эти поездки разработчиков.

Ключевые пользовательские поездки разработчиков на карту:

Золотой путь: строительство новых, готовых к производству услуг.

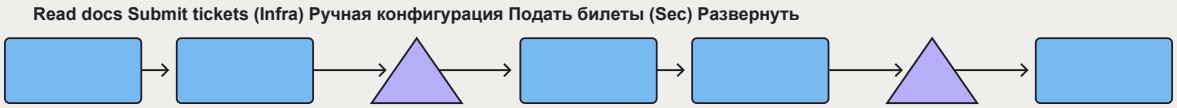
Внутренняя петля: разработчик, который изменяет код и видит его работу в среде разработки.

Внешний цикл: разработчик объединяет функцию и развертывает ее в производстве.

Инцидент: разработчик, который пытается отладить производственный вопрос.

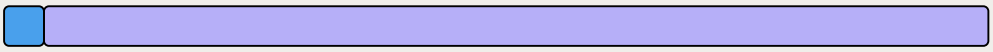
• Погрузка: новый разработчик от получения своего ноутбука до объединения своего первого запроса на вытягивание / слияние.

VSM: Путешествие «Новый микросервис» (текущее состояние)



Время обработки:	2h	2h	0	2h	10m	0	30m
Время ожидания:	0	0	**2 дня**	0	0	**3 дня**	0
%C&A:	50%	50%	100%	60%	95%	100%	90%

Сводная разбивка времени



Время работы: 4,8 часа **Время ожидания: 5 дней**

Общее время выполнения: 5 дней, 6 часов и 40 минут
Общее время работы: 6 часов 40 минут
Эффективность потока: 5,3%

Рисунок 28: VSM нового служебного путешествия (текущее состояние)

Пример VSM: новый путь микросервиса

Представьте себе, что вы наносите на карту золотой путь

для строительства новых лесов.

1.Карта потока (текущее состояние):

Открытие: разработчик перемещается по нескольким страницам вики.

○Билет: Отправляет запрос на репозиторий в инфраструктуру. (Подождите: два дня)

Ручная работа: Копии кода boilerplate; попытки конфигурации трубопровода.

○Билет: Отправляет запрос на сканирование системы безопасности. (Подождите: три дня)

Результат: наконец-то развернута базовая услуга.

2. Соберите метрики:

Общее время выполнения: более пяти дней

Труд разработчика: от четырех до шести часов разочаровывающей работы с низкой стоимостью.

Профиль риска: высокий. Ручная работа приводит к ошибкам и дрейфу конфигурации.

Основное трение: ручные «билетные операции», передачи и высокая когнитивная нагрузка.

3. Составьте карту будущего состояния и подключитесь к ИИ:

Будущее состояние: разработчик запускает одну команду и имеет безопасную, готовую к производству службу, развернутую за пять минут.

Этот анализ представляет собой четкий бизнес-кейс: будущее государство - это агент, работающий на ИИ. Обученный вашим внутренним документам, политике безопасности и передовым методам, этот агент вызывается этой единственной командой. Он организует весь путь - предоставление, конфигурацию и генерацию кода на основе ИИ - превращая пятидневное узкое место в пятиминутную, асфальтированную дорогу.

VSM нового сервисного путешествия (будущее состояние)

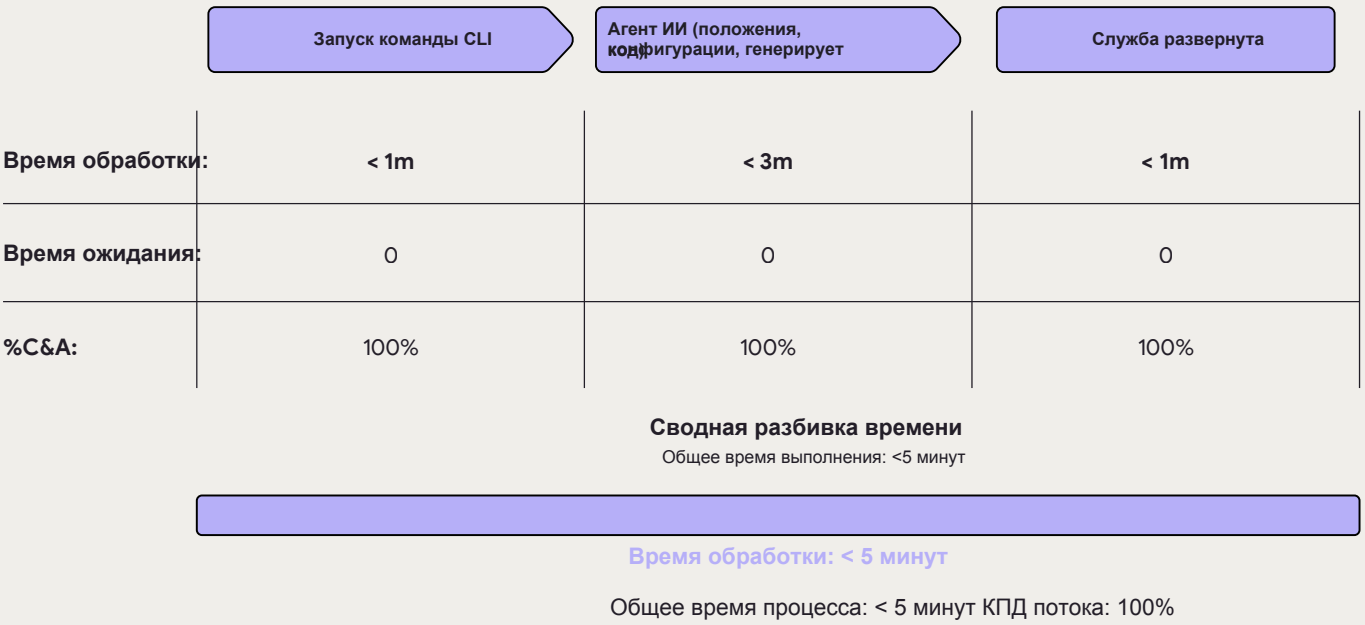


Рисунок 29: VSM нового служебного путешествия (будущее состояние)

Применяя VSM к своим внутренним путешествиям разработчиков, вы превращаете разработку платформы из центра затрат (управление билетами) в драйвер стоимости (ускорение всех команд). Вы используете те же принципы VSM, чтобы найти реальные ограничения на производительность разработчиков и применить ИИ для решения их на системном уровне.

Выводы

Мы считаем, что это исследование — больше, чем просто наблюдения за VSM; это задача для улучшения командных процессов.

Первый шаг к выходу из цикла текущей деятельности может быть непростым, но он прост. Спросите свою команду: «Можем ли мы нарисовать наш поток стоимости доставки программного обеспечения на доске?»

Если ответ отрицательный, или если на рисунке больше вопросов, чем ответов, вы нашли свою отправную точку. Если ответ да, то у вас есть карта и вы можете сразу перейти к этапу анализа: измерение времени процесса, времени выполнения и %C&A для каждого шага, чтобы найти скрытые состояния ожидания. Этот единственный разговор - начало становиться лучше в улучшении.

1. «Как использовать отображение потока ценности для улучшения доставки программного обеспечения: руководство по отображению потока ценности». https://dora.dev/guides/value-stream-management
2.Ускорить: Состояние DevOps Report 2023. https://dora.dev/dora-report-2023



Командный семинар по расстановке приоритетов

Седрик Яо

Инженер по решениям, ИИ Application Innovation, Google Cloud

Оценка - это просто данные без плана действий, которому нужно следовать. Сделав первый шаг к изменению, основанному на оценках, эти идеи превращаются в целенаправленные коллективные действия.

Мы создали следующий 90-минутный семинар для разделения различных и конвергентных этапов мышления.

Цель состоит в том, чтобы привести всю команду к общему пониманию ваших основных ограничений, достичь консенсуса в отношении наиболее эффективных улучшений и найти согласованный приоритетный список того, что делать дальше.



Прежде чем начать: Предпосылки

Этот 90-минутный семинар предназначен для работы с данными, а не для их сбора. У вас должно быть несколько вещей, готовых максимально использовать ваше время вместе.

Посредник: Этот человек ведет дискуссию, следит за часами и остается нейтральным. Он может быть членом команды, но часто легче, если он не вкладывается напрямую в результаты. Он также может задавать уточняющие вопросы в своей роли аутсайдера, чтобы способствовать разговору, когда это необходимо.

Вся команда: Концепция доставки программного обеспечения как командного вида спорта эхом повторялась в залах индустрии программного обеспечения на протяжении десятилетий. Ваша команда - это группа людей, с которыми вы работаете на ежедневной основе, чтобы создавать, поддерживать или поддерживать одно и то же приложение или услугу. Этот семинар предназначен для того, чтобы принести эмпатию другим частям команды и требует представителей от всех аспектов доставки. Некоторые роли, которые иногда забываются:

1.Продукт-менеджер

2.QA

3. Исполнительный спонсор

4 Архитектор

5. Девопс.

Завершенная карта потока значений: у вас должна быть карта потока значений, которая четко идентифицирует верхние одно или два узких места или ограничения в вашем текущем рабочем процессе.

Вам понадобится большая доска (или виртуальный эквивалент, такой как Miro), липкие заметки и маркеры. Вам также понадобится шаблон для матрицы удара / усилия, показанный на шаге 3 ниже.

(Необязательно) Идентифицированный профиль команды: Подготовьте идентифицированный профиль команды к отображению. См. Идентификацию вашего профиля команды для дополнительного руководства.

Роль исполнительного спонсора в семинаре

Хотя участие исполнительного спонсора в полном 90-минутном семинаре может эффективно сигнализировать о бай-ин, их присутствие также может быть блокировщиком для открытого, честного разговора.Динамика мощности в некоторых случаях препятствует психологической безопасности, необходимой для эффективного мозгового штурма и расстановки приоритетов.

Исполнительный спонсор должен:

Откажитесь от семинара: Присоединяйтесь в течение первых 10 минут, чтобы объяснить, почему упражнение является приоритетным, и четко сформулируйте свою поддержку улучшений, которые идентифицирует команда.

После введения спонсор должен уйти, позволяя команде свободно переходить к различным и конвергентным этапам мышления мастерской.

Возвращайтесь в мастерскую на последние 15 минут, так как команда представляет свои «быстрые победы» и приоритетный «первый шаг». Спонсор должен укрепить свою приверженность и предложить помощь в устранении любых потенциальных блокировщиков плана команды.



План семинара на 90 минут

Шаг 1: Настройка контекста и идентификация проблемы (10 минут)

Цель этого шага - обеспечить, чтобы все начинали с одной и той же общей реальности. Наиболее эффективные семинары делают это путем демократизации обмена и принятия решений. Используйте такие действия, как анонимное голосование и таймбоксинг, чтобы защититься от сильных голосов, предвзятых к разговорам.

1. (Необязательно) Просмотрите профиль команды (5 минут): попросите команду привести, чтобы представить идентифицированный профиль команды.

   Проанализируйте любые точки релевантного разговора, которые возникли во время деятельности по идентификации команды. Цель состоит в том, чтобы обеспечить эмпатию и общий контекст вокруг результатов команды.

2. Обзор выводов карты потока значений (5 мин): Представить карту потока значений команды; просмотреть любые точки, которые нуждаются в ясности. Направьте непосредственно на одну или две самые большие точки трения, которые были ранее идентифицированы (например, «ждите времени для проверки кода» или «ручное тестирование»). Укажите четко: «Для этого семинара мы собираемся сосредоточиться на решении этого конкретного узкого места».

Шаг 2: Инициативы мозгового штурма (20-40 минут)

Следующим шагом эффективных семинаров является генерация как можно большего количества идей. Ваша цель как фасилитатора — собрать как можно больше идей, потому что иногда лучшие приходят из наименее ожидаемых мест.

1. Поколение идей (5 минут каждое узкое место): Для каждого из этих выбранных узкого места состояния: «Я хочу, чтобы вы подумали о конкретных, действенных вещах, которые мы могли бы сделать, чтобы уменьшить это узкое место. Напишите каждую идею на одной липкой ноте».

• Поощрять все идеи. На данном этапе речь идет о количестве, а не о качестве.

• Напишите каждую идею на своей липкой ноте.

• Пример:

  Определенное ограничение: пересмотр кода является узким местом.

  Идеи, основанные на мозговом штурме: «Внедрить рецензент пользовательской истории ИИ-агента», «Внедрить проверку размера PR», «Провести тренировку по нарезке истории», «Установить командную норму 4-часового обзора», «Попробовать спаривание на всех сложных изменениях».

2. Обзор и группа (10-15 мин. каждое узкое место): Обходя по комнате, попросите одного члена команды за раз прочитать одну липкую (15 секунд). Спросите команду, есть ли у них похожая идея. Постарайтесь сгруппировать все подобные идеи в одну липкую.

Шаг 3: Беспощадно расставлять приоритеты с помощью отображения ударов / усилий (30 минут)

Скорее всего, у вас слишком много идей без консенсуса о том, с чего начать. Чтобы сохранить концентрацию команды, вам нужно безжалостно расставлять приоритеты по всем идеям и позволить лучшим подняться на вершину. Вы будете строить это общее понимание и выравнивание между командой с конвергентной деятельностью по расстановке приоритетов 2x2.

1. Настройте упражнение: Создайте единую ось и метки, как показано на рисунке 30.

Ось усилий



Примечание: Ось усилий выглядит обратной, но это нарочно, так как мы хотим расставить приоритеты для изменений с низким усилием. Рисунок 30: Ось усилий

2.Начните с чего-нибудь: случайным образом выберите идею и поместите ее в прямой центр.
3. Расставьте приоритеты и обсудите усилия для каждой идеи: возьмите по одной липкой ноте за раз и, как команда, решите, где расположить ее по отношению к другим липким на доске. При обсуждении усилий иногда помогает думать об этой оси как о сложности, а не о оценках времени.

Ось усилий с первоначальными идеями

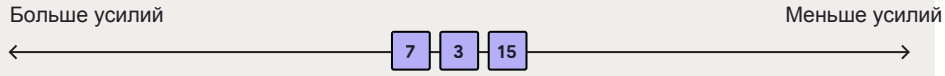
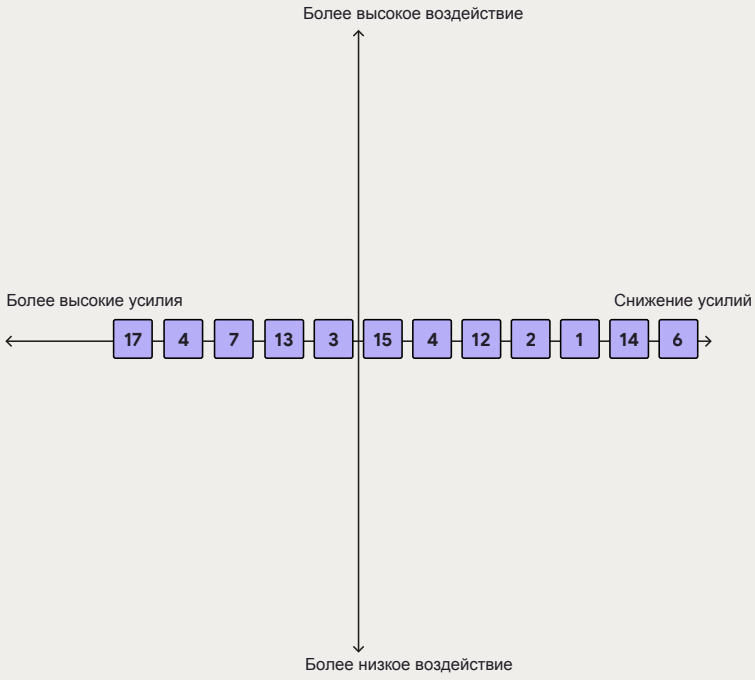


Рисунок 31: Ось усилий с первоначальными идеями

4. Добавить ось удара: создать перпендикулярную ось и метки, как показано на рисунке 32.

Добавить ось удара



Совет: Не позволяйте какой-либо липкой ноте оседлать ось удара. Рисунок 32: Добавить ось удара

5. Сосредоточьтесь на воздействии: Обсудите потенциальное воздействие каждой идеи. Переместите каждую идею вдоль оси воздействия, позиционируя ее относительно других. Повторяйте, пока каждая липкая нота не будет размещена на сетке.

Все идеи отображаются в соответствии с усилиями и воздействием

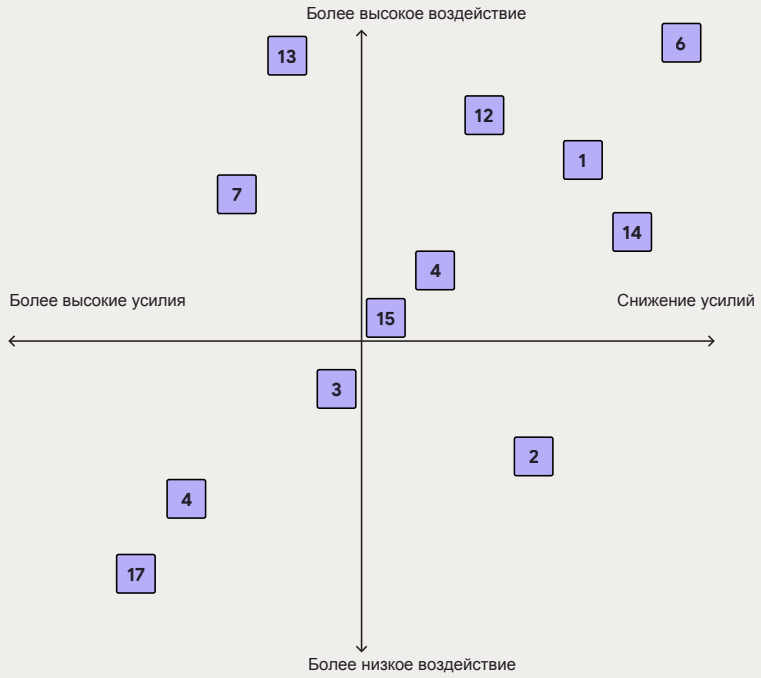


Рисунок 33: Все идеи отображаются в соответствии с усилиями и воздействием

6. Оцените свою дорожную карту и определите свои «быстрые победы»: Одним из преимуществ матрицы приоритетов 2x2 является то, что теперь у вас есть стратегическая дорожная карта, в которой есть согласие и консенсус от команды на основе расположения квадрантов:

Более высокое воздействие, более низкое усилие: этот квадрант — ваши «быстрые победы»; сделайте их сейчас. Более высокое воздействие, более высокие усилия: это квадрант «делай следующее». Для решения этих проблем потребуется больше времени, усилий и размышлений.

Низкое воздействие, меньшее усилие: это квадрант «сделай это позже».

Низкое воздействие, более высокое усилие: не делайте этого.Если идея становится более высоким воздействием или более низким усилием, в какой-то момент она переместится в более высокий сектор.

Порядок приоритетности идей

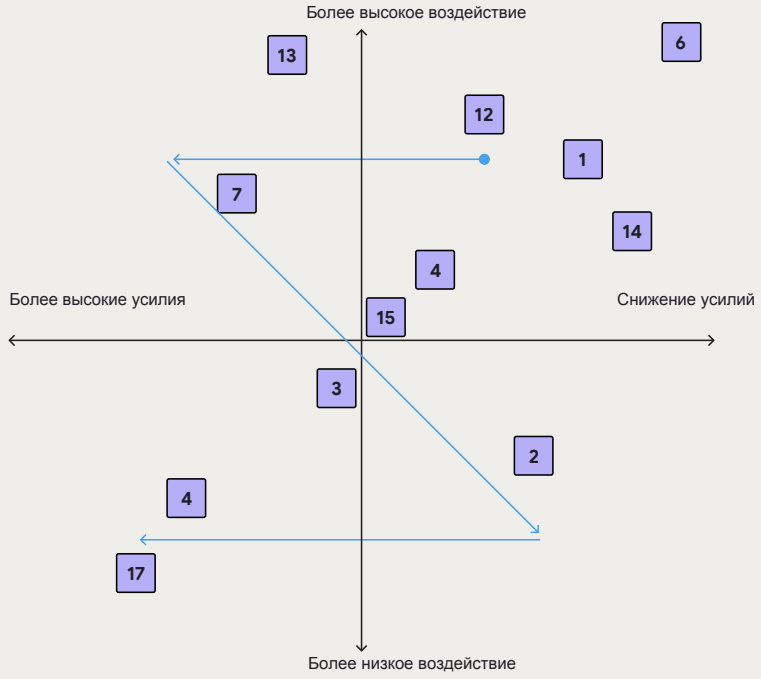


Рисунок 34: Порядок приоритетности идей

Шаг 4: Владение «первым шагом» (10 минут)

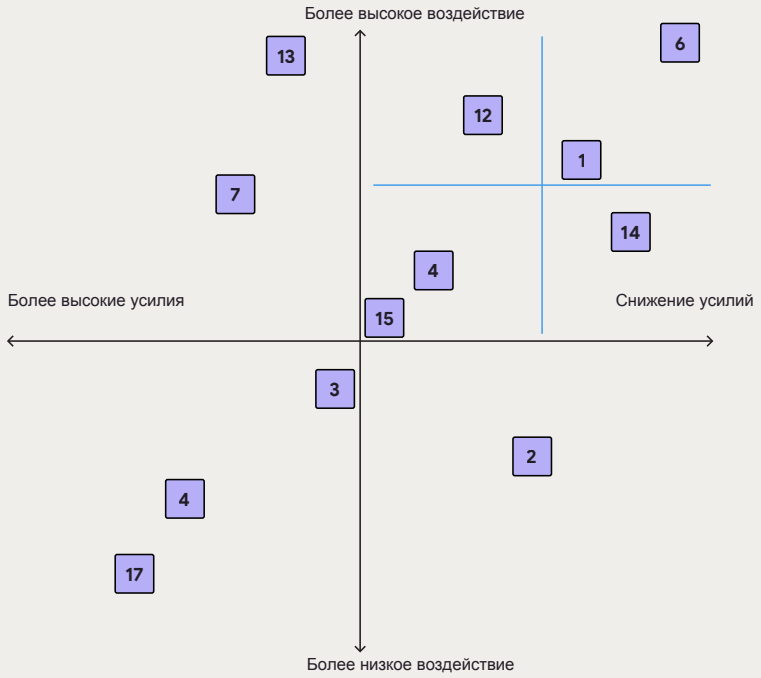
Для этого необходимо определить первую или две идеи и их владельцев.

1. Обсудить и согласовать только одну или две инициативы. Избегать ловушки чрезмерной приверженности; реальные изменения происходят от завершения одной вещи, а не от того, что несколько вещей остались незавершенными.

2.Важно идентифицировать кого-то в команде, чтобы «владеть» каждой идеей.Имейте в виду, что владелец идеи несет ответственность за продвижение идеи к завершению; они не обязаны выполнять идею самостоятельно.

Команда должна покинуть семинар с четким, действенным и коллективным планом их следующего улучшения. Каждое согласованное действие должно иметь четкого владельца и цели для завершения и критерии успеха.

Новые оси, втянутые в квадрант с высоким ударом и низким усилием.

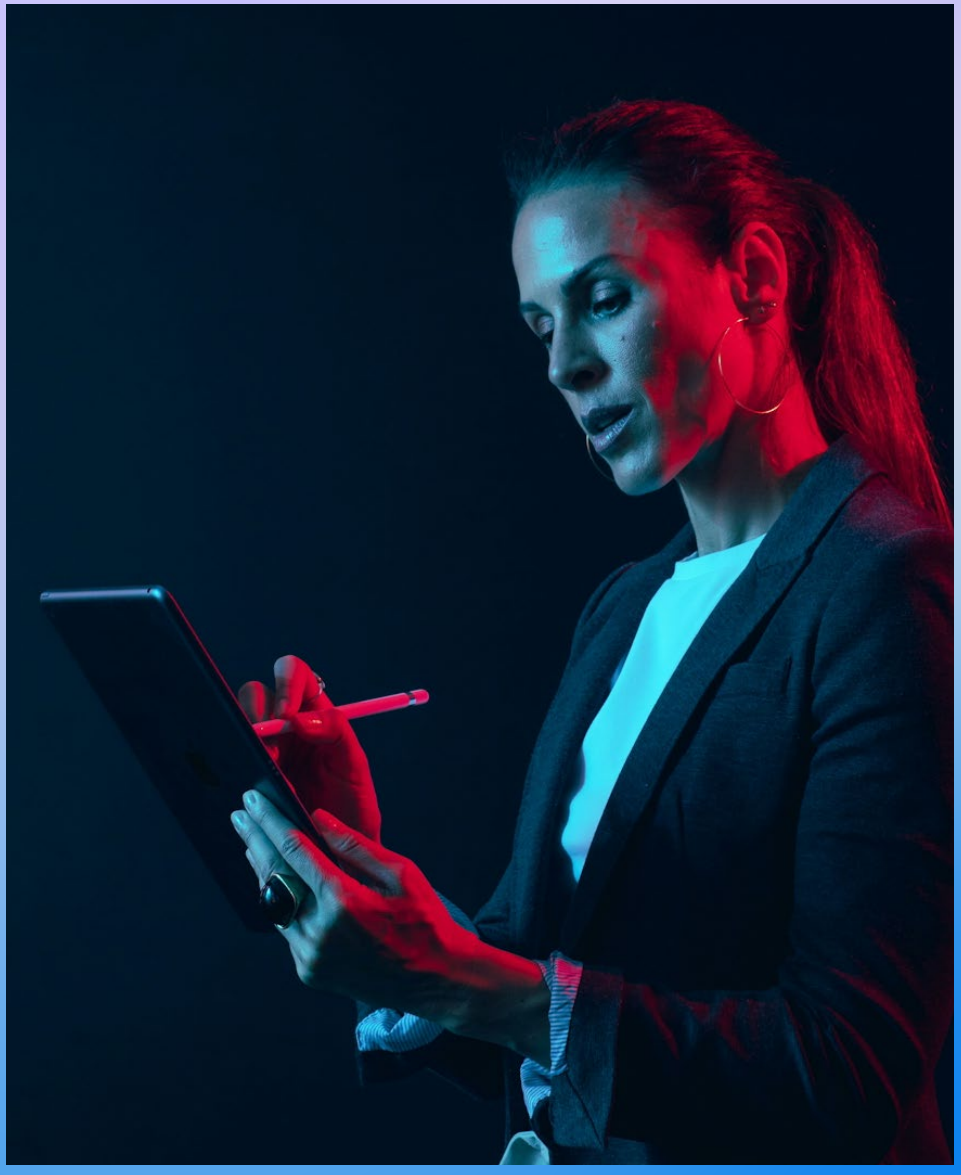


Совет: Иногда команда получает слишком много «быстрых побед». Рассмотрите возможность перемещения осей или рисования нового 2x2 в этом квадранте, чтобы вынудить к новой расстановке приоритетов. Рисунок 35: Новые оси, нарисованные в квадранте с высоким ударом и низким усилием

Заключение. Дорожная карта внедрения ИИ

Дэйв Стэнк

Специалист по работе с клиентами Google Cloud



Для подавляющего большинства технологических организаций ключевые вопросы, связанные с использованием ИИ, больше не являются, если или когда — на них был дан ответ: да, и теперь. Поэтому мы обращаемся к более тонкому вопросу о том, как. Исследования DORA и других, а также коллективный опыт многих тысяч инженеров, показали, что ИИ сам по себе не гарантированно полезен. Он должен быть принят с осторожностью.

Для того, чтобы получить отдачу от инвестиций, сделанных в приобретение и адаптацию к ИИ, команды должны также следить за тем, как они общаются, сотрудничают и работают в более широком социально-техническом контексте.

Исследование DORA выявило семь ключевых предикторов успеха с ИИ, которые представлены в модели возможностей DORA ИИ. Эта модель работает совместно с установленной моделью DORA Core Model.1

Инвестируя в основные возможности, которые управляют всеми высокопроизводительными командами, а затем сосредотачиваясь на этих семи специфических возможностях ИИ, организации могут создать оптимальную среду для усиления преимуществ ИИ.

Используйте эту модель в качестве отправной точки и ориентира, продолжая свой уникальный путь непрерывного совершенствования.

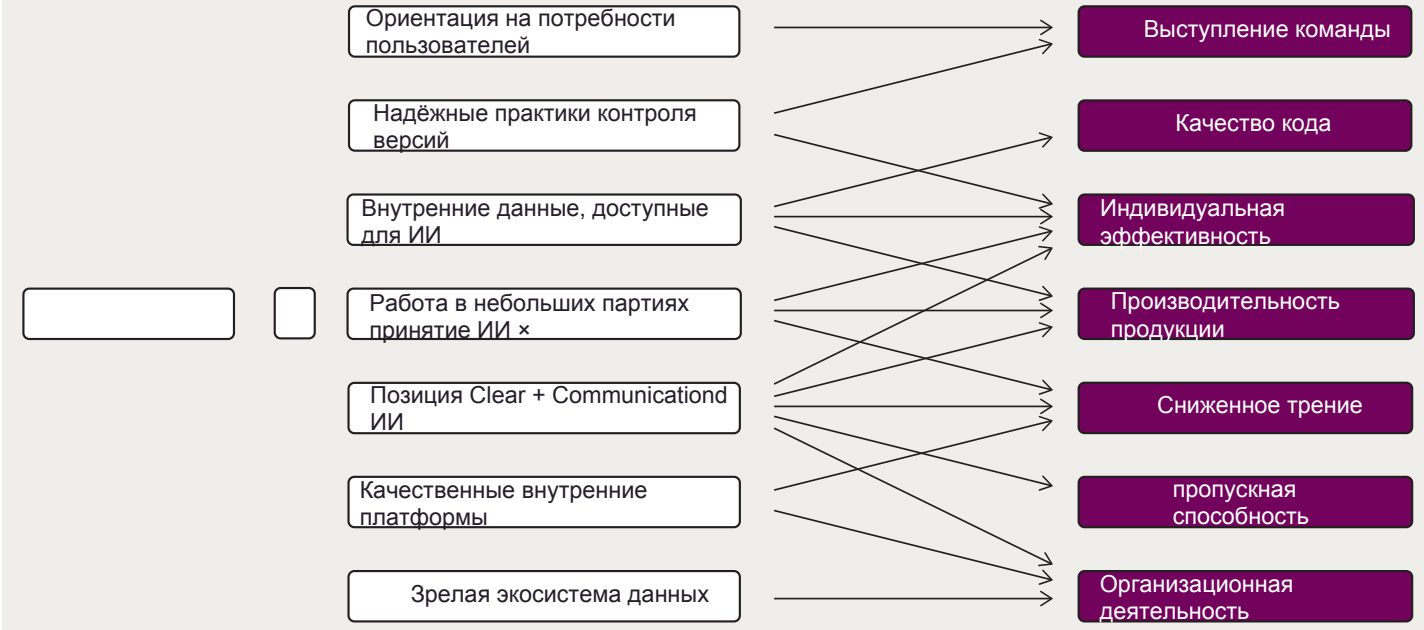


Рисунок 36: Модель возможностей DORA ИИ

Оценка вашего текущего состояния

Путь улучшения начинается с установления базового уровня. DORA использует как количественные, так и качественные методы исследования, и вы можете использовать оба в своей организации. Используя модель возможностей DORA ИИ, определите свое текущее знание каждой способности. Вы можете провести внутреннее обследование с теми же вопросами2, которые используются при разработке модели.

Кроме того, профили команд, полученные в результате кластерного анализа ответов на опросы DORA, предоставляют архетипы, описывающие большинство современных команд.

Проконсультируйтесь с этими профилями и найдите тот, который наиболее точно соответствует вашему опыту. Проведите упражнение по отображению потока значений (VSM), чтобы визуализировать конкретный поток ценности вашей команды от идеи до конечного пользователя и найти возможности для улучшения.

Качественные данные из интервью, наблюдений и открытых вопросов опроса могут обогатить ваше понимание не только статуса команды по известным измерениям, но и предложить ранее неизвестные области исследования: ответы на вопросы, которые вы не думали задавать.

Определение приоритетов

После установления базового уровня, установите приоритеты для улучшения. Они формируют вашу дорожную карту, упорядоченную - самое главное, первый пункт будет первым экспериментом, который вы попробуете. Рекомендуется многомерная рубрика приоритетов и соображения могут включать:

Естественно, что возможности, которые, как ожидается, окажут наибольшее влияние на ключевые результаты (такие как производительность или удовлетворенность пользователей), будут иметь приоритет.

В настоящее время каждая команда имеет свой собственный профиль производительности во всех возможностях ИИ. Подумайте о том, чтобы начать работу в области, где команда в настоящее время не работает, поскольку она может предложить «низкие плоды» и быстрое улучшение.

В то время как команды должны поощряться к преодолению организационных барьеров и влиянию на изменения через границы команды, также разумно признать ограничения. Лучше всего начать с локальной оптимизации и продемонстрировать успех, прежде чем пытаться осуществить более широкие изменения.

Субъективные факторы могут иметь и другую динамику. Например, конкретный проект может иметь уникальную привлекательность для людей в вашей команде: чей-то любимый проект. Если это мотивирует их инвестировать в него, это с большей вероятностью приведет к успеху.

Используйте данные опросов, метрики и беседы, чтобы составить полную картину возможностей для улучшения и постоянно пересматривать приоритеты по мере внесения изменений и оценки их влияния.

Начальные точки

Ваша дорожная карта и проекты, которые она охватывает, будут уникальными для вашей организации. В качестве отправной точки, вот несколько практических советов, основанных на семи возможностях DORA ИИ:

Чёткая и доведённая до сотрудников позиция по ИИ



Неоднозначность вокруг ИИ подавляет принятие и создает риск. Установить и социализировать четкую политику в отношении разрешенных инструментов и использования для создания доверия разработчиков. Эта ясность обеспечивает психологическую безопасность, необходимую для эффективного экспериментирования, снижения трений и усиления положительного влияния ИИ на индивидуальную эффективность и организационную эффективность.

Зрелая экосистема данных



Преимущества ИИ в организационной эффективности значительно усиливаются здоровой экосистемой данных. Инвестируйте в качество, доступность и унификацию ваших внутренних источников данных. Когда ваши инструменты ИИ могут учиться на высококачественных внутренних данных, их ценность для вашей организации возрастает.

Внутренние данные, доступные для ИИ



Подключите свои инструменты ИИ к внутренним системам, чтобы выйти за рамки общей помощи и разблокировать повышение индивидуальной эффективности и качества кода. Это означает выход за рамки простого приобретения лицензий и инвестирования инженерных усилий, чтобы дать вашим инструментам ИИ безопасный доступ к внутренней документации, кодовым базам и другим источникам данных. Это обеспечивает специфический для компании контекст, необходимый для того, чтобы инструменты были максимально эффективными.

Ориентация на потребности пользователей



Разработчики могут испытывать значительное повышение своей личной эффективности, когда они принимают ИИ. Однако, если потребности их пользователей не являются их фокусом, они могут быстро двигаться в неправильном направлении. Мы обнаружили, что принятие инструментов разработки с помощью ИИ может нанести вред командам, которые не имеют ориентированного на пользователя фокуса. И наоборот, сохранение потребностей пользователей в качестве продукта North Star может направлять разработчиков с помощью ИИ к соответствующим целям и оказывает исключительно сильное положительное влияние на производительность команд, использующих ИИ.

Надёжные практики контроля версий



Кодирование с помощью ИИ может увеличить объем и скорость изменений, что также может привести к большей нестабильности. Ваша система управления версиями является критической сетью безопасности. Поощряйте команды стать высококвалифицированными в использовании функций отката и возврата, поскольку эта практика связана с лучшей производительностью команды в среде с помощью ИИ.

Работа небольшими порциями



В то время как ИИ может повысить восприятие индивидуальной эффективности, генерируя большое количество кода, наши результаты показывают, что это не обязательно самый важный показатель. Вместо этого сосредоточьтесь на результатах. Обеспечьте дисциплину работы небольшими партиями, что улучшает производительность продукта и уменьшает трение для команд, работающих с ИИ.

Качественные внутренние платформы



Качественная внутренняя платформа является ключевым фактором для увеличения положительного влияния ИИ на организационную эффективность. Эти платформы обеспечивают необходимые ограждения и общие возможности, которые позволяют преимуществам ИИ эффективно и безопасно масштабироваться в организации.

Развитие культуры непрерывного совершенствования

Непрерывное совершенствование — это не деятельность, это мышление. Это целенаправленная практика долгосрочного роста и обучения, посредством программы итеративного экспериментирования, всегда на службе у Северной Звезды пользовательской ценности. Вот некоторые методы развития этой культуры:

Празднуйте прогресс, а не достижение

Каждая команда в вашей организации имеет разную отправную точку и столкнется с различными препятствиями. Прогресс будет варьироваться соответственно. Вместо того, чтобы объявлять одну планку успеха («все размеры партий должны быть менее 100 строк кода»), выравнивайте то, что «лучше» («предпочитайте меньшие размеры партий») и позволяйте каждой команде улучшать свою текущую ситуацию. Относитесь к каждому увеличению прогресса как к победе и побуждайте команды всегда стремиться к большему. Борющиеся команды могут все еще иметь долгий путь - массивный размер партии, сокращенный пополам, все еще велик - но они могут получить долю похвалы за свои усилия, что мотивирует постоянное улучшение. И высокопроизводительные команды должны поощряться продолжать нажимать конверт.

Примите неудачу

ИИ меняет роль разработчика, от того, чтобы быть в первую очередь непосредственным автором кода, к стратегу и оркестратору более высокого уровня. В этой роли любопытство, адаптация и обучение важнее, чем когда-либо прежде. Люди учатся, пытаясь, и терпя неудачу, и пытаясь снова. Облегчают инновации, позволяя разработчикам потерпеть неудачу, безопасно. Это может включать в себя организационные конструкции, такие как хакатоны, а также технические системы, такие как среда песочницы и доступ к новым инструментам. Самое главное, сделать обучение от неудачи чем-то, что вознаграждается и стимулируется: обнимая неудачу, разработчики отговариваются от сокрытия ошибок и поощряются делиться новыми идеями.

Использование сообществ практики

Вы не можете сделать это в одиночку, и вам не нужно. В то время как путешествие каждой команды уникально, проблемы, с которыми они сталкиваются, и стратегии их преодоления часто универсальны. В рамках сообщества практики 3 вы можете праздновать успехи и неудачи, находить общее дело и продолжать опираться на коллективные знания отрасли. Используйте структуры, предоставленные в вашей организации, чтобы создать внутреннюю группу для обмена информацией и присоединиться к глобальному сообществу DORA4, чтобы общаться с людьми из каждой отрасли по всему миру.



Выводы

Чтобы реализовать потенциал ИИ, помните, что его основная роль заключается в усилителе, увеличивающем как сильные стороны, так и дисфункции в вашей организации. Поэтому сосредоточьтесь на инвестировании в базовую техническую и культурную среду и культивировании ключевых возможностей, таких как те, которые используются в модели ИИ Capabilities.

Рассматривая внедрение ИИ как организационную трансформацию и устранение системных ограничений, вы можете направить силу ИИ на решение реальных проблем клиентов и достижение устойчивого успеха.

Стань лучше, стань лучше.

1. «Исследовательская программа DORA». https://dora.dev/research
2. «Исследование DORA: модель возможностей DORA ИИ». https://dora.dev/ai/capabilities-model/questions
3 «Как изменить свою организацию». https://dora.dev/guides/how-to-transform/#build-community-structures-to-spread-knowledge
4 «Общество ДОРЫ». https://dora.community

Благодарности

Этот отчет является дополнением к отчету о разработке программного обеспечения с использованием ИИ к 2025 году.1 Мы рекомендуем вам обратиться к нему для получения дополнительной информации о нашей методологии, моделях и многом другом.

Вклад авторов и редакторов

Амир Аббас

Джеймс Брукбанк

Дерек ДеБеллис

Роб Эдвардс

Нейтен Харви

Вивиан Ху

Бен Хосе

Аманда Льюис

Эрик Максвелл

Эллисон Парк

Джером Симмс

Дэйв Стэнк

Кевин М. Сторер, доктор философии.

Люсия Субатин

Сет Розенблатт

Седрик Яо

Точность имеет значение2

Партнеры по исследованиям и дизайну

Оригинальное название: 3 DORA Branding

Human After All: 4 DORA: дизайн

Продуктивность: 5 Поддержка исследовательской инфраструктуры

User Research International: 6 Поддержка исследовательской инфраструктуры

1. Состояние разработки программного обеспечения с помощью ИИ. https://dora.dev/dora-report-2025
2.Точность имеет значение. https://accuracymatters.co.uk
3. Мы видимся. https://apparent.com.au
4 Человеческий фактор: ясность через творчество. https://www.humanafterall.studio
5.Плодотворный | Легко собирать высококачественные данные от реальных людей. https://www.prolific.com
6. User Research for Product Development | User Research International. https://www.uriux.com

Проверьте, есть ли у вас последняя версия этого отчета.

Визит https://dora.dev/vc/aicm/?v=2025.1 Чтобы узнать, есть ли более новая версия этого отчета.

Модель возможностей DORA ИИ от Google LLC лицензирована в соответствии с CC BY-NC-SA 4.0.



Совершенствуйте подход к улучшениям. dora.dev