



AI-Disrupt PDLC

Стратегия AI-трансформации бизнеса:
от кода к намерению

WHITEPAPER

Кирилл Меньшов

старший вице-президент, руководитель блока
«Технологии» Сбера, автор AI-Disrupt PDLC

Май 2026



**Полная версия документа
доступна по ссылке**

Права на использование контента. Настоящим уведомляем вас о том, что это документ или любая его часть не предназначены для копирования, распространения или тиражирования любыми способами без предварительного письменного разрешения ПАО «Сбербанк». При отсылке к данным исследования упоминание автора обязательно. Этот документ был подготовлен исключительно в целях информации. ПАО «Сбербанк» не гарантирует точности и полноты всех сведений, содержащихся в документе. Информация, представленная в этом отчете, не должна быть прямо или косвенно истолкована как информация, содержащая рекомендации по дальнейшим действиям по ведению бизнеса. Все мнения и оценки, содержащиеся в данном исследовании, отражают мнение автора на день публикации и могут быть изменены без предупреждения. ПАО «Сбербанк» не несет ответственности за какие-либо убытки или ущерб, возникшие в результате использования любой третьей стороной информации, содержащейся в данном исследовании, включая опубликованные мнения или заключения, а также за последствия, вызванные неполнотой представленной информации. Задачи и выводы в настоящем отчете, являются общими и не могут рассматриваться как комплексное исследование IT- рынка того или иного товара или услуги, а также трендов.

Содержание

1. Введение и стратегическое видение	5
1.1 Стратегическое понимание текущей ситуации	5
1.2 AI-native cycle и цена бездействия	6
1.3 Архитектурный выбор 2026: окружение агента важнее модели	7
1.4 Критический энейблер: Context Engineering	7
1.5 Человек как центр системы	8
2. Эволюция разработки: преодоление рыночных вызовов	9
2.1 Почему именно Сбер	9
2.2 AI-Disrupt PDLC vs Agile: масштаб изменений	9
2.3 Ограничения «AI как ассистент»: скрытый технический долг	10
2.4 Двухпетлевая модель: Intent Loop и Implementation Loop	10
2.5 Developer Experience и устранение когнитивной нагрузки	12
2.6 Tiny Teams и масштабируемость	12
3. Техническая архитектура: от концепции к реализации	13
3.1 Integrated Development Platform: точка пересечения двух петель	13
3.2 Ключевые функциональные возможности платформы	14
3.3 Context Engineering и Context Supply Chain	15
3.4 Перевод разработки в автономный режим: Specification-Driven Development, Policy as Code, цифровой двойник, Agent Development Lifecycle	16
3.5 Golden Path: автоматическая прослеживаемость от требования до деплоя	18
3.6 Agent Marketplace и развитие платформы «снизу-вверх»	18
3.7 Архитектурные принципы зрелой IDP	19
4. Безопасность, управление и новые роли в эпоху AI	20
4.1 Enterprise-готовность: где заканчиваются эксперименты и начинается ответственность	20
4.2 Harness — окружение агентной среды	21
4.3 Адаптивная лестница автономии агента	21
4.4 Agentic FinOps и Cost Circuit Breakers	22
4.5 Эволюция ролей: трансформация человека, а не его замещение	22
5. План реализации и стратегические рекомендации	23
5.1 AI-трансформация не может быть спланирована заранее	23
5.2 Vibe Working как управленческая философия	24
5.3 Горизонты трансформации	24
5.4 Культура экспериментов	25
5.5 Заключительное слово	26

AI-нативная трансформация процесса создания программного обеспечения

Краткий обзор

В статье представлена методология AI-Disrupt PDLC, которую старший вице-президент Сбера Кирилл Меньшов вместе с командой активно внедряют в Сбере.

Документ описывает стратегию AI-трансформации организации, которая кардинальным образом меняет природу создания цифровых решений. Мы видим, что AI — это не очередной инструмент, а новая природа организации, которая меняет ежедневный ритм команд, устоявшиеся процессы и инструменты.

Благодаря AI мы можем запустить продукты, которые ещё вчера казались сочетанием взаимоисключающих требований:

- многофункциональные и при этом максимально понятные
- точно соответствующие потребностям клиента и не требующие от него лишних действий

Мы предлагаем фреймворк AI-Disrupt PDLC как стратегический ответ на вызов:

«как перестроить организацию вокруг принципиально новой технологии разработки так, чтобы получить конечный бизнес-результат»

Если раньше человек учился разговаривать на языке машин, перекладывая свои намерения в код, то AI-трансформация позволяет сосредоточиться именно на намерении, на поиске ответов на вопросы: **что нужно сделать, для кого это нужно сделать.**

Исполнение становится практически моментальным, масштабируемым – единственным дефицитным ресурсом остаётся намерение.

Стратегическое преимущество получают те организации, которые переосмысливают весь жизненный цикл разработки, а не ограничатся встраиванием AI-ассистентов поверх существующих процессов.

Документ ориентирован на руководителей высшего звена — CEO, CTO, CIO, CISO, CAIO и других лидеров цифровой трансформации.

AI-Disrupt PDLC позволит получить:

- ✓ **Кратный рост продуктивности бизнеса** за счёт новых AI-нативных подходов к разработке (уже подтверждается бенчмарками технологических лидеров)
- ✓ **Обеспечение долгосрочного технологического преимущества** через развитие собственной платформы и детализацию корпоративного контекста
- ✓ **Высвобождение творческого потенциала инженеров** за счёт делегирования рутины агентным системам
- ✓ **Создание масштабируемых корпоративных стандартов и платформенных практик**, безопасных для применения AI в регулируемой среде

Документ состоит из пяти разделов:

- 1 Определяет стратегическое видение и архитектурную развилку 2026 года — приоритет среды исполнения (harness) над моделью.
- 2 Показывает эволюцию разработки и двухпетлевую модель Intent Loop / Implementation Loop.
- 3 Детально описывает техническую архитектуру Integrated Development Platform — фундамента трансформации
- 4 Раскрывает контур безопасности, управления и трансформации ролей.
- 5 Задаёт дорожную карту, стратегические рекомендации и заключительное слово.

AI-трансформация разработки — это не модернизация инструментов, а новый взгляд на процессы, роли и платформу вокруг намерения (human intent)

Мы считаем что у организаций есть 12–18 месяцев чтобы измениться. Те, кто пройдет AI-трансформацию осознанно, получают стратегическое преимущество на горизонте 2028–2030 годов и возможность создавать продукты совершенно нового уровня. Остальные будут двигаться вдогонку — в условиях, когда конкуренты уже задают новые отраслевые стандарты.

01

Введение и стратегическое видение

1.1 Стратегическое понимание текущей ситуации

Перед руководителями высшего звена сегодня стоит ключевой стратегический вопрос: как перестроить организацию вокруг принципиально новой технологии выполнения процессов так, чтобы получить максимальный бизнес-результат.

Ответ на этот вопрос определяет архитектурную и культурную позицию организации в новом отраслевом ландшафте — и на горизонте ближайших лет, и в долгосрочной перспективе.

- Сегодня значительная часть ресурсов многих компаний уходит на отслеживание новостей о новых моделях и о новых дисциплинах в сфере AI (prompt engineering, context engineering)
- Менее чем за 12 месяцев SOTA (state of the art – лучшие в своем классе)-модели лидеров рынка уступают open-source
- Цикл релизов моделей сократился с 1-2 лет до нескольких месяцев
- Различия между ведущими AI-лабораториями и open-source сокращается: open-source модели все чаще оказываются достаточно хороши для большинства задач

Текущий подход «AI-ассистент», построенный на простых моделях и точечной генерации кода/текста, обеспечивает локальный прирост производительности — 10–15% по внутренним наблюдениям Сбера и 11–25% по данным AWS и OpenAI. Это верхняя граница отдачи, преодоление которой невозможно без перестройки процессов. Мы стремимся к результату принципиально иного порядка — за пределами этого потолка, — и поэтому собственная платформа разработки и корпоративный контекст становятся для нас приоритетом номер один.

По данным DORA 2025 (Google Cloud, ~5 000 респондентов), около 90% разработчиков уже используют AI-инструменты, однако прирост производительности неравномерен и зависит от зрелости организации.

По доступным отраслевым оценкам, улучшение продуктивности разработчиков может достигать **35–45%**, а при комплексной интеграции AI в 10+ сценариев разработки **66%** компаний фиксируют рост скорости выхода на рынок — против лишь 35% при точечном применении.

Основной эффект от AI достигается только при системном внедрении. Эффективность подхода подтверждена мировой практикой передовых технологических компаний и провалидирована на пилотных внедрениях в Сбере.

При этом AI для нас — не технология, развиваемая из исследовательского интереса, а наиболее перспективный способ повысить эффективность организации. **Результативность измеряется прежде всего в бизнес-категориях: ценность, создаваемая для клиентов, доход и прибыль для акционеров.** Технология должна приносить прибыль через бизнес-результат — или через значительное ускорение процессов, в которых этот бизнес-результат создаётся.

1.2 AI-native cycle и цена бездействия

Сбер предлагает AI-Disrupt PDLC как **фреймворк, описывающий новую природу создания цифровых решений.**

Ранее движение от идеи до бизнес-результата прямо зависело от скорости исполнения. AI Disrupt PDLC - способ выделить исполнение в отдельную петлю чтобы **радикально его ускорить.**

Возникает двухпетлевой цикл, который мы называем **human-driven (суть — намерения) и agent-driven (суть — реализация).** Люди фокусируются на постановке задачи, валидации и результате. Агентная система воплощает намерение в технический результат — настолько быстро, насколько это возможно.

Желаемая управляемость и скорость воплощения идей напрямую зависит от готовности перестраивать процессы. Работы с одними лишь инструментами больше недостаточно.

Организации, ограничившиеся встраиванием Copilot-инструментов в существующие процессы, не получают возможности фундаментально переосмыслить и трансформировать свои продукты. Они проиграют тем, кто изменят организацию целиком.

>50%



Традиционных рабочих мест разработчиков **заместит роль Product Engineer** к 2030 году

80%



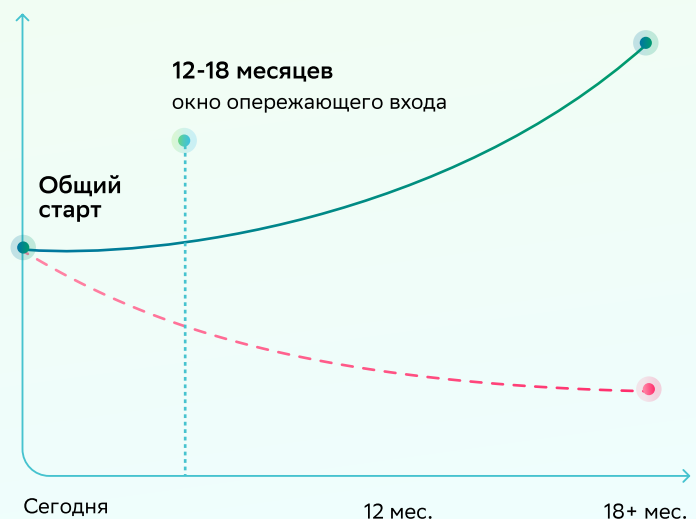
Организаций перейдут от больших инженерных команд к **компактным, усиленным AI**

Отраслевые прогнозы недвусмысленны: организации без Specification-Driven Development (разработки на базе спецификаций) к 2027 году будут **отставать на 25%** по скорости разработки.

Это случится не вследствие деградации самих организаций, а вследствие ускорения конкурентов.

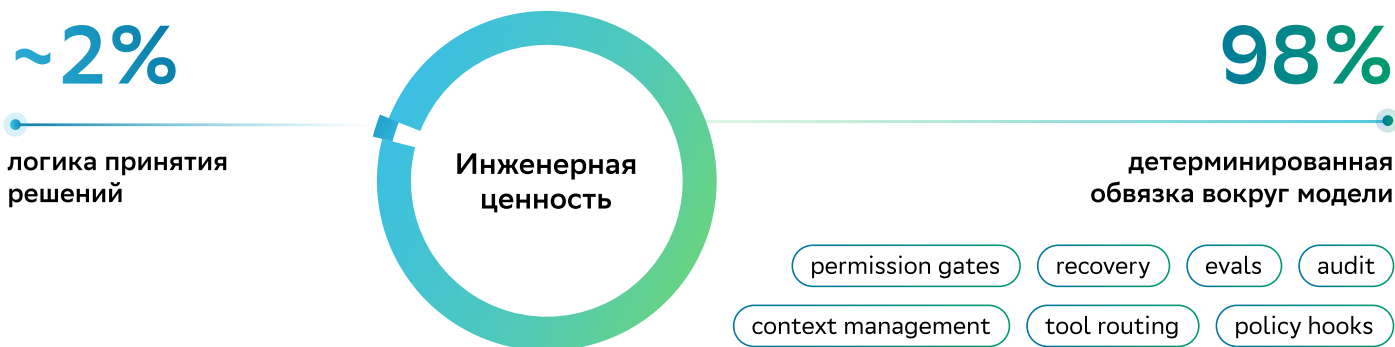
Конкурентоспособность

- **AI-нативная организация**
Стратегическое преимущество
- - - **Только Copilot**
-25% по скорости (2027)



1.3 Архитектурный выбор 2026: окружение агента важнее модели

Среда исполнения имеет доминирующее значение в сравнении с моделью. Реверс-инжиниринг реализованных агентных систем 2026 года показывает:



Из этого следует фундаментальный стратегический вывод:

Организации, концентрирующие усилия на «выборе правильной модели» или «добавлении Copilot в IDE», окажутся в проигрышной стратегической позиции. Организации, строящие собственную интегрированную среду управления агентами и разработкой (agent runtime / Integrated Development Platform) со встроенными механизмами управления агентами, формируют компетенцию, независимую от модельного цикла.

В эпоху стремительной смены поколений нейросетей главным активом становится не сама модель, а детерминированная инфраструктура вокруг неё. Именно она (harness), а не LLM, становится ключевым корпоративным активом новой эпохи инженерной продуктивности.

Это формирует ряд практических выводов:

- ✓ Усилия, направленные на развитие harness, накапливаются и сохраняются при смене модели
- ✓ Работа, заточенная под конкретную модельную интеграцию, обесценивается с каждым новым поколением моделей
- ✓ Независимая от модели архитектура устраняет привязку к вендору и превращает внутреннюю платформу в долгосрочный корпоративный актив
- ✓ Harness, а не модель, является источником воспроизводимости, аудируемости и предсказуемости поведения системы
- ✓ Зрелость harness прямо определяет, какой уровень автономии агентов организация может позволить себе без потери управляемости

1.4 Критический энейблер: Context Engineering

Сама по себе LLM-модель ещё не является полноценным инструментом. Качество и предсказуемость поведения системы «модель + harness» критически зависит от качества и доступности организационного контекста: знаний, данных, правил, регламентов, исторических решений.

Дисциплина Context Engineering делает эти знания, данные, правила и инструменты пригодными для надёжного рассуждения моделей и точных решений.

Контекст — критический энейблер AI-трансформации.

В отсутствии контекста агентная система восполняет недостающие сведения из результатов собственного обучения — и может рассуждать не так, как это нужно организации: исходя из обобщённых паттернов, заимствованных из открытых данных, а не из реальной архитектуры, нормативов и приоритетов компании.

Центральными идеями в context engineering становятся:

- 01 **Способность организации выбрать и сформулировать свод знаний о том, как должно функционировать исполнение.** Свод знаний — машиночитаемый документ, которым руководствуется AI-агент. Мало определенности или двусмысленность — агент сделает как понял. Точные, но широкие инструкции — залог успеха в поиске хороших, реализуемых решений.
- 02 **Процессы и инструментальный слой.** Характерные вопросы при проработке — какие слои контекста нужно выделить, кто отвечает за их актуальность, как технически распространяется контекст, как можно проверить актуальность и достоверность? Инструментальный слой обеспечивает следование контексту на всех этапах - формирования решения и проверке соответствия.

При наличии качественного контекста система опирается именно на корпоративное понимание задачи, что обеспечивает повторяемость и проверяемость результата. Именно контекст превращает мощную модель в работающий предсказуемый инструмент.

1.5 Человек как центр системы

**Мы не верим в тезис о замене человека искусственным интеллектом.
Мы говорим о трансформации и усилении роли, а не о её вытеснении.**

Именно человек формирует идеи, договаривается о том, как их воплотить, и принимает решение о том, что вообще стоит воплощать. Это мир человека. Мы создаём технологии, и технология должна работать на нас.

Исследования AI-трансформации, проводимые ведущими международными организациями, фиксируют отчётливый парадокс: **при высоком уровне вовлечённости AI в работу (порядка 60%) лишь небольшая доля задач (0–20%) делегируется агенту полностью.** Большинство инженеров работают в режиме тесного соавторства с агентной системой. При этом около четверти задач, выполняемых с участием AI, в принципе не были бы сделаны без него. Это расширение возможностей, а не замещение.

02

Эволюция разработки: преодоление рыночных вызовов

2.1 Почему именно Сбер

Мы обладаем сочетанием возможностей, которое позволяет нам действовать уже сейчас.

У нас есть промышленный масштаб инженерии — десятки тысяч разработчиков и критические нагрузки, при которых методология не может оставаться теоретической. У нас есть собственная LLM-платформа GigaChat и сопутствующая инфраструктура, возможность проводить передовые исследования в СберУниверситете и AIRI.

Благодаря этому сочетанию мы можем не просто исследовать AI как трансформирующую технологию, но и преобразовывать реальные промышленные процессы — в том числе процессы финансового сектора, к которым предъявляются одни из самых высоких требований по надёжности, безопасности и регуляторной устойчивости в индустрии.

AI-трансформация — задача, над которой сегодня работает вся отрасль. Мы не дожидаемся появления готовых внешних фреймворков: проводим собственную трансформацию уже сейчас, валидируем подходы на промышленной инженерной практике и готовы делиться накопленными знаниями.

2.2 AI-Disrupt PDLC vs Agile: масштаб изменений

AI-трансформация разработки кардинально сложнее agile-трансформации. Agile затрагивал только механизм определения намерений человека и логику распределения ресурсов разработки, тестирования, внедрения. Это улучшало качество решений, увеличивало скорость и создавало новые условия для запуска продуктов, но не меняло саму суть работы. CI/CD, DevOps, микросервисная архитектура — всё это улучшало IT-составляющую без изменения её природы.

AI-Disrupt PDLC описывает изменения принципиально иного уровня: меняется характер работы на каждой стадии, скорость выполнения этих стадий и компетенции, требуемые для исполнения. Стадии SDLC формально сохраняются, однако за каждой из них стоят другой субъект, другая скорость и иное содержание того, что относится к понятию «работа».

Темп подобных сдвигов в индустрии заметно ускоряется. Первые эксперименты с нейронными сетями появились ещё в 1960-х годах, прорыв глубокого обучения произошёл в 2010-х, а массовое распространение генеративного AI — уже в 2020-х. Каждый следующий технологический шок наступает быстрее предыдущего, и организациям нужно учиться готовиться к будущим шокам, а не догонять прошлые.

2.3 Ограничения «AI как ассистент»: скрытый технический долг

Базовые AI-помощники (Copilots) для крупного бизнеса дают локальный прирост производительности, но без изменения процессов они создают скрытый технический долг.

К 2028 году **40% предприятий**, использующих агентные инструменты, **удвоят расходы на AI-кодирование** именно из-за существующих сейчас пробелов в управлении.

Скорость без управления качеством формирует технический долг нового типа — скрытые проблемы, которые обнаруживаются уже на стадии промышленной эксплуатации. Harness и governance следует рассматривать не как накладные расходы на контроль, а как способ получить из мощной, но недетерминированной модели предсказуемую корпоративную технологию: они переводят ускорение в устойчивую и воспроизводимую модель ускорения, а не в накопление скрытого долга.

Copilots

на **35%**

быстрее создается
AI-генерируемый код

на **25%**

больше ошибок
кибербезопасности
без надлежащего
ревью

~25%

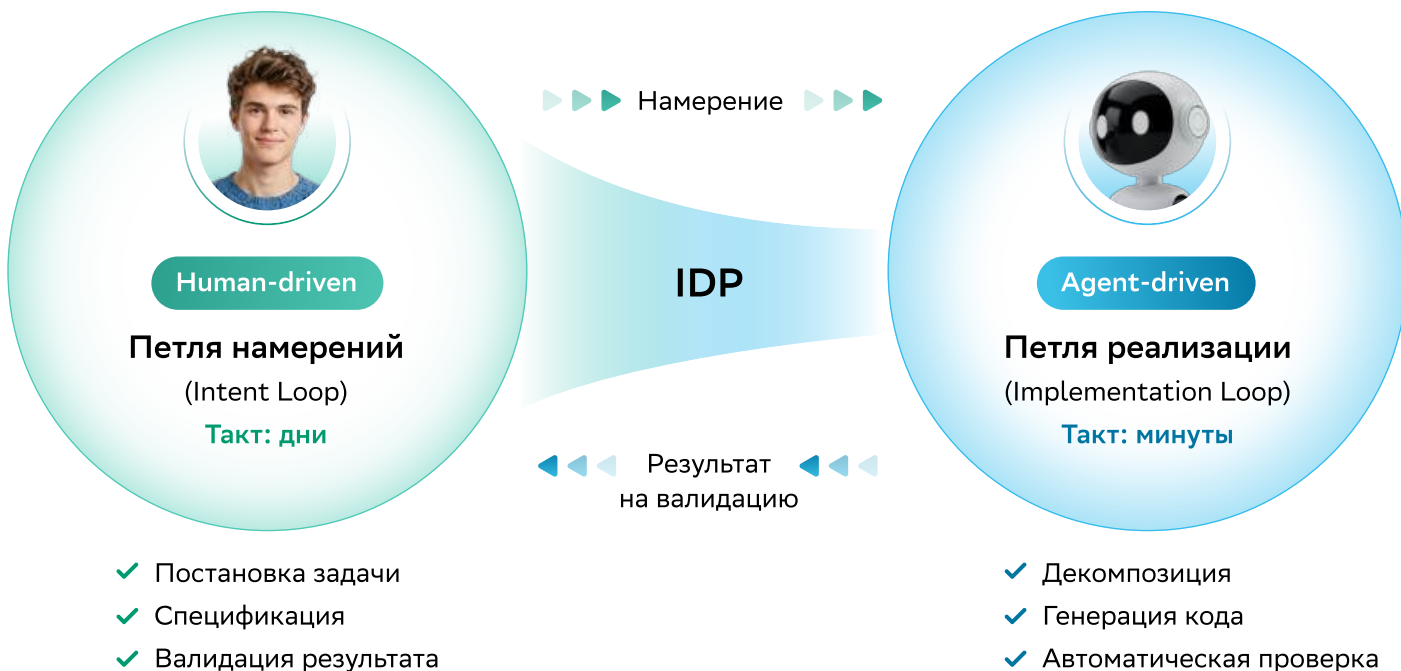
проблем сохраняется
в финальной версии

на **40%**

растет сложность
кода

2.4 Двухпетлевая модель: Intent Loop и Implementation Loop

Мы делим процесс разработки на два контура. Это не одно из возможных решений, а единственный, на наш взгляд, корректный способ описать, как меняется природа разработки и как в неё органично встраиваются агентные системы.



Петля намерений (Intent Loop) — люди определяют бизнес-правила и проектируют архитектуру. Время такта — дни. Люди фокусируются на постановке задачи, на валидации, на результате.

Петля реализации (Implementation Loop) — агентная система автономно реализует код на основе заданных спецификаций. Время такта — настолько быстро, насколько это позволяет настройка платформы и процессов организации.

Между этими петлями стоит Integrated Development Platform (IDP) — интегрированная платформа разработки. Именно она обеспечивает связность двух контуров: транслирует намерение в исполнение, возвращает результат человеку для валидации, формирует единое пространство постановки и приёмки задач, поддерживает целостность контекста, корпоративных нормативов и инструментов на стороне обоих участников. Без такой платформы две петли существуют параллельно, но не взаимодействуют между собой как часть единого процесса. Подробное описание архитектуры и функций IDP представлено в третьем разделе документа.

Принципиальная идея модели — за каждым участником закреплена своя функция.

Функция человека — формирование намерения. Функция агента — исполнение.

Человек создаёт продукт — платформа (пока) не способна замещать человека в этой функции, поскольку она спроектирована для него.

Однако, человек физически не способен вносить изменения с той скоростью, которая требуется по мере усложнения технологий. Агенты обладают этой способностью — поэтому человек усиливает собственную способность мыслить и делегирует исполнение агентной системе.

Критически важной является точка синхронизации: человек не делегирует агенту полномочия принимать архитектурные и продуктовые решения. Человек ставит задачу и принимает результат. Изменяется сама система постановки требований — она становится первичным артефактом разработки.

В результате исходный код перестаёт быть главным активом. Его место занимает спецификация (спецификация процесса, желаемого результата, поведения системы или продукта) — это формализовано в подходе Specification-Driven Development. Продукт сохраняет соответствие заданным целям, поскольку AI оперирует чётко определёнными рамками, а не реконструирует каждый раз контекст самостоятельно.

По сути, это возврат к строгой алгоритмической инженерии (новый enterprise-стандарт), но с использованием скоростей генеративного AI. Такой подход позволяет: радикально снизить технический долг; повысить прозрачность и предсказуемость разработки; безопасно масштабировать инновации без потери качества.

Желаемое состояние, к которому ведёт двухпетлевая модель, описывается термином **Zero-Friction (без сопротивления).**

Под сопротивлением понимаются ограничения, при которых скорость разработки определяется не качеством мысли, а зрелостью технологии или численностью команды. Цель двухпетлевой модели — устранить эти ограничения и оставить единственным лимитом скорости качество интеллектуальной работы команды. Zero-Friction — это способность компактных команд быстро создавать качественный продукт, концентрируя время на работе непосредственно с продуктом.

2.5 Developer Experience и устранение когнитивной нагрузки

Инновационный фокус AI-Disrupt PDLC — **Developer Experience и устранение когнитивной нагрузки.**

Функциональные задачи и роль разработчика смещаются от написания строк кода к решению содержательных проблем и созданию новых решений. Это системный сдвиг от создания к верификации. По данным DORA 2025:

45%

разработчиков уже тратят **меньше времени на написание кода**

42%

разработчиков выделяют **больше времени на ревью**

Параллельно DORA фиксирует фундаментальный принцип: AI действует как усилитель, мультиплицирующий уже сложившиеся характеристики организации.

- ✓ Сильные команды с развитыми практиками демонстрируют значительный рост эффективности
- ✓ Слабые команды — обнажают накопленные проблемы
- ✓ AI не превращает слабую команду в сильную: он усиливает существующий вектор движения
- ✓ Зрелые команды получают мультипликативный прирост скорости и качества
- ✓ Команды с накопленными процессными проблемами фиксируют их усугубление

2.6 Tiny Teams и масштабируемость

Концепция Tiny Teams — компактных команд, **состоящих из 4–5 специалистов**, — является **прямым следствием** двухпетлевой модели и состояния **Zero Friction**.

Малые группы являются естественной средой для рождения ценной мысли; на стадии исполнения уже сформулированной идеи большая численность команды тормозит развитие. Именно этот фактор устраняется агентным взаимодействием. Классическая agile-команда насчитывает 10–15 человек (two-pizza team). AI-Disrupt PDLC обеспечивает переход к составу 4–5 человек — за счёт того, что AI снимает ограничение в виде ручной работы со знаниями технологического стека и корпоративных требований. К 2030 году 80% организаций перейдут от больших инженерных команд к компактным AI-усиленным командам.

	AS IS	TO BE
Состав	 Классические команды из 10-15 специалистов с AI Copilot	 4-5 человек и агенты отвечают за E2E результат
Скорость принятия решений	Дни, недели (согласования)	Часы (прямая коммуникация)
Специализация	Узкие специалисты по ролям	Full-stack с AI -усилением

Масштабируемость в новой модели — это способность создавать большие продукты силами компактных команд без требования пропорционального найма. Запуск множества продуктов перестаёт быть ограниченным численностью людей и определяется наличием идей.

Наращивание экспертизы происходит радикально быстрее, поскольку агентные системы осваивают новые знания на скоростях, недостижимых для человека. Компактная команда получает возможность опереться на глубокую узкоспециализированную экспертизу через агентов, не расширяя штат.

03

Техническая архитектура: от концепции к реализации

Перейдем от концептуальной модели к её инженерному воплощению. Здесь раскрываются принципы построения Integrated Development Platform, работы с контекстом и спецификациями, а также организационные изменения, обеспечивающие зрелость подхода и его применимость в условиях промышленной эксплуатации, а не лабораторного эксперимента.

3.1 Integrated Development Platform: точка пересечения двух петель

Двухпетлевая схема Intent Loop / Implementation Loop соединяется в Integrated Development Platform (IDP) — интегрированной платформе разработки.

Назначение IDP — формирование точки взаимодействия и точки пересечения циклов человеческого намерения и агентного исполнения. Именно в этой точке намерение передаётся в исполнение, обратная связь возвращается человеку, формируя единый технический процесс. Концептуально IDP представляет собой пространство взаимодействия человека и агента.

Из этого следуют требования к интерфейсу:



Без IDP значительная часть усилий разработчика уходит не на решение содержательной задачи, а на ручное перекладывание корпоративного контекста в работу модели — сессию за сессией, на каждое действие. Альтернативный сценарий ещё хуже: контекст вообще отсутствует, и инструмент перестаёт соответствовать организации — создаваемое решение оказывается оторванным от её приоритетов, архитектуры и нормативов.

И в том, и в другом случае агентное кодирование вырождается в фрагментированную совокупность изолированных инструментов: различные инженеры применяют разные AI-инструменты, модели и конфигурации.

От 40 до 60% AI-взаимодействий разработчиков начинают происходить через неуправляемые каналы — феномен «shadow AI», создающий регуляторные риски и риски утечки данных. С IDP, опирающейся на корпоративный контекст, формируется единый контур, единая семантика, единый аудит.

Функциональные возможности IDP должны обеспечивать интеграцию контекста, инструментов и корпоративных систем

IDP предоставляет работоспособные инструменты для выполнения задач человеком и агентом, в том числе для работы человека в разных ролях — Product Engineer, Platform Engineer, AI Architect. Именно на базе этой точки интеграции IDP становится средой, в которой реализуется трансформация в техническом смысле. Это пространство, в котором сегодняшние аналитики, разработчики, тестировщики и администраторы систем начнут исполнять свои функции по-новому.

Дисциплина работы с корпоративным контекстом, опирающаяся на IDP, имеет ещё одно важное следствие: она увязывает работу продуктовых команд и инженерных ролей с задачами тех подразделений организации, которые отвечают за выпуск нормативных требований, надёжность и безопасность, целостность архитектуры — внутренних регуляторов компании.

В сегодняшнем ландшафте знания об этих нормативах передаются плохо, проверяются тяжело и, зачастую, вручную. Эти ограничения также часто остаются в головах ограниченного круга специалистов. AI-нативная разработка качественно меняет эту ситуацию: нормативы становятся частью корпоративного контекста, поступают в работу агента в момент решения задачи и автоматически проверяются на каждом шаге.

3.2 Ключевые функциональные возможности платформы

Платформа объединяет несколько классов возможностей, каждый из которых решает свой контур задач AI-нативной разработки.

Один из ключевых классов инструментов — мультиагентная система, которая позволяет end-to-end закрывать задачи разработки. Мультиагентность реализует принципы параллельного исполнения, производительности, масштабируемости и многофункциональности при сохранении изолированности функций и способности к координации. Изолированность нужна для отладки, качества специализации и концентрации каждого агента; синхронизация — для получения совместного результата.

Функционально мультиагентная система поддерживает разные языки программирования, особенности технологического стека, паттерны тестирования, инструменты CI и DevOps.

→ На уровне архитектуры применяется принцип изоляции через субагентов

Субагенты используются прежде всего как механизм изоляции контекста, прав и аудита, а не как способ простого увеличения числа «исполнителей». Такая идея естественным образом реализует подход *mixture of experts*: разные агенты получают разные роли, инструкции и наборы инструментов, обеспечивая глубокую специализацию там, где это требует задача.

→ Платформа выступает оператором корпоративного контекста

Получает данные от мастер-систем, фильтрует и сжимает их, доставляет агенту по принципу *just-in-time*. Это даёт согласованность данных и предотвращает галлюцинации на всех этапах работы агента.

→ Интеграция с мастер-системами и инструментами

Через стандартизированные протоколы межагентной и инструментальной коммуникации (в первую очередь *Model Context Protocol*, переданный в управление *Linux Foundation*, но также и другие, такие как *A2A*) платформа подключает внутренние и внешние сервисы — *task*-трекеры, репозитории, *CI/CD*, базы знаний, хранилища секретов, системы мониторинга без множественных кастомных интеграций.

→ Навыки (*skills*)

Навык — это сертифицированная, переиспользуемая единица функциональности, которую агент вызывает для решения класса задач. *Skills* имеют низкую стоимость контекста и встраиваются в платформу в виде постоянно пополняемой библиотеки — версионизируемой, эволюционирующей и тестируемой. Навыки выступают одним из удобных механизмов расширения, дополняя собой *hooks* (обработчики событий), плагины и *MCP*-интеграции.

→ *Observability* и операционная прозрачность

Платформа фиксирует все действия агентов, токены, события сжатия контекста, переходы между субагентами — формируя полноценный материал как для автоматизированного контроля качества, так и для последующего анализа поведения системы.

→ *Self-service* и расширяемость

Платформа предоставляет инженерам инструменты для создания собственных решений под задачи конкретных команд — навыков, агентов, маршрутов исполнения — с возможностью их сертификации и распространения внутри организации.

3.3 Context Engineering и Context Supply Chain

Context Engineering

Критическая дисциплина *AI*-нативной разработки. Контекст представляет собой конечный ресурс с убывающей предельной полезностью (т.н. *context rot*): чем больше контекста, тем ниже внимание модели к каждому отдельному элементу. Качество и точность формулировок контекста определяют, можно ли ожидать от агентного режима безопасных и применимых решений.

Context Supply Chain

Это не класс артефактов, а дисциплина и процесс, описывающие то, каким образом контекстные артефакты (AGENTS.md, skills, политики, evaluations и др.) создаются, актуализируются и распространяются в организации.

Дисциплина исходит из того, что артефакты контекста имеют владельца, область применимости (scope), срок актуальности (validity) и покрытие проверками (eval coverage), а правила их создания, изменения и распространения должны быть формализованы и соблюдаться так же строго, как для кода.

Правильно построенный процесс Context Supply Chain позволяет работать с типичными рисками:

- ✓ **Отравление контекста (context poisoning)** — злоумышленник изменяет файлы контекста, внедряя вредоносные инструкции
- ✓ **Устаревание контекста** — описание архитектуры расходится с реальностью, и агент действует по уже несуществующей модели
- ✓ **Конфликты между источниками** — контекст на уровне команды (локальный) противоречит контексту уровня организации (глобальный), агент непредсказуемо выбирает один из источников
- ✓ **Несанкционированный ввод контекста (prompt injection)** — внешний README, issue description или ответ MCP-сервиса заносит критичные для команды инструкции, не проходя ревью

Как и любой сложный артефакт, контекст создаётся и развивается через несколько уровней.

Задача этих уровней — провести контекст от состояния сырого источника к проверенному, отфильтрованному, релевантному конкретной задаче набору данных, который доставляется агенту в нужный момент (just-in-time), а не загружается целиком по любому поводу.

Это сочетание точности, актуальности и экономии контекстного окна и есть ключевой результат корректно выстроенного Context Supply Chain.

3.4 Перевод разработки в автономный режим

Перевод разработки в автономный режим опирается на дисциплину **Specification-Driven Development** и три инженерных столпа: **Policy as Code**, **Полный цифровой двойник IT-систем** и **Agent Development Lifecycle**

3.4.1 Specification-Driven Development (SDD)

Specification-Driven Development (SDD) — методологический подход, в котором спецификация становится первичным артефактом разработки. Спецификация, а не код, описывает, что должна выполнять система; код становится производным и эфемерным. Неработающий код не «чинится» — корректируется спецификация, которая порождает новую корректную реализацию. С точки зрения управления артефактами спецификация ведётся с той же дисциплиной, что и кодовая база: версионирование, ревью, регрессионные проверки на согласованность.

С формальной точки зрения SDD-документ описывает в структурированной, машиночитаемой форме целевую функциональность, обоснование её необходимости, область применения и ограничения, метрики достижимости результата, критерии приёмки и класс риска. Качество спецификации напрямую определяет качество и предсказуемость результата, поскольку именно спецификация задаёт смысловые рамки, в которых действует агентная система.

3.4.2 Policy as Code

Policy as Code — это дисциплина и совокупность инструментов, обеспечивающих встраивание нормативных требований организации в агентные микроциклы в форме автоматически применяемых правил и проверок. Принципиальная характеристика — машиночитаемость нормативного требования; конкретный формат записи (Markdown-инструкция или другой внутренний DSL) вторичен. Подход обеспечивает не только учёт требований на этапе поиска и реализации решения, но и автоматическую верификацию факта их выполнения.

Важно зафиксировать разграничение двух подходов к обязательности контекста:

- 1 Context-as-guidance** (документы вида AGENTS.md, skills, доменные инструкции) задают рекомендательные ориентиры.
- 2 Policy enforcement** (hooks, policy as code в виде quality gates, ограничения в runtime агента) обеспечивают обязательность применения требований.

Все требования, чувствительные с точки зрения нормативов организации — обращение с персональными данными, платёжной информацией, операциями над production-средой, критичными бизнес-процессами, — должны быть обязательны, а не оставаться в форме рекомендательных указаний. Политика, соблюдение которой возложено на людей, будет периодически нарушаться; политика, реализованная как автоматическая проверка в CI/CD и в runtime агента, исполняется по алгоритму.

Принципиально новое здесь — переход от детерминированных артефактов к недетерминированным.

Автоматические проверки в CI/CD существуют и сегодня, но они работают с алгоритмическими, проверяемыми объектами — кодом и юнит-тестами. AI-нативная разработка повышает уровень сложности задачи: появляются артефакты, поведение которых нельзя верифицировать однозначным запуском.

Ответом на этот сдвиг становится дисциплина **eval-driven development** и сопутствующие практики контроля качества AI-инструментов: статистическая верификация поведения агента и его компонентов на эталонных наборах задач, регрессионные оценки качества, отдельный жизненный цикл для таких проверок как для самостоятельного класса инженерных артефактов. Без этого слоя автоматические проверки покрывают лишь часть реальной поверхности риска.



Полный цифровой двойник IT-систем — это способ целостно и в достаточном объёме создать контекст для эффективной реализации агентами end-to-end задач.

Принципиальный смысл состоит в том, что продукты, процессы и технологии невозможно проектировать изолированно от остального IT-ландшафта — а он, как правило, не является целиком агентским. Поэтому всегда существует точка, в которой создаваемое решение должно встроиться в уже существующие системы и совместно с ними обеспечить целевой результат. Эту точку необходимо проверять комплексно — на сквозном end-to-end пути, в условиях, идентичных промышленной среде. Цифровой двойник служит именно такой средой проверки: он обеспечивает полноценную верификацию работоспособного решения вместе с его контекстным окружением до того, как изменение попадёт в production.

3.4.3 Agent Development Lifecycle (ADLC)

Agent Development Lifecycle (ADLC) — это выделенный специализированный цикл разработки автономных агентов. Он охватывает разработку и обучение моделей, доставку и контроль контекста, инструментарий запуска и взаимодействия агентов со средой, а также механизмы изоляции их процессов и контекста там, где это требуется.

По отраслевым прогнозам, к 2030 году **70% инженерных команд** перейдут к формализованному ADLC (от менее чем 10% в 2025 году), а **до 80% lead time** в разработке агентов будет уходить на обеспечение надёжности и доверия

3.5 Golden Path: автоматическая прослеживаемость от требования до деплоя

Golden Path — рекомендуемый маршрут прохождения задачи через AI PDLC, оптимизированный для AI-нативной разработки.

Это собирательный образ, состоящий из ключевых функций процесса: Specification-Driven Development, Policy as Code, цифровой двойник, Agent Development Lifecycle. Каждый шаг Golden Path оснащён автоматическими проверками, снижающими необходимость ручного контроля.

Прослеживаемость (traceability) от требования до деплоя обеспечивается автоматически: бизнес-требование формулируется в виде SDD-спецификации; платформа декомпозирует её на задачи для агентов; маршрутизатор назначает задачи оптимальным исполнителям; агенты работают в изолированных sandbox-средах; автоматические проверки качества контролируют результат на безопасность, корректность тестов и соответствие стилю; при прохождении проверок — развертывание в staging среду; финальное подтверждение перед production выполняется человеком. Каждое действие на каждом шаге фиксируется в неизменяемом машиночитаемом журнале для последующего аудита и операционной аналитики.

Желаемое состояние, к которому ведёт Golden Path, удобно описать через привычное и хорошо считываемое и бизнесом, и ИТ понятие — lead time, то есть времени от момента формулирования намерения до поставки результата конечному пользователю.

Цель зрелой IDP — кратное сокращение этого показателя: для типовой фичи lead time должен составить часы, а не дни, как это принято в традиционном SDLC. Ускорение достигается не только за счёт автоматизации, но и за счёт устранения простоев: агенты работают параллельно и непрерывно, тогда как люди работают последовательно и с задержками коммуникации.

3.6 Agent Marketplace и развитие платформы «снизу-вверх»

Создание платформы, её развитие и внедрение должны идти не только «сверху-вниз», но и «снизу-вверх». Принципиальное отличие IDP от классической централизованной платформы состоит в том, что инженеры на местах получают инструмент, позволяющий создавать инструменты под задачи конкретных команд. Такие инструменты — в форме навыков или сертифицированных агентов — должны быть доступны всей организации.

В качестве решения этой задачи мы предлагаем концепцию **Agent Marketplace — внутреннего маркетплейса верифицированных агентных решений.**

Маркетплейс, с одной стороны, делает видимым для всех заинтересованных участников факт существования готового решения, доступного для переиспользования; с другой — помогает понять, когда инженеру целесообразно переиспользовать существующее решение вместо создания нового. Это позволяет существенно сократить время на поиск и исключить дублирование трудозатрат.

Integrated Development Platform в этой логике объединяет два ключевых элемента:

- 1 Внутренний маркетплейс сертифицированных агентов (Agent Marketplace)**
Каталог проверенных AI-решений для разных корпоративных задач с описанием функциональных возможностей, ограничений, стоимости и рейтингов использования
- 2 Процесс управления корпоративным контекстом (Context Supply Chain / Context Engineering)**
Обеспечивает согласованность данных и предотвращает галлюцинации агента на всех этапах разработки

3.7 Архитектурные принципы зрелой IDP

Архитектурно зрелая IDP опирается на пять принципов:

1 Agent Runtime / Deterministic Harness

Агенты должны быть запущены в специализированной среде, которая создает вокруг агента закрытый контролируемый контур. Акцент - единый цикл выполнения агента, диспетчер инструментов, контроль прав, sandbox, сохранение и восстановление сессий. Это единственный компонент, который должен быть создан внутри. Если он будет передан harness наружу — компетенция уходит наружу.

2 Golden Path

Инструменты без процессов не дадут желаемого эффекта. Работа над агентами и их применением базируется на бесшовной среде вывода изменений и доступности непротиворечивой информации.

3 Self-service и расширяемость

AI как технология радикально повышает возможности по адаптации решений к конкретным задачам. Общий подход к организации должен оставлять место для bottom-up решений: навыки, реестр MCP-сервисов, API и инструменты.

4 Observability и AgentOps

Работа агентов должна быть прозрачной для человека, особенно – в этапах апробации\тиражирования. Вариативность не должна восприниматься как бесконтрольность. Ключевые инструменты: журналы выполнения, токен-бюджет, события сжатия контекста, трассировки использования инструментов, дашборды.

5 Security и Governance (Policy Hook Framework)

Свобода выбора действий агентом не должна приводить к «свободной трактовке» правил. Часть контекста представляет собой требования к исполнению, среда управления агентом должна обеспечить однозначное исполнение требований. Агент находится в системе разрешений в логике implicit deny – все, что явно не разрешено, запрещено и требует запроса.

Над этим фундаментом надстраивается слой ключевых функциональных возможностей:

единый портал разработчика, Agent Marketplace, управление контекстом, реестр спецификаций, AI-усиленный CI/CD, дашборд токенов-экономики, инструменты политики безопасности, наблюдаемость, корпоративный граф знаний, контур аудита.

Успех определяется корректной приоритизацией исходя из ценности и текущей зрелости команд.

В заключение раздела хочется подчеркнуть ещё раз: **платформа позволяет управлять тысячами агентов в едином корпоративном контуре**, обеспечивая прозрачность операций и аудит изменений — и делать это безопасно при условии качественного контекста и точности доступных агенту сведений.

По имеющимся отраслевым данным, до 80% lead time в разработке агентов сегодня уходит на обеспечение надёжности и доверия, и в абсолютном большинстве случаев перво-источником ошибок агентного режима является именно низкое качество контекста. Поэтому контекст и harness — не вспомогательные элементы, а главные объекты внимания при построении AI-нативной разработки.

04

Безопасность, управление и новые роли в эпоху AI

4.1 Enterprise-готовность: где заканчиваются эксперименты и начинается ответственность

Исследование, апробация технологий и эксперименты принципиальным образом отличаются от enterprise-среды. Любая критически важная функция — и любая функция с высокими требованиями к точности и надёжности — не может допускать неопределённости и отсутствия управления.

Принципиальный вызов в создании AI-нативного цикла разработки в целом и IDP в частности — это построение enterprise-ready решений, в которых функциональность, быстрое прототипирование и оперативная управляемость изменениями не вступают в противоречие с организационными нормативами и внутренними стандартами качества и надёжности.

Эмпирические данные международных исследований агентных систем показывают: когда модель или агент запрашивает у человека разрешение на действие, в подавляющем большинстве случаев — порядка 93% запросов — человек не вчитывается в содержание и подтверждает действие автоматически.

Иными словами, опираться на ручное yes/no-одобрение как на основной контур безопасности невозможно — он систематически вырождается. Первичный контроль обеспечивают policy as code и автоматизированные hooks; ручное согласование сохраняется только там, где этого требует класс риска, и используется как дополнительный сигнал и след аудита, а не как основной механизм защиты.

4.2 Harness — окружение агентной среды

AI-Disrupt PDLC и агентное кодирование в целом выходят за рамки работы LLM-модели и охватывают harness как самостоятельный объект внимания. Harness — это окружение агентной среды: совокупность управляющего цикла выполнения, контроля прав, инструментов, изоляции, аудита и операционных механизмов, в которых живёт модель.

Иначе говоря, harness — та обвязка, которая превращает мощную, но недетерминированную модель в работающий и предсказуемый корпоративный инструмент.

Модель может изменяться и эволюционировать; именно окружение обеспечивает воспроизводимость, управляемость и надёжность поведения системы вне зависимости от того, какая именно модель находится в её ядре.

Точность и предсказуемость поведения системы «модель + harness» критически зависит от качества и доступности организационного контекста. Именно поэтому дисциплина Context Engineering и концепция Specification-Driven Development являются краеугольными для успеха AI-трансформации. Это переход от «контроля процесса» к управлению границами доверия.

4.3 Адаптивная лестница автономии агента

Между полной автоматизацией и полностью ручным подтверждением операций существует **широкий спектр самостоятельных действий агента**.

Положение агента на этом спектре не должно быть фиксированным: оно меняется в зависимости от контекста задачи, степени её критичности, чувствительности данных и потенциального влияния изменений на промышленную среду. Чем выше класс риска, тем меньше автономии — и тем больше явных проверок и человеческих точек принятия решения.

В AI Disrupt PDLC мы руководствуемся эмерджентным подходом:

- ✓ На текущем уровне развития агенты могут выполнять задачи с участием человека или полностью автономно, однако любой результат должен быть проверен и принят человеком.
- ✓ Именно человек несет ответственность за результат всей цепочки создания продукта, AI лишь меняют распределение труда. Этот принцип останется неизменным и в будущем
- ✓ На момент данной публикации мы доверили AI-агентам работу с минорными дефектами — от отслеживания до этапа code review. Инженер по-прежнему принимает исправление, но уже не участвует в поиске решения, code review и рутине (написание кода, тестов, прогон тестов, ревью кода).

4.4 Agentic FinOps и Cost Circuit Breakers

С переходом к агентному кодированию возникает **новый класс операционных расходов — токеномика**.

Token usage объясняет 80% дисперсии производительности агентов; multi-agent архитектура потребляет ~15× больше токенов по сравнению с chat-режимом.

Без системного контроля затраты на агентов могут составить 40–60% общего ИТ-бюджета к 2027 году.

Принципиальный момент: контроль потребления токенов не должен сводиться к пассивному мониторингу. Платформа обязана не только видеть превышение бюджета, но и активно реагировать на него — приостанавливать выполнение, эскалировать решение человеку, прерывать неконтролируемые сценарии.

Особое внимание требуется к контролю заикливания агентов — состояний, в которых агент входит в неограниченный цикл уточнений и потребляет ресурсыкратно выше ожидаемого объёма. Без активного механизма защиты подобные сценарии способны нанести значительный финансовый урон, оставаясь технически «корректным» поведением системы.

Финансовый контроль агентов (Agentic FinOps) реализуется в IDP как:

- ✓ **Мониторинг потребления токенов** в реальном времени по командам, проектам, типам задач, моделям
- ✓ **Расчет бюджета токенов на задачи** с автоматической эскалацией к человеку при превышении пороговых значений
- ✓ **Выбор оптимальной модели под задачу** — автоматический выбор модели, оптимальной для решения конкретной задачи, достижимый эффект — экономия 60–70% токенов
- ✓ **Кэширование/уровень попадания для системных инструкций** (до 10× снижения стоимости input tokens для повторных запросов)
- ✓ **Блокировка заикливания (cost circuit breakers)** — применение концепции контроля заикливания кода на потребление токенов

4.5 Эволюция ролей: трансформация человека, а не его замещение

Классический разработчик не исчезает. Его роль **расщепляется на несколько специализаций — Product Engineer, Platform Engineer, AI Engineer и другие**.

Меняется внутренняя структура каждой из них, но за каждой сохраняется собственная группа задач. Каждый специалист, создающий продукты сегодня, в той или иной мере уже работает с продуктом, инструментами платформы, поиском архитектуры решения и другими техническими задачами. Мы исходим из того, что трансформируются именно люди и роли — не профессии исчезают, а их внутренняя структура меняется.

У нас нет оснований считать, что AI приведёт к сокращению занятости в отрасли. Если ранее человек был вынужден осваивать языки машин и переводить намерения на эти языки, то AI-трансформация существенно снижает этот барьер. Она позволяет сосредоточиться непосредственно на намерении — на поиске ответов на вопросы: что необходимо сделать, для кого это нужно сделать.

Параллельно с продуктовыми специализациями для harness и Context Supply Chain формируется отдельная enterprise-инфраструктурная линия ролей:

- ✓ Инженеры, отвечающие за политики и контролирующих агентов
- ✓ Инженеры платформы и runtime
- ✓ Специалист по безопасности AI
- ✓ Владелец Context Supply Chain
- ✓ Инженер AgentOps

Эти роли, как правило, концентрируются в платформенной команде и обеспечивают остальные команды управлением «как сервисом» — без необходимости разбираться в политиках и инструментах вручную.

Над технической грамотностью надстраиваются шесть ключевых компетенций будущего, все они дополняют фундамент классической инженерной грамотности, а не заменяют его:

- ✓ Формулирование намерений (prompt engineering)
- ✓ Архитектура контекста
- ✓ Системное мышление
- ✓ AI-грамотность
- ✓ Верификация и ревью
- ✓ Бизнес-ориентация

05

План реализации и стратегические рекомендации

Заключительный раздел переходит от концепции и архитектуры к практике внедрения: излагает принципы, в соответствии с которыми разворачивается AI-трансформация, описывает горизонты её движения и формулирует стратегические рекомендации, опирающиеся на реальный промышленный опыт, а не на отвлечённые лучшие практики.

5.1 AI-трансформация не может быть спланирована заранее

Мы считаем, что AI-трансформация не может быть спланирована заранее в традиционном смысле — как последовательность жёстко определённых шагов с фиксированными датами и заранее предписанными результатами. Желаемые изменения слишком масштабны, чтобы быть точно спланированными, без ошибок и пересмотра. Откладывать трансформацию до момента, когда все детали будут «прояснены», — также большая ошибка: к этому моменту окно опережающего входа окажется уже закрытым.

Эмерджентная работа — это не отказ от планирования. Это способ двигаться к ясно сформулированному образу результата в условиях, когда сама технология ещё проходит фазу становления, готовых рецептов не существует, а опыт enterprise-внедрений AI-трансформации в значительной мере ещё не накоплен ни одной отраслью.

В таких условиях корректная установка — формулировать принципы и ключевые условия успеха, удерживать целостный образ цели и выбирать следующий шаг по мере того, как предыдущий приносит измеримый результат. Это обеспечивает движение, не теряющее направления, и одновременно сохраняет способность реагировать на новое знание.

5.2 Vibe Working как управленческая философия

AI-трансформация значима не только для инженерных команд. Один из её ключевых эффектов — снятие технического барьера между человеком, обладающим точным пониманием задачи, и возможностью эту задачу реализовать.

Раньше создавать продукты с IT-составляющей могли преимущественно разработчики — те, кто владел навыками работы с языками программирования, фреймворками, средствами тестирования и развёртывания, — и они работали под руководством владельцев продуктов.

Сегодня преимущество получает тот, кто лучше понимает, что нужно сделать: точно формулирует намерение и потребность. Этот сдвиг ценен даже в сравнении с теми, кто просто хорошо пишет код.

Принципы AI-Disrupt PDLC переносятся за пределы инженерии. Юрист, маркетолог, аналитик, HR-специалист работают на уровне намерений; агенты обеспечивают исполнение. Это не метафора, а рабочая дисциплина, применимая в любой функции организации.

Vibe Working — управленческая философия, в которой команды любого профиля принимают AI-нативную модель: чёткое намерение, машиночитаемая спецификация, автономное исполнение, человеческое суждение на ключевых развилках.

5.3 Горизонты трансформации

Дорожная карта AI-Disrupt PDLC организована не как последовательность жёстких этапов, а как три горизонта движения. Они накладываются: пока в одних командах завершается первый горизонт, в других уже начинается второй, а в наиболее зрелых — третий.

Горизонт 1 Устранение ограничений

Первый горизонт посвящён устранению структурных ограничений, при которых AI-трансформация невозможна. На этом этапе организация фиксирует базовое состояние инженерных метрик, выстраивает дисциплину работы с контекстом и спецификациями, формирует минимальный контур платформы (включая базовый Agent Runtime, единый портал и автоматизированные проверки в CI/CD), внедряет принципиальную политику использования AI и пилотирует подход на нескольких командах.

Цель горизонта — снять ограничения, унаследованные из доAI-эпохи, и создать управляемую среду, в которой агентные системы могут действовать предсказуемо.

Горизонт 2 Включение AI-нативных процессов

На втором горизонте организация переходит от точечного применения агентов к AI-нативным процессам в полном смысле. Specification-Driven Development становится обязательным стандартом для новых продуктов. IDP расширяется до полного стека возможностей. Маршрут Golden Path активируется для типовых задач. Команды переходят к компактному составу. Появляются устойчивые сценарии мультиагентного взаимодействия. На этом горизонте в работу вовлекаются механизмы автоматического применения политик, контролирующие агенты и операционная прозрачность.

Горизонт 3

Новые роли и AI-нативная организация

Третий горизонт — переход к AI-нативной организации разработки, где автономные агенты выполняют большинство рутинных задач, а инженеры сосредоточены на формулировании намерений, архитектурных решениях, надзоре и продуктовой стратегии. Именно на этом горизонте новые роли — Product Engineer, Platform Engineer, AI Architect, Forward-Deployed Engineer — становятся базовыми. К этому моменту организация работает с собственной AI-фабрикой, в которой код является производным артефактом от спецификации, а первичной валютой создания ценности становится качество мысли.

Ожидаемые результаты на каждом горизонте:

- ✓ Подтверждение жизнеспособности подхода на практике
- ✓ Формирование внутреннего центра компетенций
- ✓ Сбор реальных данных о результативности подхода и корректировка стратегии
- ✓ Подготовка к безопасному масштабированию решения на все подразделения бизнеса

5.4 Культура экспериментов

Без специально выстроенной культуры эмерджентная трансформация не работает: дорожная карта без культуры превращается в формальный документ, а технологические возможности — в неиспользуемые инструменты. Поэтому отдельный акцент дорожной карты — формирование культуры, в которой быстрые циклы обратной связи, готовность к проверке гипотез и право на ошибку являются не лозунгом, а рабочей нормой.

Инновационный фокус трансформации — культура экспериментов:
быстрая адаптация к рыночным изменениям через короткие циклы обратной связи и систематическая работа с гипотезами.

Vibe Working и эмерджентный подход определяют, что значит быть AI-нативной организацией по сути, а не лишь декларировать этот статус. В этой культуре AI не становится фактором сокращения занятости в отрасли.

Если ранее человек был вынужден осваивать языки машин и переводить намерения на эти языки, то AI-трансформация существенно снижает этот барьер. Она позволяет сосредоточиться непосредственно на намерении — на поиске ответов на вопросы: что необходимо сделать и для кого.

Более того, AI создаёт возможности для тех, кто ранее оставался вне IT-индустрии: для людей, обладавших способностью к содержательной мысли, но не располагавших техническими навыками, чтобы формулировать инструкции для машины. Теперь эта возможность открыта для них.

5.5 Заключительное слово

Невозможно управлять тем, что нельзя измерить. Именно поэтому AI-трансформация начинается не с выбора инструментов и не с найма AI-инженеров — **она начинается с фиксации точки отсчёта.**

Без понимания того, где организация находится сегодня, какие метрики характеризуют её работу и в каких метриках мы намерены увидеть изменение, любая дальнейшая работа превращается в цепочку ярких, но несравнимых эпизодов, не дающих оснований для управленческих решений и оценки достигнутого результата.

Из этого следует чёткая последовательность первых решений, которые могут быть приняты сегодня:

01 Измерение текущего состояния (baseline)

Провести оценку инженерной организации, зафиксировать текущие метрики продуктивности, качества и стоимости. Без точки отсчёта не существует «до» и «после».

02 Запуск пилота

Выбрать одну команду из 3–5 инженеров, наделить её полномочиями экспериментировать и сформулировать задачу: пройти первый горизонт трансформации в управляемом режиме, с измеримыми результатами на каждом шаге.

03 Назначение AI Champions

Определить инженера или руководителя, ответственного за накопление и распространение практик. AI Champion — не работа на полный день, а роль катализатора изменений и фокус-точки накопленного опыта.

Эти три шага не требуют значительного бюджета, отдельного одобрения совета директоров или смены архитектуры. Они требуют единственного — решимости начать.

Резюме

Переход к модели AI-Disrupt PDLC — это не просто смена инструментария, а фундаментальный сдвиг в сторону гибкости и скорости. Мы освобождаем команды от ненужной когнитивной нагрузки, от инженерной рутины, обеспечивая им возможность концентрироваться на инновациях, которые напрямую влияют на рыночные позиции и удовлетворённость клиентов.

Организации, отказавшиеся от трансформации, понесут потери — в первую очередь в части бизнес-результативности. Конкурентное преимущество в горизонте 2028–2030 годов будет определяться не наличием AI-инструментов (они будут у всех), а качеством AI-культуры: способностью эффективно взаимодействовать с агентными системами, удерживать и развивать человеческую экспертизу, непрерывно обучаться на результатах AI и принимать ответственные решения на стыке автономии и контроля.

Сегодня в отрасли на первый план выходят новые понятия и образы — Intent Loop, Implementation Loop, Vibe Working, Context Engineering, Specification-Driven Development, evidence bundle, harness, risk ladder, Agent Marketplace, Tiny Teams. Каждое из них в горизонте одного-двух лет войдёт в повседневный словарь руководителей и инженеров. Мы формируем эти концепции совместно с ведущими специалистами, валидируем их на собственном промышленном масштабе и готовы делиться накопленным знанием. Мы видим преимущество открытого подхода: распространение знаний приносит пользу всей отрасли. Во многом именно благодаря открытому обмену стал возможен прогресс в области AI.

Закрытость технических знаний перестаёт быть устойчивой опорой развития и конкурентных преимуществ. Технические решения окажутся в распоряжении всех участников рынка, а стоимость их воплощения снизится практически до нуля. Главным дефицитом — как и в столетия, предшествовавшие AI-эпохе, — остаётся намерение: способность постичь суть, понять клиента, понять человека. Это и есть собственно человеческая работа. Это и есть ключ к процветанию — и мы убеждены, что он останется неизменным.

Источники

1. Google Cloud. DORA AI Capabilities Model 2025: Google Cloud DevOps Research and Assessment. 2025.
2. Anthropic. Measuring Agent Autonomy in Practice. 2026.
3. Anthropic. AI Assistance and Coding Skills. 2026.
4. Anthropic. Agentic Permissions Telemetry. 2026.
5. Anthropic. Writing Effective Tools for AI Agents. 2025.
6. Amazon Web Services. AI-Native Development Life Cycle (AI-DLC). 2025.
7. Dive into Claude Code. arXiv: 2604.14228v1. 2026.
8. Bain & Company. Engineering Practices and Financial Performance Research.
9. Consortium for Information and Software Quality (CISQ). The Cost of Poor Software Quality in the US. 2022.
10. Linux Foundation. Agentic AI Foundation and Model Context Protocol (MCP). 2025.
11. Внутренние материалы Сбер: AI-PDLC, AI-Disrupt PDLC, GigaChat, программы СберУниверситета и AIRI. 2025–2026.

Кирилл Меньшов

старший вице-президент, руководитель блока
«Технологии» Сбера, автор AI-Disrupt PDLC

Май 2026