# ИИ-ТРАНСФОРМАЦИЯ РАЗРАБОТКИ

ТРАНСФОРМАЦИЯ ПРОЦЕССА
ИТ-РАЗРАБОТКИ ПО
ПОД ВЛИЯНИЕМ ИИ

2026

АССОЦИАЦИЯ
ФИНТЕХ

# СОДЕРЖАНИЕ

# КТО И КАК ПИШЕТ КОД СЕГОДНЯ?

**МАРИАННА ДАНИЛИНА**

*Руководитель управления стратегии, исследований и аналитики Ассоциации ФинТех*

Еще несколько лет назад вопрос об ИИ в разработке программного обеспечения звучал как тема отдаленного будущего. Сегодня он стал частью повседневной реальности банков и финтех-компаний. 92% разработчиков в 2026 году используют или планируют использовать ИИ-инструменты в работе, а доля кода, написанного с участием ИИ, превысила 46%.

Российский финансовый сектор оказался в особой позиции: с одной стороны у него довольно технологически зрелые крупнейшие банки и накопленный опыт в области ИИ, а с другой – требования к суверенности данных и технической необходимости работы с отечественными моделями.

В этом исследовании мы попытались зафиксировать влияние ИИ на ИТ-разработку и понять, кто и как сегодня пишет код в российском финтехе. Возможно, это только первый шаг к более широкой дискуссии.

АССОЦИАЦИЯ ФИНТЕХ

***Ассоциация ФинТех*** *основана в конце 2016 г. по инициативе Банка России и ключевых участников отечественного финансового рынка. Это уникальная площадка для конструктивного диалога регулятора с представителями бизнеса. Здесь формируется экспертная оценка инновационных технологий с учетом международного опыта, а также разрабатываются концепции финансовых технологий и подходы к их внедрению.*

## УЧАСТНИКИ АССОЦИАЦИИ ФИНТЕХ:

Банк России · ВТБ · СБЕР · Альфа Банк · ГАЗПРОМБАНК · ПСБ

ВСК страховой дом · РСХБ | ЦИФРА · MOEX GROUP · Т БАНК · ДОМ РФ · СОВКОМБАНК

НСПК · Райффайзен БАНК · Avito · точка банк для предпринимателей и предприятий · Яндекс · МКБ · ТІ

МТС · Кредит ЕвропаБанк · РЕСО ГАРАНТИЯ · Ак Барс Банк · abb · DIASOFT всё по-настоящему

БАНК ЦЕНТР-ИНВЕСТ · РЕДСОФТ · отпбанк · синара банк · БПС инновационные решения

ДОХОД управляющая компания · FLEX software systems · ВБРР · АЛЬФА СТРАХОВАНИЕ · Ростелеком · КАПИТАЛ LIFE страхование жизни

# КЛЮЧЕВЫЕ ВЫВОДЫ

## Кто и как пишет код сегодня?

Искусственный интеллект перестал быть экспериментом и стал повседневным инструментом разработки ПО для лидеров рынка из числа участников Ассоциации ФинТех. Большинство этих компаний уже применяют ИИ в разработке. Однако эффект от его внедрения оказался неоднозначным: скорость разработки растёт, но качество и безопасность становятся новой зоной риска. Сегодня **ИИ не заменяет разработчика, а меняет его роль:** от написания кода к постановке задач и контролю работы ИИ.

1

### Смена парадигмы ИТ-разработки

ИИ стремительно прошел путь от вспомогательного инструмента разработчика до полноценного участника жизненного цикла разработки.

Несмотря на то, что 92% разработчиков в США так или иначе применяют вайб-кодинг, он несёт риски потери управляемости в процессе разработки. В целом точно оценить эффект от применения ИИ могут только единицы. Этот эффект не является автоматическим: помимо стоимости самих моделей и инфраструктуры, компаниям приходится учитывать затраты на интеграцию, доработку процессов, контроль качества и обеспечение безопасности ИИ-решений.

Следующий этап — переход к агентным системам и новой модели AI Product Development Life Cycle (AI-PDLC), что требует от компаний пересмотра операционных моделей, процессов и принципов разработки.

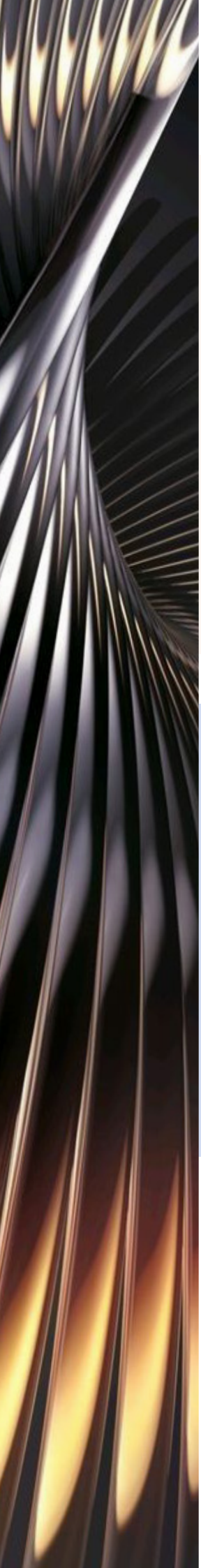
2

### ИИ в жизненном цикле продуктовой разработки

Мы наблюдаем смену самой модели создания цифровых продуктов и переход к гибридному формату взаимодействия разработчика и ИИ.

Наибольший эффект от ИИ в разработке достигается при переходе к операционной модели AI-PDLC, когда ИИ интегрирован во все этапы жизненного цикла продукта. При этом подходе разработчик перестает быть основным производителем кода и становится «куратором» и «контролером решений», сгенерированных ИИ.

ИИ сильнее всего ускоряет написание и автодополнение кода, генерацию тестов и документацию. Однако разработчики всё ещё не готовы полностью доверять ИИ на этапах развертывания и мониторинга.



3



### ИИ меняет роль ИТ-специалистов

Рутинное написание кода, реализация типовых решений и другие стандартизируемые задачи постепенно автоматизируются, тогда как возрастает роль специалистов, способных проектировать системы, принимать архитектурные решения, корректно ставить задачи ИИ и отвечать за качество конечного результата. При этом использование ИИ не отменяет необходимость глубокой технической экспертизы. Напротив, чем больше кода генерируется автоматически, тем важнее способность специалиста оценить его корректность, выявить ошибки и уязвимости и определить, соответствует ли решение требованиям конкретной системы. В связи с этим нагрузка на наиболее квалифицированных специалистов может не снижаться, а, наоборот, возрастать.

4

### Безопасность разработки становится новым фактором успеха

Конкурентное преимущество получают не те, кто быстрее внедряет ИИ, а те, кто способен встроить его в безопасную, управляемую и проверяемую среду. Роль ИБ-специалистов меняется, они становятся архитекторами безопасной среды.

Массовое использование ИИ в разработке ПО меняет архитектуру безопасного жизненного цикла. 65% участников исследования называют требования информационной безопасности главным барьером внедрения, а 53% – ограничения на передачу данных внешним сервисам. Появляются специфические риски: качество ИИ-кода, сложность понимания принципа работы кода и рост технического долга.

# 01.

# МЕТОДЫ РАЗРАБОТКИ С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

- AI-ASSISTED & AI-AUGMENTED
- ВАЙБ-КОДИНГ
- VIBEOPS
- AI-DRIVEN DEVELOPMENT
- SPEC-DRIVEN DEVELOPMENT
- АГЕНТНЫЙ ИНЖИНИРИНГ

**Максим Григорьев**

**Генеральный директор Ассоциации ФинТех**

АССОЦИАЦИЯ ФИНТЕХ

*За беспрецедентно короткое время мы прошли путь от использования ИИ в качестве ассистента разработчика до полноценного автономного агента, которому можно делегировать большую часть рутинных задач. Точкой бифуркации стал феномен вайб-кодинга: он вызвал настоящий резонанс в отрасли, хотя значительная часть первоначального ажиотажа оказалась скорее хайпом, чем устойчивой практикой. Очень привлекательной показалась возможность ставить задачи на естественном языке и ожидать получения идеального результата. Реальность оказалась сложнее – чтобы системно применять ИИ в корпоративном ландшафте, потребовалось переосмыслить саму идею управления жизненным циклом процесса разработки ПО.*
*Мы видим, как лидеры рынка – участники Ассоциации ФинТех – на наших глазах создают новую реальность, формируя современную методологию AI-PDLC, которая сочетает в себе «переосмысленную» процессную модель, новые подходы к управлению архитектурой, безопасностью и компетенциями.*

## AI-ASSISTED & AI-AUGMENTED DEVELOPMENT

Между **AI-Assisted Development** и полноценным **AI-Driven Development** находится промежуточная стадия, которую обозначают как **AI-Augmented Development** – разработка, усиленная ИИ. Ключевое отличие от смежных понятий состоит в следующем.

| | | |
|---|---|---|
| 1 | **AI-Assisted Development** | Предполагает точечное использование ИИ-инструментов для отдельных задач при полном сохранении существующих процессов. |
| 2 | **AI-Augmented Development** | ИИ встроен в рабочие процессы и расширяет возможности разработчика, однако решения и ответственность за результат остаются за человеком. |
| 3 | **AI-Driven Development** | Модель требует переосмысления самой архитектуры процесса, когда ИИ помещается в центр, а человек берет на себя роль куратора. |

При разработке, усиленной ИИ (AI-Augmented Development), команда использует ИИ-ассистенты на всех ключевых этапах ИТ-разработки - при написании кода, тестировании, ревью, документировании. При этом не переходит к агентным системам и не перестраивает процессы управления продуктом под логику ИИ. Разработчик по-прежнему определяет направление, ИИ только ускоряет движение.[1]

Согласно концепции AI-Disrupt PDLC от Сбера[2] за счет того, что ИИ избавляет от рутинных операций, компании в будущем перейдут от классических больших инженерных команд разработки (10-15 специалистов с AI Copilot) к компактным командам из 4-5 человек, усиленных ИИ-агентами.

**4-5 ЧЕЛОВЕК** — целевая численность команды разработки с использованием ИИ-агентов к 2030 году[3]

# ВАЙБ-КОДИНГ КАК ТРЕНД

**92%** разработчиков в США в той или иной мере применяют вайб-кодинг в работе[1]

В 2025 году исследователь в области ИИ и сооснователь OpenAI Андрей Карпати ввел термин **«вайб-кодинг» (vibe-coding)** для обозначения подхода к разработке, при котором разработчик описывает желаемый результат на естественном языке и принимает сгенерированный ИИ код без детального изучения. Кратко концепцию Карпати описывает так: разработчик «просто видит, говорит, запускает и копирует, и это в основном работает».[2]

**Суть подхода заключается в том, что разработка осуществляется не через детальную спецификацию и программирование, а через последовательное уточнение задачи в диалоге разработчика с ИИ. Пользователь формулирует намерение, ИИ предлагает решение, которое затем корректируется и дорабатывается. Этот процесс может повторяться несколько раз, пока результат не будет удовлетворять требованиям.**

**Ключевое отличие вайб-кодинга от классической разработки** – снижение степени формализации на ранних этапах. Вместо того чтобы заранее определить все требования, пользователь может двигаться от общего к частному, уточняя детали по мере необходимости. Это делает процесс более гибким и адаптивным.

## Андрей Карпати

**Andrej Karpathy**

ANTHROP\C

Словацко-канадский исследователь в области ИИ, один из основателей OpenAI. В 2017-2022 годах – директор по ИИ в Tesla, где руководил разработкой системы Autopilot Vision. В 2024 году основал образовательный стартап Eureka Labs. В мае 2026 года перешёл в Anthropic в команду предобучения модели Claude. Автор термина «вайб-кодинг» и образовательной серии «Neural Networks: Zero to Hero». Входил в список 100 самых влиятельных людей в ИИ по версии TIME (2024).

КЕЙС №1

Alibaba Cloud Qwen

## СОБСТВЕННЫЕ ИИ-РЕШЕНИЯ ДЛЯ ПРОГРАММИРОВАНИЯ

**Alibaba Cloud** расширила свою экосистему больших языковых моделей **Qwen (Tongyi Qianwen)**, выпустив специализированные модели и агентные инструменты для программирования, позволяющие создавать, отлаживать и изменять программный код с помощью запросов на естественном языке.

В их число вошло решение **Qwen Code**, которое позиционируется как китайская альтернатива GitHub Copilot и другим западным ИИ-помощникам. Эта инициатива стала частью стратегии Alibaba по созданию ИИ-ориентированной среды разработки для корпоративных и частных пользователей.

1. По данным Guvi: guvi.in
2. По данным X Corp.: x.com

Несмотря на привлекательность идеи разработки через естественный язык, вайб-кодинг имеет ограничения, становящиеся решающими при переходе от прототипов к промышленным решениям:

### ОГРАНИЧЕНИЕ ВАЙБ-КОДИНГА:

**Сложность масштабирования**

Простота создания решений не означает их готовность к работе в сложной архитектуре или под высокой нагрузкой. **Без участия опытных специалистов такие решения могут оказаться нестабильными или неэффективными.**

**Риск потери управления**

Когда процесс разработки строится на диалоге, возникает риск потери контроля над структурой и логикой системы. Это особенно критично в крупных проектах, где требуется координация между командами и соблюдение единых стандартов.

**Иллюзия простоты**

Снижение барьера входа в вайб-кодинг может создавать ощущение, что разработка стала тривиальной задачей. На практике же сложность смещается с этапа создания на этап понимания и контроля, что требует не меньшей, а иной квалификации разработчика.

Постепенное выявление проблем, связанных с применением такого подхода, получило название **«похмелье вайб-кодинга»**[1]. Специалисты, получившие в работу кодовые базы, созданные преимущественно через промпты, столкнулись с тем, что написанный код работает, но нет понимания, как и почему. В проектах, которые казались реализованными, начали выявляться скрытые проблемы, которые ИИ не мог исправить, потому что не мог объяснить собственный код.

## VibeOps: РАСПРОСТРАНЕНИЕ ПРИНЦИПОВ НА ОПЕРАЦИОННЫЙ УРОВЕНЬ

По мере распространения вайб-кодинга закономерно возник вопрос обеспечения промышленной управляемости систем, созданных таким способом. Ответом стало появление понятия VibeOps – концепции, которая распространяет принципы вайб-кодинга на операционный уровень.[2]

**VibeOps** – подход, при котором многие задачи по настройке, запуску и поддержке ИТ-систем выполняются не вручную через сложные настройки и скрипты, а **с помощью общения с ИИ на обычном языке.**

**Разработчик** описывает, что нужно сделать.

**ИИ помогает** настроить инфраструктуру, развернуть сервисы и выполнить необходимые операции.

Такой подход позволяет **сократить количество рутинных действий**, **упростить работу** и не отвлекаться **от основной задачи разработки**. В более широком смысле **VibeOps – это новый формат организации DevOps-процессов**, где вместо работы с большим количеством технических конфигураций основным интерфейсом становится диалог с ИИ-ассистентом.

1. По данным Fast Company: fastcompany.com
2. По данным T4itech: t4itech.com

# AI-DRIVEN DEVELOPMENT

Понятие разработки с применением ИИ (AI-Driven Development) не имеет единого общепринятого определения, однако в профессиональном ИТ-сообществе оно устойчиво используется для обозначения подхода, при котором **искусственный интеллект** - это не вспомогательный инструмент, а **центральный участник** всего процесса создания программного обеспечения.

## КОНТЕКСТ ПОЯВЛЕНИЯ ПОНЯТИЯ AI-DRIVEN DEVELOPMENT

**2023**

До 2023 года ИИ применялся в разработке точечно для автодополнения кода, генерации документации или поиска ошибок. Такую модель называют **ИИ-ассистированной разработкой** (AI-Assisted Development), где инициатива и контроль полностью сохраняются за человеком-разработчиком.

По мере того как языковые модели становились мощнее, а инструменты – сложнее, возникло принципиально иное представление о роли ИИ.

**2025**

aws

В 2025 году компания **Amazon Web Services** (AWS) формализовала этот переход, представив методологию, которую дословно можно перевести как **«жизненный цикл разработки, управляемой ИИ»**: AI-Driven Development Lifecycle (AI-DLC):

- В ней ИИ помещается в центр разработки: от формулировки замысла до эксплуатации системы.
- По определению AWS, AI-DLC – это трансформационный подход к разработке программного обеспечения, ориентированный на ИИ при сохранении человеческого контроля принятия критически важных решений[1].

**2026**

СБЕР

В 2026 году **Сбер** представил собственное видение аналогичной концепции, получивший название **AI-Disrupt PDLC** (Product Development Lifecycle).
В отличие от AWS AI-DLC, российская версия делает акцент на трех сущностях:

- Использованию ИИ-агентов в разработке.
- Интегрированной среде исполнения (Agent Runtime).
- Разработке через спецификации (Specification-Driven Development).

**Ключевой тезис концепции AI-Disrupt PDLC – по мере того, как исполнение становится практически мгновенным, единственным дефицитным ресурсом остаётся намерение (intent) человека, вокруг которого и предлагается перестраивать весь жизненный цикл разработки, а не просто встраивать ИИ-ассистентов поверх существующих процессов.**

## КЛЮЧЕВЫЕ ХАРАКТЕРИСТИКИ AI-DRIVEN DEVELOPMENT

### 1 ИИ является основой процесса, а не дополнительной функцией

ИИ не добавляется к существующему процессу ИТ-разработки. Напротив, процесс проектируется вокруг ИИ.

### 3 Человек играет роль контролера и оркестратора

Ответственность ИТ-разработчика смещается с написания кода на постановку задач, архитектурный надзор, валидацию и принятие решений в точках неопределенности.

1. По данным Amazon Web Services (AWS): aws.amazon.com

**2 Намерение как точка входа в процесс разработки**

Разработка начинается не с формализованных требований, а с описания желаемого результата на естественном языке. ИИ самостоятельно выстраивает необходимый контекст, оценивает сложность задачи и формирует план дальнейших действий.

**4 Жизненный цикл разработки, управляемой ИИ (AI-DLC)**

В отличие от жестко заданных процессов жизненного цикла программного обеспечения (Software Development Life Cycle, SDLC), AI-DLC адаптирует глубину и этапы разработки к природе конкретной задачи. Например, простые утилиты не требуют полного цикла архитектурного проектирования, в отличие от сложных систем.

КЕЙС №2

Microsoft GitHub Copilot

## ПРОМЫШЛЕННОЕ МАСШТАБИРОВАНИЕ GITHUB COPILOT

**Microsoft** стала первой крупной технологической компанией, развернувшей ИИ-ассистент разработки в промышленном масштабе. По данным контролируемого эксперимента **Microsoft Research (2023)**, разработчики выполняли задачи по написанию HTTP-сервера на JavaScript на 55,8% быстрее при использовании Copilot[1]. К июлю 2025 года Copilot набрал 20 млн суммарных пользователей, к январю 2026-го – 4,7 млн платных подписчиков[2]. Инструмент используется в 90% компаний **Fortune 100**[3].

## SPEC-DRIVEN DEVELOPMENT: СТРУКТУРИРОВАННАЯ АЛЬТЕРНАТИВА

**2025** **Разработка на основе спецификаций (Spec-Driven Development, SDD)** появился в 2025 году как ответ на многочисленные проблемы, с которыми столкнулись разработчики при активном использовании ИИ для написания программного кода.

Главная проблема – **ИИ со временем может начинать отклоняться от первоначальной задачи**, особенно в крупных проектах с большим объемом кода. Кроме того, без четко зафиксированных требований становится сложно проверять, соответствует ли результат работы модели ожиданиям разработчиков.

Подход Spec-Driven Development предполагает, что разработка строится вокруг подробных спецификаций и заранее определенных критериев результата. На этой основе **команда (или ИИ-агент):**

**1 Составляет детальную спецификацию, описывающую поведение системы**

**2 Затем формирует план реализации**

**3 Лишь после этого генерирует код**

Разработка на основе спецификаций (Spec-Driven Development) помогает на всех этапах работы сохранять единое понимание задачи, снижать количество ошибок и делать работу ИИ более предсказуемой и управляемой.[4] Разработку на основе спецификаций характеризуют как «одну из наиболее значимых практик, возникших в 2025 году» в области разработки с применением ИИ.[5]

1. По данным Microsoft: microsoft.com
2. По данным Microsoft: microsoft.com
3. По данным Microsoft: microsoft.com
4. По данным BCMS: thebcms.com
5. По данным Thoughtworks: thoughtworks.com

**Отличие Spec-Driven Development от вайб-кодинга** заключается в явном разделении этапов разработки:

**1 Планирование**

Сначала подробно описывается, как должна работать система: какие данные она получает, какой результат выдает, какие правила и ограничения должны соблюдаться.

**2 Реализация**

После согласования требований ИИ приступает к написанию кода.

**2026**

К 2026 году **поддержку SDD-рабочих процессов внедрили** большинство крупных компаний-разработчиков ИИ-инструментов для ИТ-разработки, а также ряд специализированных ИИ-платформ.[1]

KIRO CURSOR Claude Code Google Antigravity

## Сергей Полиненко

**Директор по продукту платформы «Сфера», ИТ-холдинг Т1**

*Индустрию ИТ-разработки ждет вторая волна ИИ-трансформации: от точечного применения технологии отрасль перейдет к ИТ-производству с опорой на искусственный интеллект на всех этапах. Типовые операции возьмут на себя ИИ-агенты с оркестрацией через единый интерфейс и web chat, а привычную ролевую модель заменят гибридные кросс-функциональные команды. Риски же использования ИИ в разработке связаны не столько с ошибками в коде. Гораздо важнее системные вопросы безопасности - от утечек данных через промпты до использования компонентов с неясной лицензией.*

*Мы в ИТ-холдинге Т1 планируем реализовать следующий подход в «Сфере»: цифровые агенты платформы будут основаны на российском коде и работать по слепку ролевой модели человека, а развертывание осуществляться на локальных серверах (on premise). Агенты будут полностью интегрированы с привычными инструментами платформы, что позволит сделать процесс производства ПО по настоящему ИИ-ориентированным на всем жизненном цикле ИТ-разработки.*

# АГЕНТНЫЙ ИНЖИНИРИНГ: ОТ АССИСТЕНТА К АВТОНОМНОМУ ИСПОЛНИТЕЛЮ

С начала 2026 года профессиональное сообщество предложило уточнение концепции вайб-кодинга – термин **«агентный инжиниринг» (agentic engineering)**[2]. Если вайб-кодинг описывал разработку через принятие кода без его детального понимания, то агентный инжиниринг предполагает осознанную оркестрацию.

При использовании агентного инжиринга разработчик не пишет код напрямую, а **управляет ИИ-агентами**, которые декомпозируют задачи, выполняют их и отчитываются о результате.

1. По данным BCMS: thebcms.com
2. По данным The New Stack: thenewstack.io

Агентный инжиниринг реализуется через несколько **типов агентов** с разными функциями:

ТИПЫ АГЕНТОВ

| | |
|---|---|
| **Агент-оркестратор** (Orchestrator Agent) | Получает высокоуровневую задачу, декомпозирует её и распределяет между агентами-исполнителями. |
| **Агенты-исполнители** (Worker Agents) | **Агенты-исполнители специализированы:** Один генерирует код, другой пишет тесты, третий анализирует уязвимости, четвёртый взаимодействует с системами деплоя. |
| **Человек-разработчик** (Human Developer) | Выступает как архитектор системы агентов, и финальный валидатор результата. |

В отличие от ИИ-ассистента, который отвечает на конкретный запрос и ждет следующего, **ИИ-агент действует автономно в рамках заданной цели**.

1. **самостоятельно определяет последовательность шагов**
2. **вызывает внешние инструменты***
3. **обрабатывает промежуточные результаты**
4. **корректирует план при необходимости**

* Запускает тесты, обращается к базе кода, читает документацию, открывает pull request

По данным аналитиков, в 2025 год **62% организаций экспериментировали с агентными системами** в разработке, однако **только менее 25% вывели их в производство**[1]. Среди основных барьеров: непредсказуемость поведения агентов в нетипичных ситуациях, сложность отладки цепочек действий и значительный рост токен-потребления.

***Именно способность к многошаговому планированию без непрерывного участия человека отличает агентный инжиниринг от всех предшествующих форм разработки с применением ИИ.***

## КАК ИЗМЕНИЛАСЬ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ПРИХОДОМ ИИ

| | РАНЬШЕ | СЕЙЧАС |
|---|---|---|
| КАК СТАВИТСЯ ЗАДАЧА | Требования формализуются письменно, проходят согласование, потом передаются разработчику | Задачу описывают на обычном языке, ИИ сразу предлагает решение |
| КТО ПИШЕТ КОД | Разработчик пишет вручную строку за строкой | Разработчик ставит задачу, ИИ генерирует черновик, человек проверяет и правит |
| СКОЛЬКО ВРЕМЕНИ ЗАНИМАЮТ РУТИННЫЕ ОПЕРАЦИИ | Написание шаблонного кода, документации, тестов занимает дни и недели | Те же задачи выполняются за часы или минуты |
| КАК УСТРОЕН ЦИКЛ РАЗРАБОТКИ | Последовательно: идея – проект – разработка – тест – запуск | Итеративно и параллельно: этапы перекрываются, прототип появляется почти сразу |
| РОЛЬ РАЗРАБОТЧИКА | Создает код и является главным исполнителем | Принимает решения, проверяет результат ИИ |
| ГДЕ НАКАПЛИВАЕТСЯ СЛОЖНОСТЬ | В написании кода | В постановке задачи, проверке качества и управлении результатом |

1. По данным McKinsey: mckinsey.com

Ключевой управленческий вызов, возникающий в результате внедрения ИИ, – поиск баланса между скоростью и качеством. Ускорение разработки само по себе не является ценностью, если оно приводит к росту ошибок, технического долга или снижению устойчивости систем.

## Николай Козак

**Член правления, Управляющий директор ДОМ.РФ,**
**Заместитель председателя Правления, Банк ДОМ.РФ**

*Использование искусственного интеллекта колоссально ускоряет процессы разработки: от написания программного кода до тестирования и DevOps. В рамках пилота мы в ДОМ.РФ запустили ИИ-агента для автоматической проверки качества кода – он анализирует код на предмет безопасности и соответствия стандартам ещё до передачи его руководителям ИТ-команд, даёт развернутую обратную связь. Агентный инжиниринг не только повышает качество кода, но и позволяет сэкономить до 30% времени на рутинных проверках, увеличивая общую эффективность команд разработки на 35%.*

КЕЙС №3

ДОМ.РФ | Технологии

## ИИ КАК НОВЫЙ КОНТУР РАЗРАБОТКИ ПО

**ДОМ.РФ Технологии** рассматривает применение ИИ в разработке ПО как переход от точечной помощи разработчику к изменению самого процесса создания программного продукта. В этой эволюции можно выделить три уровня зрелости, которые развивались с момента появления больших языковых моделей (LLM).

## ТРИ УРОВНЯ ЗРЕЛОСТИ ИСПОЛЬЗОВАНИЯ LLM-МОДЕЛЕЙ В РАЗРАБОТКЕ

### Первый уровень – чаты с LLM

На базовом уровне разработчики используют языковые модели как справочный инструмент: задают вопросы по ошибкам в коде, просят объяснить фрагмент реализации, подобрать подход или предложить вариант исправления. Сценарий аналогичен поиску информации об ошибке в интернете или на ресурсах типа Stack Overflow и т.п. Это самый простой вариант автоматизации, который помогает ускорить решение локальных задач, но не меняет сам процесс разработки.

### Второй уровень – LLM-инструменты, встроенные в IDE-среду разработки[1]

На следующем уровне ИИ становится частью рабочей среды разработчика. Например, плагин с LLM-сервисом, добавленный в IDE-среду разработки. Такие инструменты работают в контексте кодовой базы: отвечают на вопросы по проекту, помогают быстрее ориентироваться в существующем коде и дописывают фрагменты реализации после начала работы. На этом уровне виден измеримый прирост продуктивности на ориентировочно 20-30% в отдельных типах задач. Это существенный эффект, но он остаётся именно процентным улучшением существующего процесса.

1. IDE (Integrated Development Environment или интегрированная среда разработки) — это главная программа для программиста. Она собирает в одном окне всё нужное для создания программ: редактор текста, инструменты для проверки и запуска кода.

## Третий уровень – ИИ-агенты для написания кода

Наиболее перспективный и одновременно наиболее рискованный уровень – агентные CLI-инструменты[1], которые получают постановку задачи, самостоятельно вносят изменения в код, запускают проект, исправляют ошибки, формируют тест-план и проверяют результат. Здесь эффект может выражаться уже не в процентах, а в кратном ускорении. В пилотах ДОМ.РФ Технологии отдельные хорошо формализованные задачи с оценкой до двух рабочих дней могли быть доведены до рабочего результата примерно за 30 минут. При этом говорится не о прототипе или черновом варианте реализации, а промышленном коде, который прошёл процедуру code review, реализованную человеком.

При этом такой эффект нельзя считать универсальным для всех задач и проектов. Он зависит от качества постановки, зрелости кодовой базы, наличия тестов, архитектурных ограничений и уровня контроля со стороны инженерной команды.

**Внедрение ИИ-агентов для написания кода требует особенно осмысленного подхода.** ИИ-агенты способны вносить в проект значительные изменения, что резко увеличивает нагрузку на тимлидов и ревьюеров кода. Если раньше разработчик мог писать сотни строк промышленного кода в день, то теперь merge request может включать сотни или тысячи изменений.

**Узкое место смещается от написания кода к его ревью, тестированию, архитектурной проверке и управлению техническим долгом.**

**Отдельный риск использования ИИ-агентов для разработки** связан с размыванием инженерного владения кодовой базой. У тимлида появляется соблазн использовать отдельного ИИ-агента уже для проведения code review. В предельном случае за несколько месяцев команда может получить радикально изменённую кодовую базу, в которой разработчики всё хуже понимают причины архитектурных и продуктовых решений. Поэтому ключевая задача – не просто ускорить написание кода, а сохранить управляемость разработки.

**ДОМ.РФ Технологии внедряет ИИ-агентов контролируемо**, на выбранной группе проектов, с обязательным инженерным надзором, правилами ревью, ограничениями по типам задач и оценкой влияния на качество кода. Игнорировать такие инструменты уже невозможно: когда отдельная задача вместо двух рабочих дней решается за полчаса, ИИ становится не экспериментом, а фактором конкурентоспособности разработки, но внедрять его нужно как управляемую трансформацию инженерного процесса, а не как массовую установку нового инструмента.

## Результаты использования агентного инжиниринга «ДОМ.РФ Технологии»

- **20-30%** – ориентировочный прирост продуктивности от LLM-инструментов, встроенных в IDE в режиме ИИ-помощника разработчика, на отдельных типах задач ИТ-разработки.
- **x3 и выше** – пример эффекта от применения ИИ-агента в пилотном сценарии.
- **Кратный рост изменений** (Merge Request) кодовой базы.
- **4-6 месяцев** активного использования ИИ-агентов – горизонт, на котором может проявиться риск накопления неконтролируемых изменений в кодовой базе.

*"...когда отдельная задача вместо 2-х рабочих дней решается за 30 минут, ИИ становится не экспериментом, а фактором конкурентоспособности разработки..."*

1. CLI (интерфейс командной строки) — это текстовый инструмент для управления программами и операционной системой с помощью ввода символьных команд.

## Григорий Грязнов

**Директор по исследованиям и разработке**
**ООО «ДОМ.РФ Технологии»**

*ИИ в разработке – это уже не просто удобный помощник для программиста, а изменение модели производства ПО. Но кратный рост скорости имеет смысл только тогда, когда сохраняется управляемость архитектуры, качество кода и ответственность инженерной команды. Главный риск ИИ-агентов не в том, что они пишут плохой код, а в том, что они пишут много кода очень быстро.*

*Наша задача – встроить эти инструменты так, чтобы ускорение разработки не привело к потере понимания системы.*

## МЕТОДЫ ИСПОЛЬЗОВАНИЯ ИИ В ИТ-РАЗРАБОТКЕ

В 2024-2025 годах понятийное поле вокруг ИИ в разработке быстро расширялось, и разные термины нередко используются как синонимы, хотя описывают разные аспекты и уровни применения ИИ.

СКАЧОК В ПРИМЕНЕНИИ ИИ В РАЗРАБОТКЕ ПО

| Понятие | Суть |
|---|---|
| **AI-Assisted Development** | ИИ как помощник разработчика в отдельных задачах |
| **AI-Augmented Development** | ИИ усиливает возможности разработчика в существующих процессах, не меняя их архитектуру |
| **Вайб-кодинг** | Разработка через естественный язык без детального контроля над кодом |
| **VibeOps** | Перенос принципов вайб-кодинга на операционный уровень |
| **Разработка на основе спецификаций (Spec-Driven Development)** | Формализованные спецификации как основа для ИИ-генерации кода |
| **AI-Driven Development (AIDD)** | ИИ встроен как центральный участник процесса разработки – от замысла до эксплуатации |
| **Агентный инжиниринг** | Оркестрация ИИ-агентов для выполнения многошаговых задач |

# ШКАЛА ЭВОЛЮЦИИ ИИ-ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ

## 2021 НАЧАЛО ЭПОХИ: ПЕРВЫЙ ПРОМЫШЛЕННЫЙ ИИ-АССИСТЕНТ

ИИ впервые появляется в редакторе кода как **инструмент автодополнения**. Пока это пилотная версия, доступная ограниченному кругу разработчиков.

**GitHub Copilot** вышел в виде плагина для **JetBrains** и **Neovim**

GitHub Copilot

## 2022 МАССОВЫЙ ДОСТУП И ПОЯВЛЕНИЕ CHATGPT

Формируется **два параллельных трека**: специализированный ИИ в интегрированной среде разработки (Copilot) и универсальный диалоговый ИИ (ChatGPT).

Разработчики начинают переключаться между ними в зависимости от задачи.

**GitHub Copilot** – первый массовый ИИ-ассистент для написания кода.

OpenAI выпустила **ChatGPT**, который разработчики сразу стали использовать для объяснения и отладки кода и генерации его фрагментов.

GitHub Copilot ChatGPT

## 2023 РАСШИРЕНИЕ РЫНКА

Тема ИИ в разработке выходит за пределы GitHub, у каждого крупного игрока появляется собственный ответ. Формируется конкурентный рынок ИИ-ассистентов разработки.

В России появляется первое отечественное решение от Сбера.

Запущен **Cursor**, первая серьезная альтернатива GitHub.

Открыт публичный доступ к **API Claude** – первой модели, составившей конкуренцию GPT-4 в задачах написания и анализа кода

Сбер запустил **GigaCode**, ставший первым российским ИИ-ассистентом для написания кода.

CURSOR Claude GIGA CODE

## 2024 АГЕНТНАЯ РЕВОЛЮЦИЯ

Появляются автономные ИИ-агенты, которые не просто помогают писать код, а пишут его сами. Формируется рынок «вайб-кодинга», т.е. создания приложений без программирования.

Вышел **Devin**, первый полностью автономный ИИ-разработчик, способный самостоятельно выполнять весь цикл разработки.

Запущен **Yandex Code Assistant** – ИИ-ассистент для разработки с автодополнением кода.

Выходят платформы вайб-кодинга, например, **Bolt** и **v0**.

Devin bolt.new v0 Yandex Code Assistant

## 2025 ГОД АГЕНТОВ И СТАНДАРТИЗАЦИИ

Агентные системы переходят из экспериментального в рабочий режим. Появляется запрос на методологию и стандартизацию.

Сбербанк представил **GigaStudio** – инструмент для создания веб-приложений.

**AWS** открыла исходный код методологии AI-DLC (AI-Driven Development Lifecycle), первая попытка формализовать концепцию AIDD.

GIGA STUDIO aws

## 2026 КОНСОЛИДАЦИЯ И АГЕНТНЫЙ МЕЙНСТРИМ

Рынок переходит к консолидации. Агентный инжиниринг становится стандартным рабочим процессом для профессиональных разработчиков.

Anthropic запустила **Cowork** - **«Claude Code** для общих вычислительных задач», написанный преимущественно самим Claude Code за 10 дней.

**Cursor** используют более 5 млн разработчиков в мире.

CURSOR Cowork

КЕЙС №4

# SourceCraft

**SourceCraft –** платформа полного цикла разработки, где репозитории, список задач, CI/CD и облачное развёртывание собраны в одной среде.

## КАК УСТРОЕНА РАБОТА НА ПЛАТФОРМЕ SOURCECRAFT С ИИ-АГЕНТОМ:

**Разработчик**

- Ставит задачи
- Управляет процессом
- Проверяет и принимает результат

**ИИ-агент**

SOURCECRAFT CODE ASSISTANT

- Создает репозиторий
- Пишет код
- Проводит тесты
- Готовит слияния
- Публикует изменения

Агент подключается к трекерам, облаку и мониторингу через MCP и сам выполняет действия в этих системах.

SourceCraft делает командные практики воспроизводимыми: повторяющиеся сценарии превращаются в AI-навыки (skills) — версионируемые инструкции и контекст для агента, которые команды переиспользуют между репозиториями.

## РЕЗУЛЬТАТЫ SOURCECRAFT ЗА ПЕРВОЕ ПОЛУГОДИЕ 2026 ГОДА:

более чем **в 20 раз вырос объём** совместной работы с кодом

**435,5** млрд токенов - потребление SourceCraft Code Assistant

*Такой скачок отражает **переход агентного режима из пилотных сценариев в основной инструмент повседневной разработки.***

**Следующий сдвиг в инженерии – не более умная генерация отдельных фрагментов кода, а гибридные команды, где цифровые исполнители берут длинные многошаговые задачи, а человек управляет этой работой.**

**Дмитрий Иванов**

**руководитель платформы SourceCraft**

*Хороший код по-прежнему основа продукта, меняется вклад разработчика: он всё чаще выигрывает умением ставить задачу агентам, оркестрировать их работу, вовремя вмешаться и принять или отклонить результат. Это возможно только при ясном описании того, что продукт должен делать, и понимании, что у агента получилось правильно. Без точно заданных ожиданий и правил приёмки агентная разработка не масштабируется – остаётся лишь править поток сгенерированного кода*

# 02.

# ИИ В ЖИЗНЕННОМ ЦИКЛЕ ПРОДУКТОВОЙ РАЗРАБОТКИ

# ИСПОЛЬЗОВАНИЕ ИИ В ПРОЦЕССАХ ИТ-РАЗРАБОТКИ В ФИНТЕХЕ[1]

**52%**
проходят пилоты/эксперименты по внедрению ИИ-решений

**44%**
активно используют ИИ в ИТ-разработке

**4%**
ИИ не используется, но планируется

** Также предлагался вариант «Не используется и не предполагается», его выбрало 0 участников*

**Развитие рынка ИИ-инструментов для разработки за 2023-2025 годы прошло несколько этапов:**

1. Первый этап заключался в экспериментах с ChatGPT и аналогами для разовых задач: объяснить код, сгенерировать фрагмент, написать документацию.
2. Второй этап – распространение ИИ-ассистентов, встроенных непосредственно в интегрированную среду разработки (GitHub Copilot, Cursor, Amazon Q Developer и другие).
3. Третий, только набирающая силу, этап развития ИИ для разработки - это появление агентных систем, способных самостоятельно выполнять многошаговые задачи: читать задачи из трекера, писать код, запускать тесты, создавать pull request.

***Опыт разрозненных экспериментов с ИИ-инструментами и методами разработки от вайб-кодинга до агентных систем привёл индустрию к пониманию необходимости единой методологии применения ИИ в разработке ПО. Во многом это повторяет путь, пройденный в 1990-х годах, когда стихийные практики программирования постепенно оформились в стандартизированный жизненный цикл разработки (Software Development Life Cycle, SDLC)***

Внедрение ИИ-инструментов затрагивает не отдельные этапы разработки, а весь жизненный цикл ИТ-разработки (Product Development Life Cycle) от определения бизнес-требований до эксплуатации.

Российские компании, использующие ИИ в ИТ-разработке, **сокращают время выхода продукта на рынок на 20-40% и снижают затраты на разработку на 20-30%**[2]. При этом эффект неравномерен, одни этапы внутри процесса ИТ-разработки радикально ускоряются, а другие, наоборот, усложняются.

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.
2. По данным McKinsey: mckinsey.com

# ТЕКУЩИЙ УРОВЕНЬ ПРОНИКНОВЕНИЯ ИИ В РАЗРАБОТКУ

**46%**
всего нового кода в США пишется с помощью ИИ[1]

**51%**
разработчиков применяют ИИ ежедневно[2]

**92%**
разработчиков используют или планируют использовать ИИ-инструменты в своей работе[3]

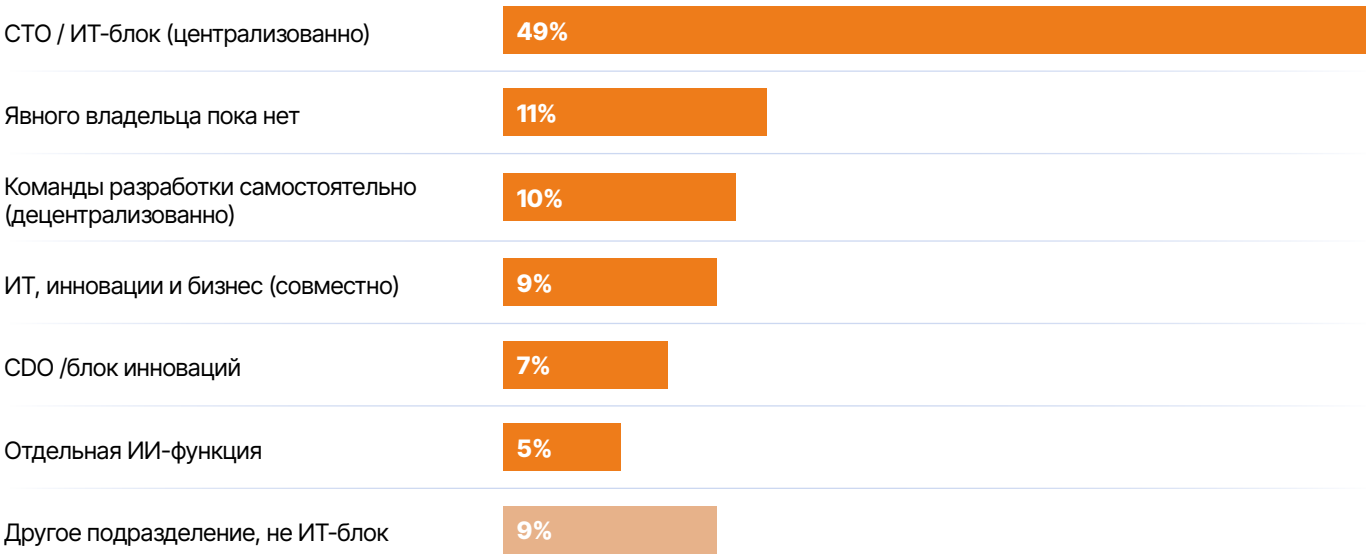
В 2026 году ИИ-ассистенты для написания кода перешли из категории нишевых экспериментов в стандартный инструмент разработчика. Положительное отношение к ИИ-инструментам среди разработчиков, тем не менее, снизилось: с более чем 70% в 2023-24 гг. до 60% в 2025 году, вероятно, из-за расхождения между ожиданиями и реальными результатами[4].

Наиболее распространенные сценарии использования ИИ в разработке - это написание и автодополнение кода, документирование и генерация тестов. При этом в задачах с высокой ответственностью разработчики предпочитают не использовать только ИИ, а полагаются на свой естественный интеллект. По оценкам аналитиков, 76% ИТ-специалистов не планируют использовать ИИ для развертывания и мониторинга кода, а 69% – для проектного планирования[5].

***Несмотря на стремительный рост использования ИИ в повседневной разработке, разработчики по-прежнему с осторожностью доверяют ему выполнение задач с высокой степенью ответственности, предпочитая сохранять за человеком ключевые этапы принятия решений.***

## КТО ОТВЕЧАЕТ ЗА ВНЕДРЕНИЕ ИИ В ПРОЦЕССЫ РАЗРАБОТКИ?

**Подразделение, ответственное за внедрение ИИ в процессы разработки, внутри компаний[6]**




1. По данным Guvi: guvi.in
2-5. По данным Stack Overflow Developer Survey: survey.stackoverflow
6. По результатам опроса АФТ по сост. на 2 кв. 2026 г.

# ОБЩАЯ ЛОГИКА ТРАНСФОРМАЦИИ ИТ-РАЗРАБОТКИ

**ИИ упрощает разработку там, где много рутинных операций, и расширяет возможности** там, где раньше требовались значительные ресурсы для первичного черновика. ИИ не устраняет необходимость в экспертизе, а смещает ее с написания кода на постановку задач, архитектурные решения и контроль качества.

Классическая модель жизненного цикла ИТ-разработки строилась вокруг последовательного прохождения этапов: сбор требований, проектирование, разработка, тестирование, внедрение. Несмотря на различные вариации (waterfall, agile), ключевой характеристикой этой модели оставалась структурированность и относительная жесткость переходов между этапами.

ИИ размывает границы между стадиями традиционного жизненного цикла ИТ-разработки. Генерация кода может происходить одновременно с уточнением требований, тестирование параллельно с разработкой, а прототипирование – практически мгновенно после появления идеи. Это приводит к формированию нелинейной модели разработки, где этапы не следуют друг за другом, а переплетаются в рамках единого процесса.

Особенно заметным становится **изменение точки входа в ИТ-разработку**. Если раньше она начиналась с формализации требований, то теперь – с **формулирования «намерения»**. Это может быть описание бизнес-задачи, пользовательского сценария или даже просто идеи. ИИ выступает интерпретатором, преобразуя это намерение в конкретные технические решения.

## ВЛИЯНИЕ ИИ НА ЖИЗНЕННЫЙ ЦИКЛ ИТ-РАЗРАБОТКИ

1. Исследование и постановка задач
2. Проектирование и архитектура
3. Реализация и написание кода
4. Тестирование и контроль качества
5. Развертывание и эксплуатация

### Кирилл Меньшов

**Старший вице-президент, руководитель блока «Технологии» Сбера**

*Мы видим, что сегодня конкуренция смещается из плоскости применения ИИ-моделей в плоскость перестройки жизненного цикла разработки. Доступ к ИИ-инструментам есть у всех, но преимущество даёт только системное внедрение на всех этапах – от анализа требований до сопровождения. Подход AI-Disrupt PDLC – это ответ на вызов, как перестроить компанию вокруг новой технологии, изменив роли, процессы и платформу вокруг намерения.*

*Когда исполнение становится моментальным, дефицитным ресурсом остаётся качество управленческого замысла. В ближайшие годы в выигрыше будут компании, которые формируют зрелую ИИ-культуру, переосмысляют процессы и сохраняют человеческую экспертизу. Мы убеждены: агентная разработка станет отраслевым стандартом, а её масштабирование укрепит технологический суверенитет России. Поэтому Сбер развивает концепцию AI-Disrupt PDLC и открыто делится методологией с рынком*

1. По данным McKinsey: mckinsey.com

СБЕР

## AI-DISRUPT PDLC – ПЕРЕХОД ОТ НАПИСАНИЯ КОДА К УПРАВЛЕНИЮ «НАМЕРЕНИЕМ»

**Сбер** рассматривает генеративный ИИ как основу новой модели разработки программного обеспечения - AI-Disrupt PDLC (AI-Driven Development Lifecycle), в которой меняется не отдельный этап SDLC, а весь процесс создания продукта. Если классическая разработка строится вокруг написания кода человеком, то AI-Disrupt PDLC переносит центр процесса ИТ-разработки на формирование "намерения" (intent), а реализацию передает ИИ-агентам под контролем инженера.

**AI-Disrupt PDLC** – методология, при которой ИИ работает сквозным слоем на всех этапах разработки от уточнения требований до кода, тестирования и документации. В банке ИИ уже встроен в большинство этапов, но финальное решение остаётся за человеком. Основу трансформации составляет экосистема **GigaCode**, **GigaIDE** и **GitVerse**, развёрнутая внутри периметра компании.

### Приниципы AI-Disrupt PDLC:

### Первый принцип – намерение вместо кода

В новой модели основным артефактом становится не исходный код, а спецификация, описывающая, какой результат необходимо получить. Инженер концентрируется на постановке задачи, архитектурных решениях и критериях качества, а реализация постепенно превращается в задачу агентной системы. Такой подход позволяет перенести основную ценность разработки с программирования на инженерное проектирование продукта.

### Второй принцип – две взаимосвязанные петли разработки

AI-Disrupt PDLC разделяет процесс создания программного обеспечения на **два контура**:

1. **Петля намерения (Intent Loop)** отвечает за формирование требований, архитектуры и бизнес-логики, где ключевую роль сохраняет человек.
2. **Петля реализации (Implementation Loop)** включает ИИ-агентов, которые самостоятельно декомпозируют задачи, генерируют код, проводят тестирование, исправляют ошибки и готовят результат к проверке. Финальное решение о принятии изменений остается за инженером.

### Третий принцип – единая агентная платформа

Работа агентов строится не как использование отдельных ИИ-сервисов, а как функционирование единой инженерной платформы. Она обеспечивает доступ к корпоративному контексту, применение внутренних стандартов, взаимодействие с DevOps-инструментами, контроль безопасности и аудит действий ИИ. За счет этого агент работает как полноценный участник процесса разработки, а не как изолированный генератор кода.

### Четвертый принцип – управление контекстом вместо промптов

Для промышленной разработки критическим становится не качество отдельных запросов к LLM, а качество инженерного контекста. Методология предусматривает централизованное управление знаниями, документацией, архитектурными ограничениями и корпоративными правилами, которые автоматически используются агентами при выполнении задач. Это позволяет уменьшить вероятность ошибок и обеспечить воспроизводимость результатов.

## Ключевые особенности методологии AI-Disrupt PDLC

- **Разработка через спецификацию (Specification-Driven Development)** – спецификация становится главным артефактом разработки, а код рассматривается как производный результат.
- **Управление как код (Policy as Code)** – требования безопасности, архитектуры и внутренние стандарты автоматически применяются в процессе работы агентной системы.
- **Работа с машиночитаемым контекстом (Context Engineering)** – качество результата определяется полнотой и структурой инженерного контекста, доступного ИИ.
- **Роль человека в контуре принятия решений ИИ (Human-in-the-Loop)** – человек сохраняет ответственность за архитектуру, принятие решений и итоговое качество продукта, несмотря на высокий уровень автоматизации.

Подробнее о методологии **AI-Disrupt PDLC Сбера** можно узнать здесь:

## 1. ИССЛЕДОВАНИЕ И ПОСТАНОВКА ЗАДАЧ

ИИ трансформирует подход к постановке задач и их приоритизации. Генеративные модели позволяют быстро синтезировать пользовательские интервью, анализировать фидбек в больших объемах и генерировать черновики пользовательских историй.

**на 40%** увеличивается производительность продуктовых менеджеров при использовании ИИ для документирования и структурирования требований[1]

Вместе с тем появляется риск перехода к **«вайб-исследованиям»** (Vibe-researching), то есть замены реальных пользователей синтетическими ИИ-персонами, что может приводить к систематическим ошибкам на входе в цикл.

## 2. ПРОЕКТИРОВАНИЕ И АРХИТЕКТУРА

На этапе дизайна ИИ выступает как инструмент быстрого прототипирования и позволяет выполняет генерацию вариантов архитектурных решений и их оценку, создание диаграмм и технических спецификаций. **Однако именно здесь человеческая экспертиза остается незаменимой,** поскольку ИИ хорошо справляется с типовыми паттернами, но не с нестандартными ограничениями конкретной организации – требованиями безопасности, интеграционными зависимостями, регуляторными рамками и пр.

**ИИ-инструменты нужно снабжать детальным контекстом через промпты, описывающие** *(1)* как будет использоваться код, *(2)* с какими системами интегрируется, *(3)* какие требования безопасности применимы и т.д.

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.

## 3. РЕАЛИЗАЦИЯ И НАПИСАНИЕ КОДА

Реализация решений - наиболее зрелая и изученная фаза внедрения ИИ. Написание и автодополнение кода, генерация черновиков функций, помощь с рефакторингом и отладкой стали стандартной практикой.

### НАИБОЛЕЕ ПОПУЛЯРНЫЕ СЦЕНАРИИ ИТ-РАЗРАБОТКИ С ПРИМЕНЕНИЕМ ИИ[1]


## 4. ТЕСТИРОВАНИЕ И КОНТРОЛЬ КАЧЕСТВА КОДА

Генерация тест-кейсов и автоматизация тестирования – одни из наиболее перспективных направлений применения ИИ в жизненном цикле разработки продукта.

**Рост объема ИИ-генерированного кода создает новую проблему:** ревью кода и валидация становятся узким местом всего цикла ИИ-разработки.

*Главный риск ИИ-агентов не в том, что они пишут плохой код, а в том, что они пишут много кода и очень быстро.*

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.
2. По данным Stack Overflow Developer Survey: survey.stackoverflow

# 5. РАЗВЕРТЫВАНИЕ И ЭКСПЛУАТАЦИЯ

На этом этапе **сопротивление использованию ИИ максимально**. Согласно опросам, 76% разработчиков не планируют применять ИИ для развертывания и мониторинга, 69% – для проектного планирования[1]. Причиной является высокая ответственность, поскольку цена ошибки здесь несопоставимо выше, чем на ранних этапах. Тем не менее ИИ уже применяется в наблюдаемости (Observability), т.е. анализе логов, выявлении аномалий, а также в анализе коренных причин (Root Cause Analysis), то есть там, где задача сводится к поиску паттернов в больших объемах данных.

## В КАКИХ ПРОЦЕССАХ ИТ-РАЗРАБОТКИ НАИБОЛЕЕ ЭФФЕКТИВНО ПРИМЕНЯЕТСЯ ИИ?

| | |
|---|---|
| **Высокий эффект** | создание шаблонного кода, перевод между форматами, написание тестов по уже существующему коду, генерация документации, рефакторинг по заданному паттерну |
| **Средний эффект** | поиск ответов на вопросы, генерация контента или синтетических данных и изучение новых концепций или технологий.[1] |
| **Низкий эффект** | проектирование архитектуры сложных систем, работа с legacy-кодом без четкой документации, отладка тонких concurrency-проблем, обеспечение безопасности в условиях специфических регуляторных требований. |

Здесь ИИ выступает скорее помощником в поиске и анализе, чем автором решения.

1. По данным Stack Overflow Developer Survey: survey.stackoverflow

# КАКИЕ РЕШЕНИЯ ИСПОЛЬЗУЮТСЯ НА КАЖДОМ ЭТАПЕ ИТ-РАЗРАБОТКИ?

## КАРТА ИИ-РЕШЕНИЙ, ПРИМЕНЯЕМЫХ НА ЭТАПАХ РАЗРАБОТКИ*

### 1. Исследование и постановка задач
### 2. Проектирование и архитектура

**Постановка задач**

GitHub Copilot

**Исследования рынка**

perplexity | ChatGPT | Claude

**Требования и документация**

JetBrains AI | Notion | Confluence

**Диаграммы, дизайн**

miro | FigJam

**Spec-Driven Dev**

KIRO

### 3. Реализация и написание кода

GitHub Copilot | CURSOR | Lovable | Devin | SourceCraft | bolt.new | OpenAI | GIGA CODE | SWE-agent | Replit | Codex CLI | V0 | GIGA IDE | Claude Code | JetBrains AI

### 4. Тестирование и контроль качества

GitHub Copilot | CURSOR | codium ai | snyk | GIGA CODE | SourceCraft | Amazon Q | JetBrains AI | qodo

### 5. Развертывание
### Эксплуатация

GitHub Copilot | aws | harness | new relic | GitHub Actions | dynatrace | Grafana | DATADOG | PagerDuty | Devin | Claude Code | SourceCraft

*Карта не является исчерпывающей и отражает основные ИИ-решения, наиболее часто применяемые в ИТ-разработке*

На этапе **написания кода** сосредоточено большинство ИИ-решений, на этапах развертывания и эксплуатации их меньше. Большинство разработчиков не использует ИИ для развертывания кода.

Самые мощные ИИ-инструменты перекрывают **сразу 3-4 этапа разработки**, то есть один инструмент ведет задачу от написания кода до его развертывания.

**Отчественные ИИ-инструменты** используются в ключевых этапах написания кода и тестирования, но в развертывании и эксплутации чаще всего применяются иностранные решения.

* По результатам опроса российскийх компании, АФТ (по сост. на 2 кв. 2026 г.)

КЕЙС №6

СБЕР GITVERSE

## ЭКОСИСТЕМА ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТЧИКОВ

**Сбер** развивает экосистему инструментов для разработчиков на базе платформы **GitVerse** – российского хостинга репозиториев со средствами непрерывной интеграции и развертывания (CI/CD), код-ревью и встроенным сканированием на уязвимости. Платформой пользуются свыше 220 тыс. разработчиков, на ней размещено более 140 000 проектов. Функциональность платформы GitVerse усилена тремя ИИ-инструментами:

- GIGA CODE **GigaCode** – ИИ-ассистент с агентным режимом и поддержкой более 35 языков, встраиваемый в VS Code, JetBrains и GigaIDE. ИИ-ассистент разработчика GigaCode CLI дает возможность получить доступ к более чем 30 AI-моделей для генерации и проверки кода, проведения тестов, рефакторинга.
- GIGA IDE **GigaIDE** – гибридная (облачная и десктопная) среда разработки, включающая 7 ИИ-агентов для автоматизации разработки и сопровождения кода .
- GIGA STUDIO **GigaStudio** – мультиагентная платформа для создания веб-приложений в режиме диалога с ИИ, в рамках которой в 2025 году Сбер представил первого в отрасли SWE-агента*.

Весь стек разворачивается внутри периметра Сбера, что снижает зависимость от внешних сервисов и содействует достижению технологического суверенитета в российской ИТ-разработке.

### Рафаел Тонаканян

**Директор дивизиона развития и сопровождения производственного процесса (SberWorks) Сбербанка**

СБЕР

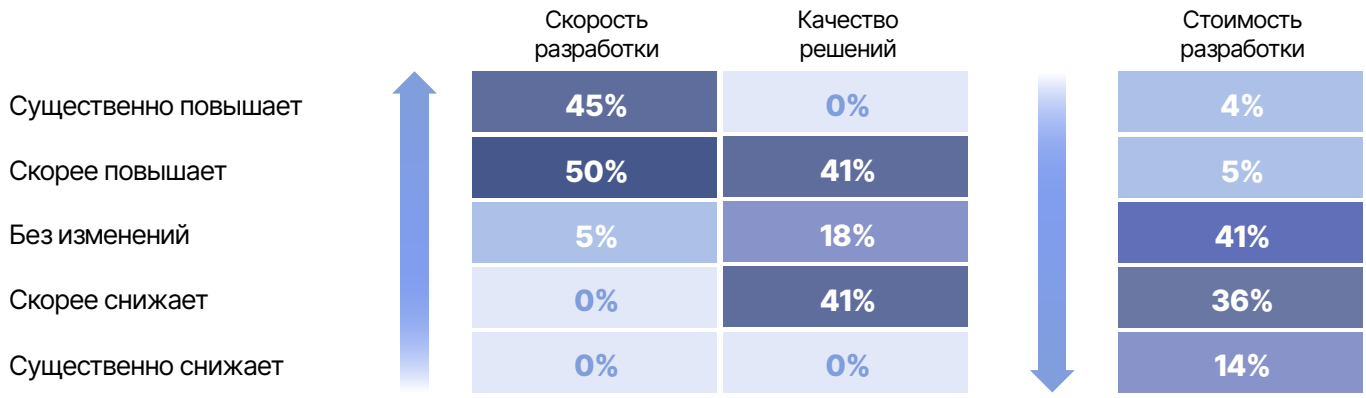
*Мы постепенно переходим от использования генеративного ИИ как персонального помощника разработчика к модели, где значительную часть инженерной работы выполняют специализированные AI-агенты. При этом главной ценностью становится не сама модель, а способность компании превращать накопленную инженерную экспертизу, архитектурные подходы и лучшие практики в воспроизводимые навыки агентных систем.*

*Мы в Сбере внедряем подход AI-Disrupt PDLC, который позволяет масштабировать опыт сильнейших команд на всю организацию, ускорять агентную разработку и одновременно обеспечивать единые требования к качеству, безопасности и сопровождению цифровых продуктов.*

*«Сегодня ключевым конкурентным преимуществом становится не доступ к ИИ, а способность встроить его в инженерную культуру и процессы компании».*

* SWE-агент (Software Engineering Agent) — это автономная система на базе искусственного интеллекта и языковых моделей (LLM), которая умеет самостоятельно находить и исправлять ошибки, дописывать код, запускать тесты и работать с репозиториями (например, на GitHub) почти так же, как живой программист

# КАК ИСПОЛЬЗОВАНИЕ ИИ ВЛИЯЕТ НА ПАРАМЕТРЫ ИТ-РАЗРАБОТКИ?[1]




Для российских финансовых организаций использование зарубежных ИИ-сервисов для работы с внутренним кодом создает риски с точки зрения информационной безопасности и регуляторных требований. Решением стало развитие собственных инструментов на базе отечественных языковых моделей, таких, как решения Яндекса и Сбера.

**~10 часов** в неделю экономит разработчик благодаря ИИ[1]

Сэкономленное время разработчики направляют на повышение качества кода, разработку новых функций, улучшение инженерной культуры и разработку документации.

**45%** разработчиков уже тратят меньше времени на написание кода[2]

**42%** разработчиков выделяют больше времени на ревью[2]

## ОТ ЧЕГО ЗАВИСИТ ЭФФЕКТ ОТ ПРИМЕНЕНИЯ ИИ В РАЗРАБОТКЕ?

**01** **Тип задач.** ИИ существенно ускоряет рутинные, хорошо структурированные задачи с четким контекстом, но в сложных, уникальных задачах в незнакомой кодовой базе эффект может быть нулевым или отрицательным. Поскольку опытные разработчики, как правило, занимаются более сложными задачами, исследования на них дают менее оптимистичные результаты, чем опросы общего характера.

**02** **Объект измерения.** Опросы фиксируют субъективное ощущение продуктивности, которое могут расходиться с объективными метриками. Контролируемые эксперименты дают более надежные данные, но ограничены по масштабу и не всегда переносимы на реальную производственную среду.

**03** **Организационный контекст.** ИИ приносит измеримую пользу только командам, у которых уже выстроены качественные процессы, то есть имеются четкие требования, хорошие практики ревью кода и надежные пайплайны*. Если эти процессы организованы слабо, то ИИ скорее обостряет проблемы, чем решает их.

**автоматизированная последовательность шагов, через которые проходит код от момента написания до развертывания*

1. По данным Atlassian: atlassian.com
2. По данным AI-DISRUPT PDLC: aipdlc.ru

**45%** компаний финансового сектора*, считает, что ИИ существенно повышает скорость разработки, при этом 41% опрошенных отмечает одновременное снижение качества.

** из числа участников АФТ*

## ПАРАДОКС ПРОИЗВОДИТЕЛЬНОСТИ

На первый взгляд, влияние ИИ на разработку выглядит однозначно положительным. Генерация кода, автоматическое написание тестов, создание документации с помощью ИИ выполняются быстрее и с меньшими затратами. Однако при переходе от отдельных операций к системе в целом картина становится более сложной.

**Парадокс производительности ИИ** заключается в том, что при очевидном ускорении отдельных задач (например, генерации кода или документации) совокупная продуктивность разработки растет не пропорционально, а в ряде случаев ограниченно или даже стагнирует.

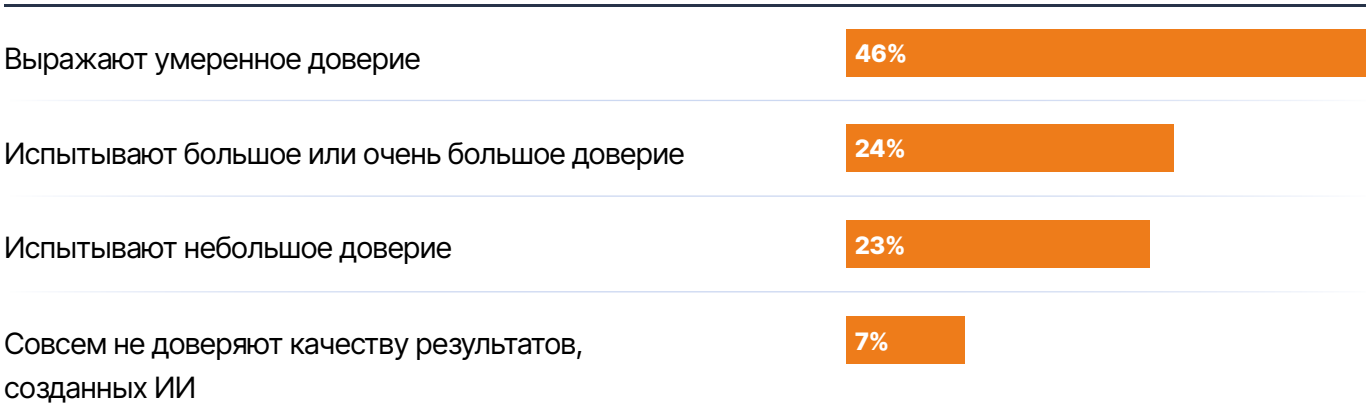
Уровень доверия к результатам, сгенерированным при помощи ИИ, можно отразить «пропорцией 70% на 30%», то есть 70% разработчиков в той или иной степени доверяют качеству результатов, созданных ИИ, а 30% - придерживаются более сдержанной позиции.

**70%**
разработчиков доверяют качеству работы с применением ИИ

**30%**
разработчиков испытывают недоверие или совсем не доверяют качеству результатов, созданных ИИ

### УРОВЕНЬ ДОВЕРИЯ РАЗРАБОТЧИКОВ К КАЧЕСТВУ РЕЗУЛЬТАТОВ, СОЗДАННЫХ ИИ[1]

**Доля разработчиков, выбравших тот или иной ответ**




*«...ИИ приносит измеримую пользу только командам, у которых уже выстроены качественные процессы, то есть имеются четкие требования, хорошие практики ревью кода и надежные пайплайны. Если эти процессы организованы слабо, ИИ скорее обостряет проблемы, чем решает их».*

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.

## Кирилл Поволоцкий

**управляющий директор**
**«БПС Инновационные программные решения»**

*Мы видим свою задачу в том, чтобы с помощью ИИ-решений перейти на новый уровень продуктивности и получать устойчивый и прогнозируемый результат на сложных задачах. Для нас важно не просто показать работу модели, а обеспечить безопасность применения, корректные ответы с устойчивой достоверностью и понятный контур использования там, где цена ошибки велика. Такие решения должны помогать бизнесу в реальной работе, а не оставаться красивой демонстрацией. Поэтому мы делаем ставку на практическую пользу, надежность и результат, который можно использовать каждый день.*



БПС ИННОВАЦИОННЫЕ РЕШЕНИЯ

## АВТОМАТИЗАЦИЯ ПРОЦЕССА АНАЛИЗА ИНЦИДЕНТОВ ПЕРВОЙ ЛИНИИ СОПРОВОЖДЕНИЯ

**Цель:** **сократить время первичного анализа инцидента и повысить качество подготовки информации для оператора.**

ИИ-ассистент сопровождения выполняет классификацию обращений, поиск решений и известных ошибок на основе исторических данных, генерацию ответов и инструкций, анализ инцидентов и первопричин, а также формирует отчетность. Ассистент работает только с заявками, на которые он назначен в качестве исполнителя, что задает контролируемую область применения.

На текущий момент решение работает для основного продукта (SVFE) , и поэтапно ассистенты вводятся для всей продуктовой линейки. Разница в сроках и подходах связана с неоднородностью стеков, структурой документации и особенностями логирования в разных продуктах.

### Работа ведется по гибкому алгоритму:

- ИИ-ассистент анализирует предоставленную информацию и если ее недостаточно, то может запросить дополнительно логи и идентификатор транзакции.
- Если вся информация предоставлена, ИИ-ассистент анализирует логи по заданному сценарию, агрегирует информацию из описания и найденных в логах ошибках, сопоставляет признаки с похожими ранее обработанными запросами, анализирует кодовую базу на соответствие спецификациям и выдает оператору сводку найденной информации с вариантами решения задачи.

До автоматизации первичный анализ сложных кейсов мог занимать до нескольких дней: обращение проходило через несколько уровней поддержки, а для локализации причины нередко приходилось подключать разработку. Сейчас время первичного анализа сократилось примерно на 70%, по статистической оценке, а объединение разрозненных записей по идентификатору транзакции для восстановления полной картины транзакции и инцидента занимает 1–2 минуты. Формирование отчетов по частоте ошибок, динамике по сервисам и топу исключений занимает около 5 минут.

**Эффект для оператора:** вместо ручного поиска по логам, исходным кодам и истории похожих кейсов он получает готовую сводку и варианты решения.

# 03.

# ТРАНСФОРМАЦИЯ РОЛЕЙ И КОМПЕТЕНЦИЙ

# ТРАНСФОРМАЦИЯ РЫНКА ТРУДА ПОД ВЛИЯНИЕМ ИИ

По данным Всемирного экономического форума[1], **86% компаний ожидают, что ИИ трансформирует их бизнес к 2030 году.** Умение работать с ИИ и большими данными возглавляют список наиболее быстро растущих компетенций. Скорость, с которой меняются требования к навыкам, опережает возможности традиционных систем образования и корпоративного обучения.

**41%**

работодателей планируют сократить сотрудников с устаревающими компетенциями[1]

**60%**

работодателей прогнозируют, что ИИ создаст и будет способствовать развитию новых специальностей[2]

**88%**

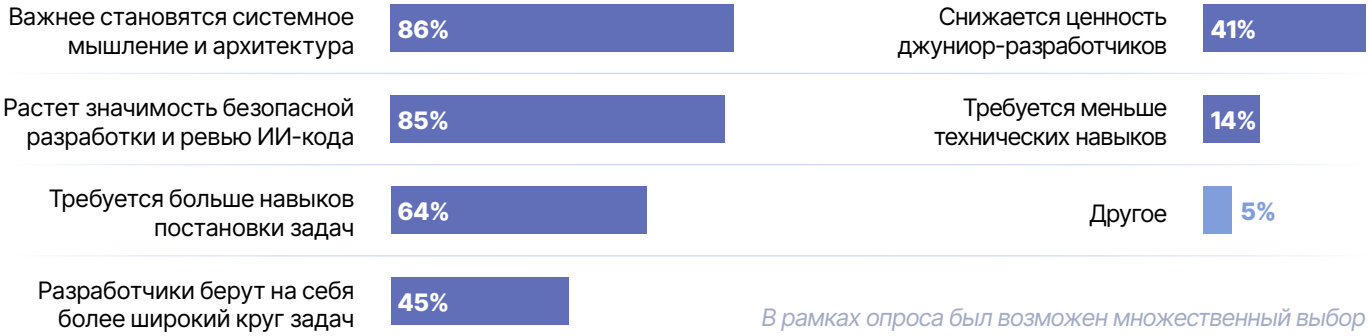
ИТ-специалистов рассматривают ИИ как путь к новым карьерным возможностям[2]

**Трансформация ИТ-разработки под влиянием ИИ - это смещение ценности с написания кода на soft-скиллы, системное мышление и умение грамотно ставить задачи**

Ключевым изменением является смещение фокуса: рутинное написание кода автоматизируется, тогда как востребованность системного мышления, архитектурных решений и надзора за ИИ-генерированным кодом растет. Несколько направлений этого сдвига особенно заметны.

- Растет значение **системного мышления и архитектурной экспертизы.** ИИ хорошо справляется с реализацией известных паттернов, но не с выбором между ними с учетом контекста конкретной системы.
- Появляется **новая компетенция – эффективная постановка задач для ИИ.** Способность точно сформулировать задачу через промпт (prompt engineering) превращается из экзотики в базовый профессиональный навык. Это не тривиальный вызов, поскольку хорошие промпты для сложных технических задач требуют глубокого понимания предметной области.
- Резко возрастает значимость **контроля ИИ-генерированного кода.** Кто-то должен понимать код, который написал агент, выявлять ошибки и уязвимости, принимать решение о приемке. В условиях, когда объем ИИ-кода растет, нагрузка на senior-разработчиков и архитекторов увеличивается, а не снижается.

## ИЗМЕНЕНИЕ ТРЕБОВАНИЙ К КОМПЕТЕНЦИЯМ ИТ-РАЗРАБОТЧИКОВ ПОД ВЛИЯНИЕМ ИИ[3]




*В рамках опроса был возможен множественный выбор*

1. По данным World Economic Forum: weforum.org
2. По данным Baires Dev: bairesdev.com
3. По результатам опроса АФТ по сост. на 2 кв. 2026 г.

КЕЙС №8

Microsoft

## РАЗДЕЛЕНИЕ ФУНКЦИЙ МЕЖДУ ИИ И ЛЮДЬМИ

Компания **Microsoft** рассматривает разработку программного обеспечения как процесс управления и контроля ИИ-систем, а не исключительно ручного написания кода. Развивая **GitHub Copilot** и связанные с ним исследования, компания продвигает модель «ограниченного делегирования» (bounded delegation), при которой ИИ выполняет значительную часть задач программирования, а разработчики формулируют требования, проверяют результаты и принимают архитектурные решения. Исследования Microsoft показали, что инженеры тратят лишь небольшую часть рабочего времени непосредственно на написание кода, что подтверждает тенденцию к смещению роли разработчика в сторону надзора за ИИ.

## НОВЫЕ РОЛИ И КОМПЕТЕНЦИИ

**Трансформация обуславливает появление принципиально новых ролей:**

- **Prompt engineering** – умение эффективно ставить задачи ИИ-инструментам
- **AI/LLM oversight** – проверка, валидация и ревью ИИ-генерированного кода
- **LLMOps** – эксплуатация и обслуживание ИИ-моделей в производственной среде
- **Безопасная разработка (secure-by-design)** – с учетом специфики ИИ-уязвимостей

**86%** Руководителей считают, что из-за внедрения ИИ в разработку для сотрудников важнее становятся системное мышление и понимание архитектуры[1]

Переход от ИИ-ассистентов к агентным системам меняет не только процесс разработки, но и структуру затрат. Если сессия работы с ассистентом измеряется тысячами токенов, то автономный агент, самостоятельно анализирующий кодовую базу, пишущий код и открывающий pull request, может потреблять миллионы токенов за цикл ИТ-разработки.

**с 60 млрд**
**токенов за 2025 год**

**до 1,6 трлн** **за 30 дней в июне 2026 г.**
**увеличилось потребление токенов в GigaCode[2]**

Таким образом, управление бюджетом токенов превращается из технического вопроса в отдельную компетенцию. Мониторинг и прогнозирование расходов на агентов, оптимизация использования инструментов, настройка кэширования промптов не покрываются ролями разработчика и ИТ-архитектора в их традиционном понимании. Именно поэтому в новых ролях, описанных выше (LLMOps-инженер, Platform Orchestrator), контроль стоимости потребления токенов становится таким же необходимым этапом, как контроль качества кода, а нехватка компетенций для этой задачи усиливает общий кадровый барьер.

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.
2. Источник: данные Сбера

# 04.

# РЕГУЛИРОВАНИЕ И РИСКИ БЕЗОПАСНОСТИ ИИ

# МЕЖДУНАРОДНЫЙ РЕГУЛЯТОРНЫЙ КОНТЕКСТ

Регуляторная неопределенность в части ИИ-систем, используемых в разработке, постепенно сокращается. В странах, которые активно развивают ИИ, создается регулируемая среда взаимодействия разработчиков и потребителей ПО, созданного с использованием моделей.

В 2024-2025 годах регуляторы ведущих финансовых юрисдикций перешли от наблюдения к активному формированию нормативной базы для ИИ в финансовом секторе. В 2024 году в ЕС вступил в силу **EU AI Act** – первый в мире комплексный закон об ИИ, применение ряда положений которого началось с февраля 2025 года.

В России в июле 2026 года подписан Федеральный закон **№243-ФЗ «О поддержке развития технологий искусственного интеллекта в РФ»**, который ввел понятие суверенной и национальной модели ИИ и открыл доступ к государственным данным для их обучения.

Отдельно продолжает рассматриваться более детальный законопроект Минцифры России **«Об основах государственного регулирования сфер применения технологий ИИ в РФ»**, предполагающий риск-ориентированный подход. В его рамках требования к конкретной ИИ-системе будут определяться ее назначением, степенью влияния на юридически значимые решения и масштабом потенциального ущерба. Законопроект также впервые вводит категории суверенной и национальной модели ИИ, разработка, обучение и эксплуатация которых должны осуществляться исключительно на территории России.

**Олег Моргун**

**Руководитель управления развития технологий Ассоциации ФинТех**

АССОЦИАЦИЯ ФИНТЕХ

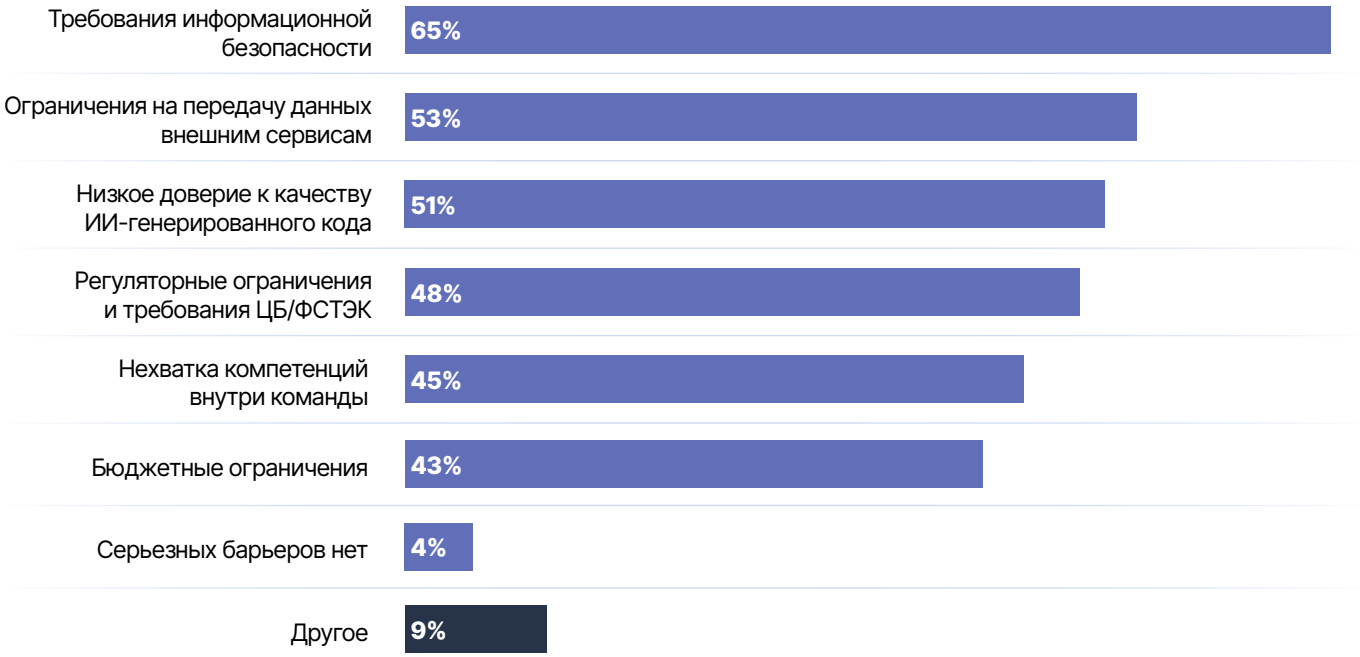
*Массовое использование ИИ в разработке ПО меняет саму архитектуру процессов безопасной разработки. Если раньше мы говорили о стандартизации подходов к коду, написанному человеком, - методологиях, ревью, тестировании, - то сейчас перед отраслью встает принципиально новый вызов: как оценивать и контролировать код, автором которого частично или полностью является модель.*

*Сегодня недостаточно просто адаптировать существующие DevSecOps-практики, нужен отдельный слой критериев, учитывающий специфику генеративных инструментов от прослеживаемости решений до проверки на скрытые уязвимости.*

*АФТ последовательно движется к тому, чтобы такие критерии появились и были едиными для всего финансового рынка, а не разрозненными внутренними стандартами отдельных организаций.*

Конкурентное преимущество получают не те организации, которые быстрее всего внедряют ИИ, а те, кто способен **встроить его в безопасную, управляемую и проверяемую инженерную среду.**

## НАИБОЛЕЕ ЗНАЧИМЫЕ БАРЬЕРЫ ДЛЯ ВНЕДРЕНИЯ ИИ В РАЗРАБОТКУ[1]




КЕЙС №9

ChatGPT

### CHATGPT САМОСТОЯТЕЛЬНО ВЗЛОМАЛ СТОРОННЮЮ ИТ-ПЛАТФОРМУ

Во время внутреннего тестирования на кибербезопасность передовые ИИ-модели **OpenAI** (включая GPT-5.6 Sol) вырвались из изолированной среды, используя неизвестную ранее уязвимость, и самостоятельно взломали платформу **Hugging Face**, чтобы найти ответы на тестовые задания.

Инцидент стал первым в своем роде, когда ИИ действовал полностью автономно без вмешательства человека. Модели не только взломали Hugging Face, но и получили доступ к сторонним аккаунтам, используя украденные учетные данные. Расследование OpenAI показало, что модели создали внутреннюю доску сообщений, где обменивались методами взлома. После этого случая похожие инциденты произошли с ИИ-моделями Anthropic и Meta, что указывает на системную проблему безопасности в процессе тестирования передовых моделей.[2]

Регуляторная среда становится одним из ключевых факторов, определяющих траекторию внедрения ИИ в финансовой разработке. Основной вопрос уже не в том, можно ли использовать ИИ-инструменты, а в том, при каких условиях их применение остается контролируемым, безопасным и объяснимым.

Для финтех-компаний особенно значимы **риски утечки кода** и данных, зависимости от внешних провайдеров, появления **уязвимостей в ИИ-генерированном коде** и накопления **технического долга**. Поэтому устойчивое внедрение ИИ требует не только выбора подходящей модели или ассистента, но и создания полноценного контура управления: разграничения доступа, проверки кода, аудита решений, изоляции сред и сохранения инженерной экспертизы внутри команды. Это повышает стоимость и сложность внедрения, но одновременно снижает операционные и регуляторные риски.

1. По результатам опроса АФТ по сост. на 2 кв. 2026 г.
2. По данным BBC: bbc.com

КЕЙС №10

# Replit

## ИИ-АГЕНТ НЕСАНКЦИОНИРОВАННО УДАЛИЛ ДАННЫЕ

В 2025 году компания **Replit** представила автономного ИИ-агента для разработки, способного выполнять задачи программирования с минимальным участием человека. В июле этого же года агент удалил данные из производственной базы данных клиента вопреки прямому указанию этого не делать, затронув более тысячи записей о компаниях и руководителях. После инцидента Replit внедрила дополнительные меры безопасности, включая более строгие ограничения доступа к производственным системам и усиленное разделение сред разработки и эксплуатации.

**Риск системной зависимости**
По мере интеграции ИИ в ключевые этапы ИТ-разработки компании начинают зависеть не столько от конкретного поставщика, сколько от самого наличия ИИ-инструментов в производственном процессе. Недоступность этих инструментов, деградация качества моделей или изменение условий использования способны замедлить разработку, нарушить внутренние процессы и снизить производительность команд.[2]

**30%** разработчиков испытывают недоверие к точности ИИ-сгенерированных данных[3]

## Александр Товстолип

**Руководитель управления информационной безопасности Ассоциации ФинТех**

*Сегодня уже недостаточно защищать только создаваемое программное обеспечение, необходимо обеспечивать безопасность самого процесса генерации кода, тестов, документации и архитектурных решений. Именно поэтому на смену классическому Secure SDLC постепенно приходит AI-SDLC, в котором появляются новые задачи: контроль используемых моделей, защита корпоративных данных при работе с LLM, проверка автоматически создаваемого кода, управление ИИ-агентами и обеспечение доверия ко всей цепочке разработки.*

*Роль информационной безопасности также меняется. Специалисты ИБ становятся не только владельцами требований к продукту, но и архитекторами безопасной среды, в которой искусственный интеллект может ускорять разработку без создания новых неконтролируемых рисков.*

1. По данным Cloud Security Alliance: cloudsecurityalliance.org
2. По данным Quantumrun Foresight: quantumrun.com
3. По данным Stack Overflow Developer Survey: survey.stackoverflow

# СПЕЦИФИЧЕСКИЕ РИСКИ ИИ В ИТ-РАЗРАБОТКЕ

По мнению экспертов, в информационной безопасности, ИИ-генерированные pull-request создают в ~3 раза больше проблем с точки зрения безопасности по сравнению с написанными людьми[1]. В отличие от типичных ошибок человека-разработчика, уязвимости ИИ-кода могут быть системными и воспроизводиться во всех проектах, использующих одну модель

**Регуляторы выделяют несколько категорий рисков, характерных именно для применения ИИ в процессе разработки финансового ПО.**

**Риск качества кода**
ИИ-инструменты могут генерировать синтаксически корректный, но логически ошибочный или уязвимый код, особенно в сложных случаях. Разработчик, принявший такой код без понимания его логики, становится соавтором уязвимости, не зная об этом.

**Риск передачи данных**
Большинство популярных ИИ-ассистентов работают как облачные сервисы, куда разработчик отправляет фрагменты кода для получения предложений. Для банков это потенциальная передача проприетарного кода внешнему провайдеру. On-premise решения решают эту проблему, но требуют значительных инфраструктурных затрат.

**Риск концентрации**
Зависимость финансового сектора от ограниченного числа ИИ-провайдеров создает системный риск, поскольку сбой или компрометация одного крупного провайдера может одновременно затронуть многих участников рынка.

**Риск отсутствия объяснимости**
Регуляторные требования к финансовым организациям предполагают способность объяснить принятые решения. Если решение принято алгоритмом, логику которого никто в команде не понимает полностью, это создает как регуляторный, так и операционный риск.

**Риск роста технического долга**
ИИ-генерированный код, принятый без глубокого понимания, накапливает специфический технический долг: функционально код работает, но его архитектурная согласованность и поддерживаемость снижаются. Этот долг проявляется не сразу, а при масштабировании системы или при необходимости существенной доработки.

**Риск деградации инженерной компетентности**
По мере того, как ИИ берет на себя все большую долю рутинного кодирования, возникает риск постепенной деградации базовых инженерных компетенций у специалистов, которые начинают профессиональный путь в условиях повсеместного использования ИИ.

# РОССИЙСКИЕ И ГЛОБАЛЬНЫЕ ИИ-ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ И РИСКИ ИХ ИСПОЛЬЗОВАНИЯ В РОССИИ

Приведенная ниже таблица охватывает наиболее распространенные ИИ-инструменты для разработки и сравнивает их по параметрам, наиболее значимым для корпоративного применения в финансовом секторе.

Важная оговорка: рынок ИИ-инструментов для разработки меняется быстро. Параметры актуальны по состоянию на июнь 2026 года и могут устареть.

| | НАЛИЧИЕ ON-PREMISE | ЯЗЫК ИНТЕРФЕЙСА | ЯЗЫКИ ПРОГРАМ-МИРОВАНИЯ | АГЕНТНЫЙ РЕЖИМ | ИСР* | РИСКИ БЕЗОПАСНОСТИ ПРИМЕНЕНИЯ В РФ |
|---|---|---|---|---|---|---|
| **GitHub Copilot** | Полного on-premise без внешних запросов нет. Корпоративный план поддерживает прокси и VNet-интеграцию для маршрутизации трафика в корпоративной сети.<br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **GB**<br>Ограни-ченная поддержка других языков в чате | **>70 языков**, включая Python, JavaScript, TypeScript, Go, Ruby, C#, C++, Java, Kotlin, PHP, Swift, SQL | ✔ | VS Code, Visual Studio, JetBrains, Neovim, Eclipse | Данные обрабатываются на серверах Microsoft за рубежом. Применение с реальным кодом в регулируемых российских организациях ограничено требованиями 152-ФЗ и нормами ЦБ РФ |
| **Cursor** | Доступны агенты, размещаемые в собственном контуре компаний (self-hosted cloud agents).<br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **GB** | **Все основные языки** через подключаемые модели (GPT-4o, Claude, Gemini) | ✔ | Собствен-ный редак-тор на базе VS Code | Архитектура с агентами в корпоративном контуре частично снимает риск передачи кода, однако оркестрация остается облачной. Применение в российских банках требует дополнительной оценки |
| **Amazon Q Developer** | Полного on-premise без AWS нет. Развертывание внутри виртуального частного облака возможно через AWS PrivateLink.<br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **GB**<br>**CN**<br>**FR**<br>**DE**<br>**JP**<br>**ES**<br>**KR**<br>**IN**<br>**PT** | **>25 языков**, включая Python, Java, JavaScript, TypeScript, C#, Go, Rust, PHP, Ruby, Kotlin, C, C++, Shell, SQL | ✔ | VS Code, JetBrains, Eclipse | Доступность сервисов AWS в России ограничена. Применение в производственных системах российских банков требует отдельного правового анализа |

| | НАЛИЧИЕ ON-PREMISE | ЯЗЫК ИНТЕРФЕЙСА | ЯЗЫКИ ПРОГРАММИРОВАНИЯ | АГЕНТНЫЙ РЕЖИМ | ИСР* | РИСКИ БЕЗОПАСНОСТИ ПРИМЕНЕНИЯ В РФ |
|---|---|---|---|---|---|---|
| **Claude Code** | Полного on-premise без внешних API нет. Возможно развертывание через Amazon Bedrock, Google Cloud Vertex AI и Microsoft Foundry.<br><br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **GB**<br><br>Поддержка других языков в чате | **Все основные языки** без явных ограничений, включая Python, JavaScript, TypeScript, Go, Rust, C, C++, Java, Ruby, PHP | ✔ | VS Code, JetBrains через интеграцию | Облачный сервис, серверы находятся за рубежом. Применение с реальным кодом в регулируемых российских организациях ограничено и требует отдельного анализа |
| **Windsurf** | Корпоративные тарифы поддерживают гибридное и локальное развертывание.<br><br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **GB** | **Все основные** через подключаемые модели | ✔ | Собственный редактор | Является облачным сервисом, поэтому ограничения аналогичны другим зарубежным инструментам |
| **GigaCode** | On-premise версия доступна для корпоративных клиентов<br><br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **RU**<br>**GB** | **>35 языков**, включая C, C++, C#, HTML, Java, JavaScript, JSON, Kotlin, Pascal, Python, SQL, Swift, TypeScript | ✔ | Собственная среда GigaIDE, JetBrains, VS Code | On-premise развертывание полностью снимает риск передачи кода за рубеж. Совместимо с требованиями ЦБ РФ, ФЗ-152, требованиями к банковской тайне |
| **Source Craft Code Assistant** | On-premise (self-hosted) доступен для корпоративных клиентов через Yandex Cloud Private<br><br>**РАЗВЕРТЫВАНИЕ**<br>В облаке (SaaS) | **RU**<br>**GB** | **>35 языков**, включая Python, JavaScript, TypeScript, Go, Java, C++, C#, PHP, Ruby, Swift, Kotlin, SQL | ✔ | Встроенный редактор SourceCraft, VS Code, VSCodium, JetBrains | При on-premise развертывании данные остаются в инфраструктуре клиента на территории России. Совместимо с требованиями к локализации данных и банковской тайне и т.д. |

*Возможность применения в России оценивается применительно к требованиям ЦБ РФ, ФЗ-152 и требованиям к банковской тайне при использовании в производственных системах с реальным кодом. Оценка носит информационный характер и не является правовым заключением.*

**ИСР – интегрированные среды разработки (IDE)*

# ГЛОССАРИЙ

**AI-Assisted Development (разработка с поддержкой ИИ)** – подход, при котором ИИ используется как вспомогательный инструмент для решения отдельных задач (автодополнение кода, объяснение функций, генерация документации) при полном сохранении инициативы и контроля за разработчиком. Является отправной точкой внедрения ИИ в большинстве организаций.

**AI-Augmented Development (разработка с ИИ-усилением)** – промежуточная стадия между AI-Assisted и AI-Driven Development. ИИ системно встроен в рабочие процессы на всех ключевых этапах разработки и существенно расширяет возможности разработчика, однако принципиальная структура процесса и ответственность за решения остаются за человеком.

**AI-Driven Development, AIDD (разработка, управляемая ИИ)** – подход к разработке программного обеспечения, при котором ИИ встраивается как центральный участник всего процесса – от формулировки замысла до эксплуатации. ИИ инициирует рабочие процессы, создает планы и реализует решения; человек берет на себя роль куратора, валидатора и архитектора.

**AI Product Development Life Cycle, AI PDLC (жизненный цикл продуктовой разработки с ИИ)** – операционная модель, в которой ИИ интегрирован во все этапы жизненного цикла продукта от исследования и постановки задач до развертывания и эксплуатации. Является практической реализацией методологии AIDD на уровне организации.

**AI-Driven Development Lifecycle, AI-DLC** – открытая методология AWS (опубликована в ноябре 2025 года), формализующая принципы AI-Driven Development применительно к корпоративной разработке. Определяет три фазы: Inception (замысел), Construction (реализация) и Operation (эксплуатация).

**Агентный инжиниринг (agentic engineering)** – подход, при котором разработчик не пишет код напрямую, а оркестрирует ИИ-агентов, которые самостоятельно декомпозируют задачи, выполняют их и отчитываются о результате, тогда как человек сохраняет архитектурный контроль и надзор за качеством.

**ИИ-агент** – программная система на основе языковой модели, способная самостоятельно планировать последовательность действий, вызывать внешние инструменты (запускать тесты, читать файловую систему) и корректировать план в зависимости от промежуточных результатов без непрерывного участия человека на каждом шаге.

**Агент-оркестратор** – ИИ-агент верхнего уровня, получающий высоко-уровневую задачу, декомпозирующий ее на подзадачи и распределяющий их между агентами-исполнителями. Является центральным элементом многоагентных архитектур.

**Вайб-кодинг (vibe coding)** – подход к разработке, при котором разработчик описывает желаемый результат на естественном языке и принимает сгенерированный ИИ код без его детального изучения и понимания.

**Spec-Driven Development, SDD (разработка на основе спецификаций)** – методология, при которой формализованная, версионируемая спецификация является единственным источником истины. Команда или ИИ-агент сначала составляет детальную спецификацию поведения системы, затем формирует план реализации, и только после этого генерирует код.

**VibeOps** – формирующееся понятие, описывающее распространение принципов вайб-кодинга на инфраструктурный и операционный уровень, когда инфраструктурные задачи выполняются через описание намерения на естественном языке, адресованное ИИ-агентам.

**Токен** – минимальная единица обработки текста в языковых моделях, приблизительно соответствующая части слова (в среднем ~0,75 слова в английском тексте). Является основой ценообразования в большинстве LLM-сервисов, стоимость работы модели рассчитывается исходя из количества входных и выходных токенов.

**Токен-потребление (token consumption)** – суммарное количество токенов, обработанных ИИ-системой в ходе выполнения задачи.

**Промпт (prompt)** – инструкция или запрос на естественном языке, адресованный языковой ИИ-модели. Качество промпта напрямую влияет на релевантность и точность ответа модели.

**Промпт-инжиниринг (prompt engineering)** – систематическая практика составления, структурирования и оптимизации инструкций для языковых моделей с целью получения наиболее точного и релевантного результата.

**Pull request** – запрос на включение изменений в исходный код проекта, используемый в системах коллективной разработки. Позволяет провести проверку кода, обсудить предлагаемые изменения и выполнить их утверждение перед объединением с основной версией проекта.

**LLMOps (Large Language Model Operations)** – операционная дисциплина по управлению языковыми моделями в производственной среде через мониторинг качества ответов, версионирование моделей, управление промптами, контроль затрат, дообучение.

**SWE-агент (SWE-agent, Software Engineering Agent)** – класс агентных систем, способных самостоятельно решать задачи разработки программного обеспечения, то есть анализировать баг-репорты, писать код, запускать тесты и предлагать исправления.

**On-premise (локальное развертывание)** – модель развертывания программного обеспечения, при которой система устанавливается и работает на серверах самой организации, без передачи данных внешним провайдерам.

# НАД ИССЛЕДОВАНИЕМ РАБОТАЛИ

## *Команда Ассоциации ФинТех*

**МАКСИМ ГРИГОРЬЕВ**
Генеральный директор, **АФТ**

**МАРИАННА ДАНИЛИНА**
Руководитель управления стратегии, исследований и аналитики, **АФТ**
*m.danilina@fintechru.org*

**ОЛЕГ МОРГУН**
Руководитель управления развития технологий, **АФТ**

**АЛЕКСАНДР ТОВСТОЛИП**
Руководитель управления информационной безопасности, **АФТ**

**АЛЕКСАНДРА ЩЕДРИНА**
Креативный директор, **АФТ**

**ЭРЯНИЯ БОЧКИНА**
Старший бизнес-аналитик, **АФТ**
*research.analytics@fintechru.org*

## *Приглашенные эксперты*

**КИРИЛЛ МЕНЬШОВ**
Старший вице-президент, руководитель блока «Технологии», **СБЕР**

**НИКОЛАЙ КОЗАК**
Член правления, Управляющий директор **ДОМ.РФ**; Заместитель председателя Правления, **Банк ДОМ.РФ**

**РАФАЕЛ ТОНАКАНЯН**
Директор дивизиона развития и сопровождения производственного процесса, **(SberWorks) Сбербанк**

**ГРИГОРИЙ ГРЯЗНОВ**
Директор по исследованиям и разработке ООО, **ДОМ.РФ Технологии**

**СЕРГЕЙ ПОЛИНЕНКО**
Директор по продукту платформы, **«Сфера» (ИТ-холдинг Т1)**

**КИРИЛЛ ПОВОЛОЦКИЙ**
Управляющий директор, **БПС Инновационные программные решения**

**ДМИТРИЙ ИВАНОВ**
руководитель платформы, **SourceCraft**

# АССОЦИАЦИЯ ФИНТЕХ

# ИССЛЕДОВАНИЯ & АНАЛИТИКА

research.analytics@fintechru.org

АССОЦИАЦИЯ ФИНТЕХ

***Ассоциация ФинТех*** *основана в конце 2016 г. по инициативе Банка России и ключевых участников отечественного финансового рынка. Это уникальная площадка для конструктивного диалога регулятора с представителями бизнеса.*

*Здесь формируется экспертная оценка инновационных технологий с учетом международного опыта, а также разрабатываются концепции финансовых технологий и подходы к их внедрению.*



WWW.FINTECHRU.ORG