

# КАК СОЗДАВАТЬ ПРОДУКТЫ СИИ

От первых экспериментов  
до работающей системы



НАТАН РОУ

Натан Роу

**Как создавать продукты с ИИ.  
От первых экспериментов  
до работающей системы**

«Автор»

2026

**Рой Н.**

Как создавать продукты с ИИ. От первых экспериментов до  
работающей системы / Н. Рой — «Автор», 2026

Хватит делать демоверсии, за которые никто не платит. Ваш ИИ красиво отвечает на презентации. А выдержит ли он реального клиента, плохие данные и собственную ошибку? Именно здесь заканчивается восторг и начинается работа над продуктом. В этой книге разобран весь путь: выбрать задачу, проверить спрос и возможности модели, собрать систему, пройти проверку перед запуском и наладить обратную связь. Вы поймёте, когда нужен агент, зачем продукту память и база знаний, какие эксперименты пора закрывать и по каким признакам можно двигаться дальше. Для основателей, продакт-менеджеров и разработчиков, уставших начинать заново после каждого удачного прототипа. Откройте книгу и доведите наконец одну идею до продукта, который можно предложить клиенту.

© Рой Н., 2026

© Автор, 2026

# Содержание

|                                                                                                |    |
|------------------------------------------------------------------------------------------------|----|
| Предисловие: Революция в области программных продуктов в эпоху ИИ                              | 6  |
| Глава 1. Возникновение интеллектуальных систем                                                 | 10 |
| 1. От системы правил к системе процессов                                                       | 11 |
| 2. Сравнение типичных примеров трех типов систем                                               | 12 |
| 3. Пределы системы рабочих процессов                                                           | 13 |
| 4. Появление систем способностей                                                               | 14 |
| 5. Переход от функционального мышления к мышлению, ориентированному на способности             | 15 |
| 6. Взрывной рост применения ИИ-агентов                                                         | 16 |
| 7. Новые проблемы, возникающие в эпоху систем способностей                                     | 17 |
| 8. Почему эти проблемы приводят к появлению новой методологии                                  | 18 |
| Глава 2. Новые проблемы, возникающие в связи с парадигмой «AI Native»                          | 19 |
| 1. Программное обеспечение превращается из инструмента в помощника                             | 20 |
| 2. Переход программного обеспечения от функциональных систем к системам на основе способностей | 21 |
| 3. Сравнение трёх типичных категорий продуктов                                                 | 22 |
| 4. Поведение программного обеспечения начинает зависеть от контекста                           | 23 |
| 5. Программное обеспечение начинает участвовать в выполнении задач                             | 24 |
| 6. Новые проблемы, возникающие в связи с парадигмой AI Native                                  | 25 |
| 7. Почему для этого требуется новая методология разработки продуктов                           | 26 |
| Глава 3. Необходимость модернизации методов программной инженерии                              | 27 |
| 1. Методология всегда меняется в соответствии с эволюцией систем                               | 28 |
| 2. Откуда берется неопределенность в продуктах на базе ИИ                                      | 29 |
| 3. От инженерного замкнутого цикла к продуктовому замкнутому циклу                             | 30 |
| 4. Теоретические истоки данной методологии                                                     | 31 |
| 5. Продукт-менеджеры и инженеры должны повысить свою квалификацию                              | 32 |
| Глава 4. Методология AI Native Product                                                         | 33 |
| 1.1. Почему генеративный ИИ делает эту методологию жизнеспособной                              | 35 |
| 1.2. Для каких типов продуктов эта методология подходит больше всего                           | 37 |
| 1.3. Какие типы продуктов не подходят для прямого применения                                   | 39 |
| 1.4. В каких сценариях нецелесообразно внедрять данную методологию                             | 40 |

|                                                                                                                        |    |
|------------------------------------------------------------------------------------------------------------------------|----|
| 1.5. Метод быстрого принятия решений                                                                                   | 41 |
| 1.6. Непосредственные преимущества данной методологии                                                                  | 42 |
| Краткий обзор входов и выходов пяти этапов                                                                             | 44 |
| 1.7. Объяснение ключевых концепций и новых терминов данной главы                                                       | 45 |
| 2.1. Что на самом деле означает каждый узел на блок-схеме                                                              | 47 |
| 2.3. Помимо основной вертикальной линии этапов, существует также горизонтальная линия взаимодействия человека и машины | 48 |
| 3. Что на самом деле является результатом каждого этапа                                                                | 48 |
| 3.1. Результаты определения направления                                                                                | 49 |
| 3.2. Результаты проведения эксперимента                                                                                | 50 |
| 3.3. Результаты построения системы                                                                                     | 51 |
| 3.4. Результаты аудита и одобрения                                                                                     | 52 |
| 3.5. Результаты эксплуатации в производственной среде                                                                  | 53 |
| 3.6. Понимание этой основной линии в одном предложении                                                                 | 54 |
| 4. Переход между этапами осуществляется через контрольные точки, а не на основе интуиции                               | 54 |
| 4.1. «Шлюз» «Определение направления — развертывание эксперимента»                                                     | 55 |
| 4.2. Переход от экспериментальной фазы к построению системы                                                            | 56 |
| 4.3. Этап «Создание системы» → «Одобрение аудита»                                                                      | 57 |
| 4.4. Шлюз «Одобрение аудита» → «Производственная эксплуатация»                                                         | 58 |
| 5. На самом деле в процессе происходит не линейное продвижение, а сближение целей                                      | 58 |
| 6. «Теневая система» — важный мост от эксперимента к производству                                                      | 59 |
| 7. Три цикла вместе составляют полную методологию                                                                      | 59 |
| 8. Два уровня поддержки: уровень управления (Governance Layer) и уровень возможностей (Capability Layer)               | 60 |
| 8.2. Уровень возможностей (Capability Layer)                                                                           | 61 |
| 9. Как читать всю книгу                                                                                                | 61 |
| 10. Конечная цель данной методологии                                                                                   | 62 |
| Конец ознакомительного фрагмента.                                                                                      | 63 |

**Натан Роу**

# **Как создавать продукты с ИИ. От первых экспериментов до работающей системы**

## **Предисловие: Революция в области программных продуктов в эпоху ИИ**

В данной книге речь идет не о какой-то конкретной модели и не о какой-то популярной архитектуре, а о более фундаментальном вопросе: как следует определять, проектировать и создавать продукты и управлять ими, когда программное обеспечение начинает обладать способностями к пониманию, логическому выводу, вызову инструментов и непрерывному обучению. Если в тексте не указано иное, термин «ИИ» в данной книге в основном относится к генеративному искусственному интеллекту, основанному на крупных языковых моделях и связанных с ними мультимодальных моделях, а не является общим обозначением всей традиционной области искусственного интеллекта.

В прошлом индустрия программного обеспечения уже пережила несколько значительных смен парадигм. В эпоху настольного программного обеспечения акцент делался на реализации функциональности, в эпоху Интернета — на подключении и масштабируемости, а в эпоху облачных вычислений — на непрерывной доставке и сервисном подходе. Сегодня крупные языковые модели, интеллектуальные сущности, RAG, контекстная инженерия и системы памяти начинают становиться ядром продуктов: программное обеспечение перестает быть просто «инструментом для выполнения заранее заданной логики» и постепенно превращается в «систему способностей, способную участвовать в выполнении задач».

Для многих читателей эти изменения не новы. Возможно, вы уже пользовались ИИ-агентами для программирования, которые напрямую генерируют работоспособный демо-вариант; или же ИИ-инструментами для прототипирования, позволяющими быстро создавать эскизы интерфейсов и концептуальные схемы продукта. Более того, некоторые платформы генеративной разработки уже объединили «создание приложения» и «его публикацию», благодаря чему пользователи могут напрямую размещать проекты в сети и впоследствии продолжать их обновлять. Работа, для выполнения которой раньше требовалось участие нескольких специалистов и прохождение множества этапов, сейчас стремительно упрощается. Инструменты совершенствуются, а путь продукта от идеи до прототипа и от прототипа до валидации перестраивается, что приводит к значительному повышению эффективности.

Однако вместе с этим возникают и новые вопросы: эти возможности кажутся очень мощными, почему же, как только дело доходит до реальной разработки продукта, они вновь становятся разрозненными? Когда следует сначала создать демо-версию, а когда вернуться к определению проблемы? Как сделать так, чтобы созданные прототипы, код, интерфейсы и документация действительно стали частью основной линейки продукта, а не оставались лишь красивыми, но изолированными экспериментами? Или, например, почему команде, хотя она и «использует ИИ в своей работе», по-прежнему так сложно организовать эти возможности в стабильный процесс разработки, принятия решений о продукте и механизм запуска?

Эти сомнения — не просто абстрактные вопросы методологии; в реальности они уже обходятся дорого. Некоторые отраслевые примеры показывают, что когда ИИ участвует в изменениях производственной среды, а механизмы управления ещё не доработаны, небольшие ошибки могут быстро перерасти в инциденты системного уровня. В то же время многочисленные исследования показывают, что код, сгенерированный ИИ, даже если он функ-

ционально правильный, всё равно может содержать уязвимости безопасности или скрытые дефекты, поэтому его внедрение в производственную среду без тщательной проверки увеличивает системные риски. Эти примеры показывают, что при разработке продуктов с использованием ИИ ключевой проблемой является не сам ИИ, а то, **насколько механизмы контроля доступа, проверки кода и управления производственной средой соответствуют возможностям автоматизации ИИ.**

Кроме того, следует помнить, что подобные риски существуют не только в «созданном вами продукте на базе ИИ», но и в «ИИ-инструментах, которые вы используете для производства продуктов». Когда команда передает репозитории кода, проектные документы, внутреннюю документацию, журналы, базу знаний и возможности выполнения команд ИИ-инструментам, объект, подлежащий аудиту, фактически становится двухуровневым: один уровень — это ИИ-продукт, который будет предоставлен пользователям, а другой — цепочка ИИ-инструментов, участвующих в процессе разработки.

Это приводит к прямому следствию: традиционные методы разработки программного обеспечения больше не подходят. Раньше мы привыкли сначала определять функционал, затем составлять техническое задание (PRD), после чего планировать разработку, проводить тестирование и запускать продукт в эксплуатацию; однако в случае с ИИ-продуктами многие ключевые возможности невозможно точно описать до проведения экспериментов. Зачастую у команды есть лишь область задачи, набор данных и ряд неопределенных гипотез, и ей необходимо постепенно, путем экспериментов, выявлять возможности, которые действительно можно воплотить в продукте.

Поэтому отправной точкой разработки продуктов ИИ больше не является «определённость функций», а «проверяемость возможностей». Именно поэтому в этой книге предлагается методология «AI Native». Она пытается ответить на три взаимосвязанных вопроса:

- Как найти перспективное направление для разработки продукта на базе ИИ.
- Как превратить нестабильные возможности ИИ в стабильные возможности продукта.
- Как сформировать в реальных условиях использования продукта цикл обратной связи, цикл создания ценности и «ров данных».

Здесь также необходимо сначала уточнить вопрос о сфере применения. Полная методология, изложенная в данной книге, в основном предназначена для продуктов типа «AI Native», поскольку именно такие продукты чаще всего сталкиваются одновременно с неопределённостью в отношении возможностей, формы продукта и границ управления. Однако если команда разрабатывает не «AI Native»-продукт в строгом смысле этого слова, а использует инструменты программирования, проектирования, исследования и цепочки инструментов ИИ для разработки обычного программного обеспечения, то многие принципы, изложенные в этой книге, остаются актуальными, просто, как правило, нет необходимости применять все главы в полном объеме. Другими словами, у этой методологии есть как «полная версия», так и «упрощённая версия», из которой можно выбрать необходимые элементы. Если выразиться более просто, то это, по сути, означает «ИИ продолжает использовать ИИ»: инструменты ИИ уже начали участвовать в процессе создания программного обеспечения, а команды, в свою очередь, должны научиться определять, тестировать, создавать, проверять и управлять этими процессами разработки с участием ИИ.

Основная линия, которой следует эта книга:

Определение направления -> Проведение экспериментов -> Создание системы -> Аудит и допуск -> Ввод в эксплуатацию

При этом этап «Определение направления» (Direction) отвечает за формулировку задачи и определение направления, этап «Проведение экспериментов» (Experiment) — за проверку возможностей, этап «Создание системы» (Building) — за техническую реализацию, этап «Аудит и одобрение» (Audit) — за управление и установление границ, а этап «Производственная эксплуатация» (Production) — за обратную связь и развитие после запуска. Это не линейный процесс, а цикл разработки продукта, состоящий из повторяющихся итераций.

Эта методология, на первый взгляд, содержит больше явных этапов, чем традиционные процессы, но в случае с продуктами на базе ИИ она, как правило, оказывается более эффективной. Это связано с тем, что команда раньше выявляет ошибочные направления, раньше обнаруживает проблемы с данными и ограничениями, раньше определяет, какие возможности стоит развивать дальше, что позволяет сократить масштабную доработку в неправильном направлении. Другими словами, она обеспечивает не только большую определённость, но и более высокую эффективность НИОКР, более низкие затраты на пробные ошибки и более стабильное качество после запуска.

В рамках этой основной линии в книге далее рассматриваются три категории вопросов: во-первых, как ИИ меняет программные системы; во-вторых, как следует проектировать, разрабатывать и внедрять «AI Native»-продукты; в-третьих, как такие продукты формируют коммерческую ценность, цикл взаимодействия с клиентами и долгосрочное конкурентное преимущество.

Если вы — менеджер по продукту, эта книга поможет вам переориентироваться с «определения функций» на «определение проблем» и «проектирование возможностей»; если вы — инженер, эта книга поможет вам понять, почему при разработке архитектуры продуктов с ИИ нельзя ограничиваться только интерфейсами и страницами; если вы — предприниматель или руководитель бизнес-направления, эта книга поможет вам оценить взаимосвязь между данными, возможностями и бизнес-моделями.

Кому предназначена эта книга

Для более плавного чтения этой книги читателю желательно обладать несколькими видами базовых знаний. Во-первых, желательно иметь базовый опыт работы с программными продуктами или в области программной инженерии, по крайней мере понимать такие распространённые понятия, как требования, процессы, системы, запуск и обратная связь. Во-вторых, желательно иметь самое базовое представление о современных системах ИИ, например, понимать, о чем в общих чертах говорят такие термины, как генеративный ИИ, модели, подсказки, RAG, интеллектуальные сущности, контекст и системы памяти. В-третьих, желательно иметь реальное представление о своей отрасли или бизнес-сценарии, поскольку в книге неоднократно подчеркивается: продукты ИИ — это не абстрактная методология, оторванная от бизнеса; они всегда связаны с условиями данных, структурой процессов, границами рисков и оценкой ценности.

Однако эта книга не требует от читателя предварительного статуса эксперта по алгоритмам и не предполагает предварительного освоения сложной теории машинного обучения. Вам не нужно сначала разбираться в деталях обучения, настройке параметров или системах академической оценки, чтобы начать понимать эту методологию. Более важная подготовка заключается в другом: готовности перейти от «списка функций» к «определению проблемы», а также готовности принять, что эксперименты, сходимость, управление и обратная связь являются нормой в разработке продуктов ИИ, а не исключением.

Если вы уже имели опыт работы с какими-либо сложными бизнес-системами — будь то продукты, программное обеспечение, данные, операционная деятельность или техническое обслуживание — то эта книга будет вам легче понятна. Ведь в центре обсуждения находятся не абстрактные технологии сами по себе, а то, как команде следует перестроить подход к про-

дукту, инженерной работе, организации и принятию бизнес-решений после того, как система начнет обладать необходимыми способностями.

Основную идею книги можно сформулировать так: при создании продуктов AI Native не следует сначала полностью разработать продукт, а затем приступить к его разработке; вместо этого необходимо обнаруживать возможности в ходе экспериментов, стабилизировать их в процессе разработки и формировать сложный эффект в плане ценности, данных и организации в ходе реального использования.

## Глава 1. Возникновение интеллектуальных систем

Вопросы, на которые будет дан ответ в этой главе:

- Почему программные системы эволюционируют от систем правил к системам способностей?
  - В чём заключается принципиальное отличие продуктов «AI Native» от традиционного программного обеспечения?
  - Почему такая эволюция вынуждает появление новой методологии разработки продуктов?
  - Объяснение ключевых концепций и новых терминов
  - Rule System: программная система, основанная преимущественно на чётких правилах и фиксированных соотношениях между входными и выходными данными.
  - Workflow System: акцент делается на оркестрации процессов с участием различных ролей, этапов и систем.
  - Система возможностей (Capability System): акцент делается не на фиксированных процессах, а на наделении системы способностями к пониманию, принятию решений, генерации и планированию.
- Понимание содержания этой главы на примере
- Возьмём в качестве примера сферу обслуживания клиентов. На ранних этапах программное обеспечение больше напоминало «систему правил» (Rule System): после получения заявки она распределялась в соответствии с фиксированными правилами. Позже оно развилось в «систему рабочих процессов» (Workflow System): в ней процесс обработки заявок, их утверждение, эскалация и поиск знаний были объединены в единый поток. В эпоху искусственного интеллекта система стала «системой возможностей» (Capability System): она не только обрабатывает заявки, но и понимает суть проблемы, объясняет логику, генерирует варианты ответов и предупреждает о рисках.
- Этот пример показывает, что эволюция системы — это не просто «увеличение количества функций», а переход системы к выполнению задач более высокого уровня.
- Описание процесса эволюции
- Выполнение по фиксированным правилам -> Организация рабочих процессов -> Использование возможностей для выполнения задач
- С каждым последующим шагом неопределённость, с которой сталкивается программное обеспечение, возрастает, и поэтому всё больше возникает потребность в новых продуктах и инженерных методологиях.

## **1. От системы правил к системе процессов**

Если рассматривать развитие программного обеспечения в более широком историческом контексте, его можно условно разделить на три этапа:

Система правил -> Система рабочих процессов -> Система возможностей

Первый этап — система правил. Поведение программного обеспечения в основном определяется правилами, написанными вручную программистами, а взаимосвязь между входом, обработкой и выходом является относительно фиксированной. Второй этап — система рабочих процессов. Программное обеспечение начинает выполнять задачи по организации процессов, охватывающих различные роли, подразделения и системы, например, CRM, ERP, потоки утверждений, потоки рабочих заказов и системы BI. Основная ценность программного обеспечения на этом этапе заключается в цифровизации, стандартизации и масштабировании организационных процессов.

## 2. Сравнение типичных примеров трех типов систем

Одних абстрактных концепций недостаточно — если рассмотреть конкретные сценарии, различия между этими тремя типами систем станут гораздо более очевидными.

### Сценарий

#### Система правил (Rule System)

#### Система рабочих процессов (Workflow System)

#### Система возможностей (Capability System)

#### Служба поддержки

Маршрутизация по ключевым словам, фиксированные FAQ, шаблоны стандартных ответов

Обработка заявок, эскалация, SLA, процессы утверждения и поиска знаний

Copilot / Агент службы поддержки: понимание проблемы, поиск заказов, разъяснение логистики, генерация вариантов ответов, предупреждение о рисках

#### Анализ данных

Стандартные отчеты, расчет стандартных показателей, стандартный экспорт

Процесс запросов BI, подача заявок на отчеты, утверждение аналитики, передача прав доступа к данным

AI-помощник по анализу: понимание вопросов на естественном языке, объяснение аномалий, организация аналитических путей, формирование выводов

#### Техническое обслуживание

Пороговые оповещения, сопоставление по правилам, фиксированные скрипты

Обработка событий, эскалация дежурным, выполнение инструкций по устранению неполадок

Помощник AIOps / агент: агрегирование предупреждений, обобщение журналов, вывод первопричин, предложение рекомендаций по устранению

Эта сравнительная таблица иллюстрирует один очень важный момент: эволюция программного обеспечения заключается не в «увеличении количества функций», а в том, что система берет на себя выполнение задач все более высокого уровня. Система правил (Rule System) решает задачи определения действий, система рабочих процессов (Workflow System) — определения процессов, а система возможностей (Capability System) начинает обрабатывать полуструктурированные суждения, организовывать сложную информацию и выполнять многоэтапные задачи.

### **3. Пределы системы рабочих процессов**

Система рабочих процессов очень мощна, но её эффективность зависит от того, насколько чётко определён сам процесс. При условии чёткого определения правил и границ процесса система может стабильно функционировать. Проблема заключается в том, что многие высокоценные задачи не сводятся лишь к выполнению процессов, а включают в себя понимание, суждение, обобщение, генерацию и совместную работу. Например, анализ сложного лога, обобщение разговора с клиентом, интерпретация набора операционных показателей, извлечение ответов из фрагментарных данных — все эти задачи трудно полностью описать в виде фиксированных процессов.

В этом и заключается типичное ограничение традиционного программного обеспечения: оно хорошо справляется с «быстрым выполнением определённых задач», но не способно «превратить неопределённые задачи в навыки».

## **4. Появление систем способностей**

Появление ИИ, в особенности крупных языковых моделей, перевело программное обеспечение на третий этап — этап систем способностей. Суть систем способностей заключается не в жестком закреплении каждого шага, а в том, чтобы система обладала набором повторно используемых способностей для выполнения задач, таких как понимание заявок, интерпретация данных, планирование задач, вызов инструментов и формулирование выводов.

На этом этапе начинает меняться базовая структура программного обеспечения. Пользователь больше не просто нажимает кнопки и проходит по рабочему процессу, а ставит цели; система также больше не просто выполняет команды, а должна понимать контекст, выбирать пути, запускать инструменты и возвращать результаты пользователю.

## **5. Переход от функционального мышления к мышлению, ориентированному на способности**

Системы правил в основном ориентированы на функции, системы процессов — на бизнес-потоки, а системы способностей — на «модули способностей, поддающиеся постоянному усовершенствованию». Это означает, что необходимо изменить и подход к определению продукта. Раньше менеджер по продукту мог напрямую определять страницы, поля и функциональные процессы; теперь же гораздо важнее определять область задач, условия данных, целевые задачи и критерии оценки.

Как только программное обеспечение переходит на этап системы способностей, при разработке продукта нельзя просто спрашивать: «Стоит ли реализовывать эту функцию?», а необходимо задавать вопрос: «Можно ли проверить, стабилизировать и масштабировать данную способность?».

## 6. Взрывной рост применения ИИ-агентов

В 2026 году количество приложений на базе агентов заметно увеличилось, и это произошло не потому, что концепция «агента» внезапно стала популярной, а потому, что за последний год одновременно созрели ключевые условия, необходимые для внедрения агентов.

Во-первых, возможности моделей по многошаговому выводу и вызову инструментов начали внедряться в продукты. В **марте 2025 года** OpenAI запустила Responses API, встроенные инструменты и Agents SDK, интегрировав вызов инструментов, оркестрацию задач и наблюдаемость в официальный набор возможностей, что сделало создание работоспособных агентов возможностью на уровне платформы.

Во-вторых, разработка агентов стала опираться на фреймворки, более близкие к программной инженерии. ADK от Google описывает агента как программную систему и предоставляет структуры рабочих процессов (Sequential, Parallel, Loop), архитектуру с несколькими агентами, а также возможности развертывания и оценки. Это означает, что агент больше не является просто демонстрационным примером, а начинает обладать повторно используемым инженерным каркасом.

В-третьих, подключение к внешним системам начинает стандартизироваться с помощью протокола. MCP абстрагирует данные, инструменты и рабочие процессы приложений ИИ в виде открытого протокола, а OpenAI в обновлении 2025 года также поддерживает удалённые серверы MCP, благодаря чему подключение агентов к корпоративным системам больше не зависит исключительно от разовой кастомизации.

В-четвертых, начали появляться среды выполнения производственного уровня. Foundry Agent Service от Microsoft объединяет среду выполнения агента, триггеры событий, мониторинг и управление в единую платформу, позволяя предприятиям запускать агентов на имеющейся инфраструктуре.

Наконец, экосистема открытого исходного кода начала быстро усиливать эту тенденцию. Например, платформа агентов с открытым исходным кодом **OpenClaw** быстро стала популярной в начале 2026 года: на GitHub она набрала более 200 000 звёздочек и поддерживает множество моделей, каналов и большое количество повторно используемых навыков, что позволяет разработчикам быстро создавать автоматизированные ИИ-агенты.

Таким образом, истинная причина роста популярности агентов в 2026 году заключается не только в повышении мощности моделей, но и в **одновременном созревании возможностей моделей, инструментальных протоколов, фреймворков разработки, платформ для запуска и экосистемы открытого исходного кода.**

## **7. Новые проблемы, возникающие в эпоху систем способностей**

С переходом программного обеспечения к системе способностей проблемы, с которыми сталкиваются команды, также изменились. Именно эти проблемы традиционные продуктовые подходы с трудом могут полностью охватить.

•

Во-первых, требования больше нельзя заранее зафиксировать. Зачастую у команды сначала есть только область задачи, но нет четкого списка функций.

•

Во-вторых, границы системы больше не определяются исключительно кодом, а определяются совместно данными, контекстом, памятью, инструментами и правами доступа.

•

В-третьих, обоснованность способностей необходимо выявлять экспериментально, а не выводить напрямую из проектной документации.

•

В-четвертых, риски больше не сводятся лишь к «логическим ошибкам», но включают в себя иллюзии, превышение полномочий, ошибочные действия, отсутствие отслеживаемости и неконтролируемые высокие затраты.

•

В-пятых, запуск больше не означает завершение сдачи продукта: обратная связь из реального мира, неудачи и ручные исправления, в свою очередь, переопределяют направление развития продукта.

Другими словами, система способностей — это не просто увеличение масштабов традиционного программного обеспечения, а переход продукта, инженерии, управления и производственных процессов на уровень более высокой неопределённости.

## 8. Почему эти проблемы приводят к появлению новой методологии

Именно из-за появления этих новых проблем команды не могут полагаться исключительно на традиционный подход «требования → проектирование → разработка → тестирование → запуск» при создании продуктов искусственного интеллекта.

В эпоху систем правил самое важное — правильно сформулировать правила; в эпоху систем процессов — правильно выстроить последовательность процессов; а в эпоху систем способностей команда должна одновременно ответить на следующие вопросы:

- 

Действительно ли поставленная нами задача заслуживает решения с помощью ИИ?

- 

Прошла ли эта способность достаточную экспериментальную проверку и действительно ли она работает?

- 

Как эту способность превратить в полноценную систему, а не оставить на стадии демо-версии?

- 

Какие границы

эта способность будет иметь при внедрении в реальную среду, кто будет давать разрешение на запуск и как откатиться назад в случае проблем.

- 

Как после запуска реальные отзывы будут учитываться при определении направления и планировании следующего раунда экспериментов.

Именно в этом заключается отправная точка методологии, представленной в данной книге. Иными словами, методология «AI Native Product» начинается не с формулировки концепции, а с реальных проблем, возникающих в связи с системой компетенций. Она стремится предложить основную линию развития, более подходящую для продуктов, основанных на компетенциях:

Определение направления -> Проведение экспериментов -> Построение системы -> Аудит и запуск -> Ввод в эксплуатацию

Эта основная линия не создает дополнительной нагрузки на процессы, а позволяет команде в эпоху систем с высокой степенью неопределённости по-прежнему упорядоченно выявлять, стабилизировать, ограничивать и управлять способностями.

Заключение к этой главе

Три этапа эволюции программных систем — это не просто изменение технологического стека, а изменение способа организации продукта. Система правил делает акцент на определённости, система процессов — на масштабируемости, а система способностей — на непрерывном формировании способностей с помощью данных, моделей, контекста, инструментов и инженерных методов. К 2026 году заметное увеличение числа приложений на базе агентов будет обусловлено не прорывом в одной отдельной модели, а совместным созреванием многошагового вывода, вызова инструментов, MCP, среды выполнения, мониторинга и управления. Именно потому, что система способностей привносит новую неопределённость, новые проблемы границ и новую ответственность за производство, возникла необходимость в методологии, представленной в этой книге.

## Глава 2. Новые проблемы, возникающие в связи с парадигмой «AI Native»

Вопросы, на которые будет дан ответ в этой главе:

- 

В чём именно заключается «новизна» программного обеспечения AI Native по сравнению с традиционным программным обеспечением?

- 

Почему продукты ИИ больше похожи на системы способностей, а не на функциональные системы?

- 

Какие новые проблемы в области продуктов, разработки и управления вызывает эта смена парадигмы?

- 

Почему эти проблемы будут стимулировать появление новой методологии разработки продуктов?

Объяснение ключевых концепций и новых терминов

- 

Функциональная система: программное обеспечение, в основе которого лежат страницы, модули и фиксированные операции.

- 

Система способностей: программное обеспечение, в основе которого лежат такие способности, как понимание, анализ, генерация, планирование и выполнение.

- 

Контекстная зависимость: поведение системы определяется не только кодом, но и зависит от данных, состояния, ролей и истории.

- 

Agentic Software: форма программного обеспечения, способная организовывать задачи вокруг цели, задействовать инструменты и продвигать работу.

Понимание данной главы на примере

Традиционные системы бизнес-аналитики (BI) требуют, чтобы пользователь самостоятельно выбирал поля, временные рамки и условия фильтрации, а затем изучал диаграммы; аналитическая система AI Native же позволяет пользователю просто сказать: «Объясни мне, почему за последний месяц снизился коэффициент повторных покупок». В первом случае пользователь должен адаптироваться к системе, во втором — система должна понять цель и организовать процесс анализа.

Этот пример показывает, что самое существенное изменение в парадигме AI Native заключается не в том, что «интерфейс заменили на окно чата», а в том, что программное обеспечение начинает переходить от реагирования на действия к реагированию на цели.

Описание процесса изменения парадигмы

Функции, управляемые пользователем -> Пользователь формулирует цель -> Система организует выполнение части работы

Как только программное обеспечение переходит от реагирования на функции к реагированию на цели, изменяются и дизайн продукта, и архитектура, и подходы к управлению.

## **1. Программное обеспечение превращается из инструмента в помощника**

Традиционное программное обеспечение больше похоже на инструмент. Его ценность заключается в том, чтобы выполнять определённое действие быстрее, точнее и более стандартизированно. Пользователь осваивает программное обеспечение, изучая страницы, кнопки и процессы. Программное обеспечение AI Native, напротив, больше похоже на помощника. Пользователь даёт не просто операционные команды, а задачи, намерения и контекст.

Например, традиционная система отчетности требует, чтобы пользователь самостоятельно выбирал параметры, период времени и нажимал кнопки фильтрации; аналитическая система AI Native же позволяет пользователю просто сказать: «Помоги мне объяснить причины снижения коэффициента повторных покупок за последний месяц», после чего система самостоятельно организует процесс анализа.

Это означает, что отношения между программным обеспечением и пользователем изменились. Раньше «пользователь учился работать с системой», а сейчас всё чаще «система понимает пользователя».

## **2. Переход программного обеспечения от функциональных систем к системам на основе способностей**

Основным активом традиционных продуктов обычно является набор функций, например управление заказами, управление запасами, запрос отчетов. Основным активом продуктов AI Native же больше похож на набор способностей, например способность анализировать журналы, отвечать на вопросы, разбивать задачи на части, реагировать на запросы клиентов. Способности являются более абстрактными, чем функции, и их легче повторно использовать в различных сценариях.

Именно поэтому в продуктах искусственного интеллекта часто встречаются такие структуры, как «Agent», «Skill», «Memory» и «Tool». По сути, ни одна из них не является «модулем страницы», а представляет собой базовую единицу, обеспечивающую функционирование системы возможностей.

Другими словами, программное обеспечение AI Native — это не просто добавление слоя «интеллектуального интерфейса» к традиционным системам, а переосмысление определения ключевых компонентов продукта.

### 3. Сравнение трёх типичных категорий продуктов

Если рассматривать только определения, система способностей по-прежнему может показаться абстрактной. В контексте конкретных продуктов эти различия в подходах становятся более очевидными.

#### **Сценарий**

##### **Традиционная функциональная система**

##### **Система возможностей AI Native**

##### Анализ данных

Отчеты, фильтрация, экспорт, запросы с фиксированными критериями

Понимание проблем, объяснение аномалий, построение аналитических цепочек, формулирование выводов

##### Служба поддержки

Система заявок, база знаний, стандартные шаблоны, ответы операторов

Понимание запросов, инструменты для управления заявками, правила поиска, формирование рекомендаций по ответам, предупреждение о рисках

##### Техническое обслуживание

Платформа оповещений, платформа журналов, руководства по устранению неполадок, ручное устранение неполадок

Агрегирование предупреждений, обобщение журналов, вывод первопричин, рекомендации по действиям, накопление опыта на основе прецедентов

Эти различия показывают, что граница между продуктами AI Native и традиционным программным обеспечением заключается не в «наличии или отсутствии модели», а в том, начинает ли система выполнять задачи более высокого уровня: понимание, принятие решений, организацию, сотрудничество и частичное выполнение.

## **4. Поведение программного обеспечения начинает зависеть от контекста**

Поведение систем, основанных на правилах, в основном зависит от программного кода, тогда как поведение программного обеспечения AI Native явно зависит от контекста. На одну и ту же проблему должны даваться разные ответы в зависимости от роли пользователя, истории взаимодействия, корпоративных данных и статуса задачи. Таким образом, контекст перестает быть второстепенным фактором и становится ключевым вкладом, определяющим качество продукта.

Это означает, что при проектировании продукта необходимо учитывать не только интерфейс взаимодействия, но и то, откуда берется контекст, как он организован, как он фильтруется и как обновляется. Производительность системы больше не зависит только от того, «правильно ли написан код», но и от того, «правильно ли предоставлена информация».

Это также является одним из фундаментальных отличий AI Native-программного обеспечения от традиционного: традиционное программное обеспечение основано на логической определённости, а AI Native-программное обеспечение — на качестве комбинации «способностей + контекста».

## **5. Программное обеспечение начинает участвовать в выполнении задач**

Самое важное изменение в AI Native-программном обеспечении заключается в том, что оно больше не является просто носителем пользовательского интерфейса, а начинает участвовать в процессе выполнения задач. Оно может понимать проблемы, предлагать рекомендации, выполнять отдельные шаги, отслеживать результаты и постоянно учиться. Такие системы по своей сути ближе к «сотруднику», чем к «набору инструментов».

Следовательно, центр тяжести конкуренции между продуктами также изменится. Раньше в центре внимания обычно находились функциональные различия, плавность процессов и удобство работы с интерфейсом; сейчас всё большее значение приобретают качество данных, проектирование возможностей, организация контекста, скорость экспериментов и замкнутый цикл обратной связи.

Как только система начинает участвовать в выполнении задач, ожидания пользователей от программного обеспечения также меняются: теперь они не просто просят «помочь мне действовать быстрее», а хотят, чтобы система «помогла мне выполнить эту задачу лучше».

## **6. Новые проблемы, возникающие в связи с парадигмой AI Native**

Именно потому, что программное обеспечение превращается из функциональной системы в систему возможностей, возникает ряд новых проблем, которые не были столь заметны в эпоху традиционного программного обеспечения.

●

Во-первых, определение продукта становится сложнее. Команды зачастую не могут заранее составить полный список функций, поскольку многие ключевые способности приходится выявлять с помощью экспериментов.

●

Во-вторых, границы системы становятся более сложными. Качество результатов зависит не только от кода, но и от моделей, данных, контекста, памяти и вызовов инструментов.

●

В-третьих, изменились типы рисков. Система больше не подвержена только «логическим ошибкам», но также может демонстрировать галлюцинации, превышение полномочий, неверные действия, а также необъяснимые и неотслеживаемые явления.

●

В-четвертых, изменились объекты разработки. Команды больше не занимаются только обслуживанием страниц и интерфейсов, а обслуживают цепочки возможностей, цепочки знаний, цепочки контекста и цепочки обратной связи.

●

В-пятых, изменился смысл запуска в производственную среду. После выхода продукта в реальную среду он не просто начинает обслуживать пользователей, но и раскрывает новые модели сбоев и новый материал для обучения.

В совокупности эти проблемы означают, что программное обеспечение AI Native не является естественным продолжением традиционных методов разработки, а представляет собой объект другого уровня сложности.

## 7. Почему для этого требуется новая методология разработки продуктов

Если система представляет собой систему с фиксированным набором функций, то классического подхода к разработке программного обеспечения обычно бывает достаточно: определение требований, разбиение на функции, планирование разработки, тестирование и запуск. Однако программное обеспечение AI Native сталкивается с иным типом проблем. Команда должна одновременно ответить на вопросы:

- Какую именно проблему мы решаем, а не просто накапливаем функции.
- Действительно ли эта возможность работает, а не только выглядит рабочей в демо-версии?
- Как эта возможность может превратиться из результатов экспериментов в стабильную функциональность продукта?
- где находятся границы этой возможности, риски, моменты ручного вмешательства и условия запуска;
- Как реальные отзывы после запуска будут учтены в следующем цикле определения задач и экспериментов.

Именно поэтому в книге главы 1 и 2 являются лишь введением, а не ответами. Главный вывод этой главы: когда программное обеспечение переходит к парадигме AI Native, разработка продукта должна перейти от «подхода, ориентированного на разработку функций» к «подходу, ориентированному на разработку возможностей».

Именно по этой причине в дальнейшем тексте будет предложена новая основная линия:

Определение направления -> Проведение испытаний -> Создание системы -> Аудит и разрешение на ввод в эксплуатацию -> Ввод в эксплуатацию

Эта основная линия не предполагает добавления дополнительного набора процессов, а предназначена для решения новых проблем в области продуктов, инжиниринга и управления, возникающих в рамках парадигмы AI Native.

Заключение к этой главе

Под термином «AI Native» подразумевается не просто «подключение модели к программному обеспечению», а использование возможностей ИИ в качестве ядра продукта. Программное обеспечение превращается из инструмента в помощника, из функциональной системы — в систему возможностей, из средства выполнения процессов — в участника выполнения задач. Такие изменения приносят не только новые возможности, но и новую сложность, новые граничные проблемы и новую ответственность за производство. Именно поэтому методология, описанная далее, является не просто дополнительным украшением, а необходимым условием для создания подлинно «AI Native»-продуктов.

## Глава 3. Необходимость модернизации методов программной инженерии

Вопросы, на которые будет дан ответ в этой главе:

- 

Почему существующие методы программной инженерии уже не способны в полной мере объяснить процесс разработки продуктов на базе ИИ?

- 

В чём заключаются границы между DevOps, MLOps и методологией AI Native Product ?

- 

Почему методология разработки продуктов также нуждается в обновлении?

- 

На каких проверенных практиках в основном основана эта методология?

Объяснение ключевых концепций и новых терминов

- 

DevOps: методология, основанная на создании замкнутого цикла непрерывной доставки, охватывающего разработку, развертывание и эксплуатацию.

- 

MLOps: метод построения замкнутого цикла моделирования, включающий данные, обучение, оценку, развертывание и мониторинг.

- 

AI Native Product Methodology: метод построения замкнутого цикла продукта, основанный на проблемах, экспериментах, инжиниринге, управлении и обратной связи от производства.

- 

Замкнутый цикл продукта: уделяет внимание не только тому, запущен ли код или модель, но и тому, сформированы ли необходимые возможности, создана ли ценность и находятся ли риски под контролем.

Понимание данной главы на примере

Если команда разрабатывает систему искусственного интеллекта для обслуживания клиентов, DevOps поможет ей обеспечить непрерывную доставку сервиса; MLOps поможет ей управлять моделями и потоками данных; но то, что действительно определяет, «следует ли сначала разработать рекомендации по ответам или сначала заняться выявлением рисков», «какие сценарии заслуживают дальнейшей инженерной проработки» и «когда можно перейти в реальную клиентскую среду», — это по-прежнему методология продукта, а не сама инженерная конвейерная линия.

Этот пример показывает, что существующие методологии не утратили своей актуальности — просто они решают разные задачи. Задача методологии AI Native Product Methodology — дополнить эти методологии продуктовыми решениями.

Описание процесса эволюции методологии

Методы реализации проектов -> Методология непрерывной доставки -> Методология замкнутого цикла модели -> Методология замкнутого цикла для продуктов на базе ИИ

По мере усложнения структуры системы методология также должна эволюционировать от подхода «как создать продукт» к подходу «как постоянно создавать правильные продукты».

## **1. Методология всегда меняется в соответствии с эволюцией систем**

Методы программной инженерии никогда не были абстрактной истиной, а представляли собой ответ на определенную структуру систем. Водопадный метод подходит для проектов со стабильными требованиями и четкими этапами; гибкие методы — для команд разработчиков, у которых требования часто меняются; DevOps решает задачи непрерывной интеграции, непрерывного развертывания и координации эксплуатации; MLOps решает задачу замкнутого цикла между данными, обучением моделей, развертыванием и мониторингом.

Все эти методы важны, но их основным объектом по-прежнему остаются «системы кода» или «системы моделей». Проблемы продуктов AI Native идут еще дальше: команды должны одновременно справляться с неопределенностью продукта, неопределенностью возможностей и неопределенностью бизнес-модели.

## **2. Откуда берется неопределенность в продуктах на базе ИИ**

При разработке традиционных продуктов можно исходить из двух предположений: во-первых, требования в целом можно определить; во-вторых, функции в целом можно реализовать. Продукты на базе ИИ, как раз, нарушают обе эти предпосылки. Команды часто могут лишь сначала сформулировать общее направление, например, «помочь ИИ ускорить ответы службы поддержки» или «помочь ИИ в поиске неисправностей при эксплуатации», но то, до какой степени это возможно, каким образом это будет реализовано и в каких сценариях можно начать внедрение, необходимо подтверждать экспериментально.

Это означает, что PRD больше не является отправной точкой — отправной точкой становятся эксперименты.

### **3. От инженерного замкнутого цикла к продуктовому замкнутому циклу**

DevOps делает акцент на замкнутом цикле от разработки до эксплуатации, MLOps — на замкнутом цикле от данных до модели, а методология AI Native Product Methodology — на полном замкнутом цикле продукта, включающем проблемы, возможности, инжиниринг, управление и период после запуска. Она объединяет продуктовый, экспериментальный и инженерный подходы в единую цепочку:

Определение направления -> Проведение экспериментов -> Создание системы -> Аудит и одобрение -> Эксплуатация в производственной среде

В этой цепочке эксперименты не являются второстепенным этапом технической проверки, а являются частью определения продукта; инжиниринг также не является заключительным этапом разработки, а представляет собой ключевой этап, на котором возможности превращаются в надежный продукт.

## 4. Теоретические истоки данной методологии

Эта методология не возникла на пустом месте; на самом деле она вобрала в себя четыре категории проверенных практик, просто перегруппировав их в единую линию, более подходящую для продуктов в области искусственного интеллекта.

•

Lean Startup: предоставляет базовую идею «проверки гипотез посредством экспериментов». Многие возможности продуктов ИИ невозможно определить заранее, поэтому необходимо выявлять границы возможностей с помощью экспериментов, а не сначала формулировать требования, а затем проверять их.

•

DevOps: обеспечивает замкнутый цикл от разработки до развертывания и мониторинга. После запуска продукта ИИ также требуется постоянное наблюдение, постоянный реверс и постоянная оптимизация, а не однократная поставка.

•

MLOps: обеспечивает замкнутый инженерный цикл между данными, моделями, развертыванием и мониторингом. Он напоминает команде, что модель — это не статический актив, а возможности системы меняются в зависимости от данных и обратной связи.

•

AI Engineering: предлагает концепцию системных компонентов, таких как Prompt, Agent, Tool, Memory, RAG и Workflow. Подход подчеркивает, что модель — лишь часть системы, а истинные возможности продукта обусловлены целостностью цепочки.

Если рассматривать только один из этих подходов, картина будет неполной. Lean Startup уделяет больше внимания проверке инноваций, DevOps — развертыванию и эксплуатации, MLOps — жизненному циклу модели, а AI Engineering — системной архитектуре. Методология AI Native Product Methodology же направлена на решение следующих задач: как продукт обнаруживается, создается, управляется, запускается и затем постоянно поддерживается.

## **5. Продукт-менеджеры и инженеры должны повысить свою квалификацию**

В рамках этой новой методологии продукт-менеджер не может ограничиваться лишь формулировкой требований, а инженер — лишь их реализацией. Продукт-менеджер должен понимать данные, возможности и оценку, а инженер — контекст, память, вызов инструментов и границы управления. Таким образом, подход к командной работе меняется с «доставки функций» на «обнаружение возможностей и управление ими».

Заключение к этой главе

Методология AI Native Product Methodology не призвана заменить DevOps или MLOps, а направлена на заполнение пробела в разработке продуктов на базе ИИ, касающегося того, «как продукт обнаруживается, определяется, стабилизируется и управляется». Это и методология разработки продуктов, и подход к работе, объединяющий продуктовые, экспериментальные, инженерные и коммерческие аспекты.

## Глава 4. Методология AI Native Product

Если в предыдущих трех главах был дан ответ на вопрос «почему нужна новая методология», то в этой главе мы ответим на следующие вопросы: что представляет собой эта методология, как она структурирует всю книгу и как она помогает реальным продуктам пройти путь от определения направления до непрерывной эволюции.

Вопросы, на которые будет дан ответ в этой главе:

- 

Что представляет собой методология AI Native?

- 

К каким типам продуктов и сценариям применима эта методология, а в каких ситуациях её нельзя применять напрямую?

- 

Почему это не линейный процесс, а система непрерывного цикла?

- 

Определение направления → Проведение экспериментов → Построение системы → Аудит и запуск → Производственная эксплуатация → Циклическая обратная связь — за что отвечает каждый из этих этапов?

- 

Чем обеспечивается связь между этапами, и как команда должна определять, когда можно переходить к следующему этапу?

- 

Какова связь между циклом продукта (Product Loop), циклом обнаружения ценности (Value Discovery Loop), циклом клиента (Customer Loop) и уровнем управления (Governance Layer), уровнем возможностей (Capability Layer)?

1. Это не блок-схема, а подход к работе

Традиционные методы разработки продуктов в большинстве случаев основаны на одном допущении: проблема относительно чётка, потребности можно определить заранее, а система в основном реализует заранее заданные функции. Однако продукты AI Native не в полной мере соответствуют этому допущению. Зачастую на начальном этапе команда знает лишь о том, что у пользователей есть определённый тип сложных проблем, но не знает, сможет ли модель действительно их решить, какие данные являются наиболее важными, а также в какой форме должен быть конечный продукт: система вопросов и ответов, Copilot, рабочий процесс или интеллектуальный агент.

Это означает, что разработка продуктов на базе ИИ не может опираться исключительно на линейную логику «сначала определить требования, а затем приступить к разработке». Команда должна одновременно справляться с четырьмя видами неопределенности:

- 

Неопределенность проблемы: какая проблема действительно заслуживает решения.

- 

Неопределённость возможностей

: на что на самом деле способна комбинация модели, данных, контекста и системы.

- 

Неопределенность формы: в каком виде — с каким взаимодействием и рабочим процессом — должен появиться продукт.

- 

Неопределенность рисков: где проявятся проблемы безопасности, доступа, затрат, надежности и границ ошибок.

Таким образом, предлагаемая в этой книге методология AI Native по созданию продуктов по сути представляет собой не набор блок-схем для инструментов управления проектами, а подход к организации неопределенности, экспериментов, инженерных работ и непрерывной эволюции.

## 1.1. Почему генеративный ИИ делает эту методологию жизнеспособной

Эта методология не была естественно обоснована на протяжении всей эпохи традиционного искусственного интеллекта. Её реальная применимость зависит от того, что генеративный ИИ уже обладает набором способностей, которые в прошлом не существовали одновременно: понимание вопросов на естественном языке, обработка неструктурированных данных, генерация разнообразных промежуточных результатов, непосредственное написание кода для создания работоспособных прототипов, вызов инструментов и постоянная корректировка результатов в процессе взаимодействия.

Без этих способностей основной цикл «определение направления → проведение экспериментов → построение системы → аудит и запуск → эксплуатация → обратная связь» вряд ли мог бы функционировать с такой скоростью, как сегодня. Команда по-прежнему может реализовывать проекты в области ИИ, но они будут ближе к последовательной модели «разработка алгоритмов + программная реализация», а не к циклической модели, в которой «определение проблемы, проверка возможностей, создание продукта и инженерная доработка» постоянно стимулируют друг друга.

Говоря более конкретно, генеративный ИИ привёл как минимум к пяти непосредственным изменениям:

- Он позволяет напрямую преобразовывать обсуждение проблем в тестируемые подсказки, примеры, наборы для оценки и сценарии взаимодействия.
- Впервые неструктурированные данные могут в массовом масштабе использоваться в экспериментах с продуктами без необходимости их предварительной полной стандартизации.
- Он позволяет быстро создавать и сравнивать эскизы процессов, функций, интерфейсов и системной документации.
- Он позволяет команде уже на этапе запуска экспериментов напрямую создавать рабочие демо-версии, а не ограничиваться обсуждением на бумаге.
- 

Это позволяет на этапе построения системы с помощью программных агентов ИИ напрямую преобразовывать результаты экспериментов в инженерную структуру, а не полностью полагаться на ручную переработку с нуля.

Таким образом, методология, представленная в этой книге, не сводится к абстрактному утверждению «нужно меняться, потому что пришёл ИИ», а основана на том, что генеративный ИИ уже превратил огромное количество промежуточных результатов между продуктом, дизайном и разработкой в объекты, которые можно быстро генерировать, быстро сравнивать и быстро итерировать. Именно благодаря этому запуск экспериментов превращается из вспомогательного этапа в основное поле битвы за продукт, а построение системы переносится с традиционного завершающего этапа НИОКР на этап перепрограммирования системы возможностей.

Также необходимо подчеркнуть, что это не означает простой замены менеджеров по продукту или инженеров. Точнее говоря, генеративный ИИ заменяет значительный объем работы по упорядочиванию данных, генерации решений, составлению документации и созданию демо-версий, которые ранее выполнялись вручную, но оценка проблем, определение при-

оритетов, выбор границ и принятие ответственности по-прежнему должны оставаться в ведении команды.

## 1.2. Для каких типов продуктов эта методология подходит больше всего

Эта методология не подходит «для всего, что связано с ИИ». На самом деле она подходит не для какой-то конкретной отрасли, а для определенного типа структуры задач. Точнее говоря, она лучше всего подходит для продуктов, обладающих следующими характеристиками:

- Ценность продукта определяется его способностями, а не только фиксированными функциями.

- Сама проблема несет в себе неопределённость и не может быть полностью сформулирована в виде правил с самого начала.

- Задачи зависят от естественного языка, неструктурированных данных, бизнес-опыта или комбинации контекстов.

- Пользователю требуется не система для ввода данных в поля, а система, способная помочь в понимании, анализе, генерации, принятии решений или продвижении задачи.

- У продукта есть возможность в процессе реального использования постоянно накапливать отзывы, примеры неудач и информационные ресурсы.

Поэтому данная методология, как правило, лучше подходит для следующих типов продуктов:

- AI Native-продукты на основе знаний, такие как корпоративные системы вопросов и ответов, «Copilot» на основе знаний, рабочие платформы на основе знаний.

- AI Native аналитические продукты, такие как анализ данных, анализ эксплуатации и обслуживания, диагностика бизнеса, атрибуция в службе поддержки.

- AI Native-продукты для совместной работы над процессами, такие как помощники по обслуживанию клиентов, помощники по продажам, операционные Copilot и помощники по внутренним процессам.

- Продукты для реализации задач на базе искусственного интеллекта (AI Native), такие как системы оркестрации задач, обладающие функциями рабочих процессов или интеллектуальных систем.

Общим для этих продуктов является не то, что они «используют большие модели», а то, что их основная ценность заключается в способности обрабатывать сложную информацию и решать сложные задачи.

Однако, если посмотреть с другой стороны, эта методология имеет значение не только для продуктов AI Native. Сегодня многие команды, хотя и продолжают заниматься разработкой традиционного программного обеспечения, SaaS, внутренних систем или функциональных продуктов, уже заметно изменили свой процесс разработки под влиянием ИИ: они используют ИИ-агентов для написания и исправления кода, а также для дополнения тестов; используют ИИ для создания прототипов, документации и эскизов процессов; а также привлекают ИИ к экспертной оценке, устранению ошибок и систематизации данных. В таком случае сам про-

дукт не обязательно является «AI Native», но подход к разработке уже движется в направлении «разработки с поддержкой ИИ».

Для таких команд методология, изложенная в этой книге, также ценна, просто им лучше подойдет «упрощённая версия», а не «полная версия». Точнее говоря:

- 

если основная ценность самого продукта заключается в возможностях ИИ, знаниях, реорганизации рабочих процессов или интеллектуальном взаимодействии, то в этом случае целесообразнее применять основную линию книги в полном объеме.

- 

Если же сам продукт по-прежнему представляет собой достаточно чётко определённую функциональную систему, но команда уже широко использует программирование с помощью ИИ и цепочку инструментов ИИ в разработке, то более целесообразно извлечь из книги принципы определения направлений, проведения экспериментов, построения систем, аудита и выпуска, а также управления инструментами, чтобы применять их для управления исследованием требований, проверкой прототипов, выпуском кода и аудитом инструментов.

Другими словами, эта книга наиболее эффективно применима к продуктам, изначально разработанным с учетом ИИ (AI Native); однако вторая по значимости область её применения — это «программные команды, уже вступившие в эпоху ИИ-поддержки разработки». Говоря более прямо, этот уровень можно охарактеризовать как «ИИ продолжает использовать ИИ»: ИИ не только присутствует в продукте, но и начинает появляться в самом процессе разработки продукта.

### 1.3. Какие типы продуктов не подходят для прямого применения

Не менее важно отметить, что данная методология подходит не для всех продуктов. Если суть продукта по-прежнему заключается в функциональной системе с высокой степенью определённости, низкой изменчивостью и низкой неопределённостью, то полное применение методологии, изложенной в этой книге, зачастую будет излишне обременительным.

К наиболее типичным непригодным формам относятся:

- 

Чисто детерминированные инструментальные продукты, такие как калькуляторы, фиксированные отчеты, простые формы документооборота, системы CRUD с полностью понятными правилами.

- 

Системы, в основном полагающиеся на обработку транзакций, а не на обработку возможностей

, например, учет запасов, стандартные процедуры утверждения, системы фоновой настройки, запускаемые по фиксированным условиям.

- 

Продукты типа «AI-декорации», представляющие собой простое добавление окна чата к исходному продукту, поскольку они не приводят к реальной переработке функциональных возможностей продукта или рабочих процессов.

- 

Одноразовые страницы мероприятий, маркетинговые страницы, результаты краткосрочных проектов, поскольку у них недостаточно длинный цикл обратной связи и не требуется полный методологический цикл.

Эти ситуации не означают, что нельзя использовать ИИ, и не означают, что данная книга полностью бесполезна, а лишь говорят о том, что не стоит с самого начала организовывать работу по полной основной схеме: «определение направления → запуск экспериментов → построение системы → аудит и одобрение → запуск в производство → обратная связь». Зачастую более целесообразно продолжать опираться в основном на традиционную разработку программного обеспечения, а затем извлекать и применять отдельные принципы из этой книги, например: сначала с помощью экспериментального развертывания проверить, действительно ли прототип, сгенерированный с помощью ИИ-кодирования, пригоден для использования; с помощью аудита и утверждения проверить полномочия и границы данных цепочки ИИ-инструментов; с точки зрения производственной эксплуатации отслеживать сигналы сбоя и необходимости отката после запуска в производство.

## **1.4. В каких сценариях нецелесообразно внедрять данную методологию**

Помимо формы продукта, многие конкретные сценарии также не подходят для непосредственного применения данной методологии, даже если команда субъективно очень хочет это сделать.

Наиболее распространённые категории неподходящих сценариев:

- 

Несоответствие требований к данным: отсутствие достаточного объема внешних данных или внутренних бизнес-данных, либо наличие данных, но невозможность их обезличивания, авторизации или структурирования.

- 

Чрезвычайно низкий уровень допустимой ошибки, при этом невозможность сохранить возможность ручного вмешательства: например, принятие решений с высоким уровнем риска, таких как окончательный медицинский диагноз, юридические заключения, предоставление финансовых кредитов, управление опасным оборудованием и т. д., непосредственно с помощью системы

- 

Требования к отклику предполагают строгую оперативность и определённость: система должна стабильно выдавать доказательно верные результаты в кратчайшие сроки; в таких случаях традиционные системы на основе правил зачастую оказываются более надёжными.

- 

Слишком низкая частота сценариев и слишком низкая ценность: проблема возникает всего несколько раз в год, но требует запуска полного цикла экспериментов, управления и производства, при этом соотношение затрат и результатов зачастую не оправдывает себя.

- 

Команда не может получить реальную обратную связь: отсутствуют ранние пользователи, отсутствуют производственные сигналы, невозможно проанализировать случаи неудач, что приводит к сбою ключевого цикла методологии.

Особое внимание следует уделить второй категории. В случае сценариев с высоким риском ИИ не является полностью неприменимым, но лучше всего подходит в качестве «уровня вспомогательного принятия решений», «уровня предварительного анализа» или «уровня теневого управления», а не в качестве непосредственного исполнителя.

## 1.5. Метод быстрого принятия решений

Если команда хочет быстро определить, следует ли применять данную методологию, ей следует сначала задать четыре вопроса:

- Заключается ли основная ценность данной задачи в понимании сложной информации, принятии полуструктурированных решений или выполнении сложных задач?
- Имеет ли команда достаточно данных, и можно ли эти данные обезличить, обеспечить к ним доступ и получать их на постоянной основе?
- Позволяют ли результаты системы наблюдение, корректировку, ручное управление и постоянную обратную связь?
- Имеет ли данный сценарий достаточно высокую частоту или достаточно большую ценность, чтобы оправдать вхождение в непрерывный цикл?

Если на большинство из этих четырёх вопросов ответ «да», то данная методология, как правило, подходит для полноценного применения. Если на первые два вопроса ответ «нет», это означает, что пока нет достаточной почвы даже для определения направления или запуска экспериментов; если на третий вопрос ответ «нет», это означает, что данный сценарий скорее не подходит для того, чтобы ИИ принимал на себя прямую ответственность; если на четвёртый вопрос ответ «нет», это означает, что, скорее всего, речь идёт лишь о локальной функциональной единице, а не о направлении развития продукта, ориентированного на ИИ (AI Native), которое стоит развивать. В этом случае более разумным подходом зачастую является не отказ от всего содержания данной книги, а переход к «упрощённому использованию»: сохранить определение проблемы, быстрое тестирование, аудит инструментов, контроль запуска и механизм отката, но не разворачивать цикл в полном объёме.

## 1.6. Непосредственные преимущества данной методологии

Многие команды, впервые увидев эту основную линию, ошибочно полагают, что она более сложна, чем традиционный процесс, поскольку в ней явно выделены несколько этапов: определение направления, проведение экспериментов, построение системы, аудит и запуск в производство. Однако в случае с продуктами искусственного интеллекта она, как правило, оказывается более эффективной. Причина проста: основные затраты при разработке продуктов ИИ обычно связаны не с написанием нескольких дополнительных страниц документации, а с преждевременным началом технической реализации, слишком поздним обнаружением проблем, слишком поздним сужением границ и, в конечном итоге, с переделкой на более дорогостоящих этапах.

Если свести всё к одному предложению, преимущество этой методологии заключается не в таком абстрактном понятии, как «большая стандартизация», а в «меньшем количестве действий вслепую, более быстром сходимости, меньшем объёме доработок и более простом повторном использовании».

Говоря более конкретно, преимущества этой методологии в основном сосредоточены в четырёх направлениях: более раннее выявление ошибочных направлений, более быстрое сближение компетенций, меньший объём доработок на этапе реализации, а также реальная интеграция оценки, управления и обратной связи от производства в единую линию. Другими словами, её ценность заключается не в «более полном процессе», а в том, что команда может раньше отказаться от неверного направления, более уверенно продвигаться в правильном направлении и превращать каждый раунд тестирования и запуска в актив для следующего раунда.

*Краткий обзор сферы применения и ограничений*

### **Критерии оценки**

**Более подходит для полного применения**

**Не рекомендуется применять в полном объеме**

**Более подходит для упрощённого применения**

### **Форма продукта**

AI Native: продукты для знаний, аналитики, совместной работы и выполнения задач

Чисто детерминированные инструменты, простые операции CRUD, страницы однократного использования

Обычное программное обеспечение, но в процессе разработки широко используется ИИ

### **Типы задач**

Понимание сложной информации, полуструктурированные суждения, выполнение сложных задач

Задачи с полностью ясными правилами, высокой определённой и низкой степенью изменчивости

Исследование локальных возможностей, верификация прототипов, управление цепочкой инструментов

### **Условия данных**

Достаточно данных, возможность обезличивания, авторизация, постоянное обновление

Отсутствие данных, невозможность обезличивания, невозможность предоставления доступа

Имеются локальные данные, достаточные для проведения испытаний в ограниченном масштабе

Структура рисков

Возможность наблюдения, корректировки и ручного управления

Крайне низкая отказоустойчивость и отсутствие возможности ручного вмешательства в крайнем случае

Сначала создаются вспомогательный, рекомендательный и тестовый уровни

Наиболее подходящий подход

Полное следование основной линии

Необязательно строго следовать всему циклу, описанному в этой книге

Выделить направления, провести быстрое тестирование, провести аудит инструментов, установить контрольные точки запуска

## Краткий обзор входов и выходов пяти этапов

### Этап

#### Ключевые входы

#### Основные задачи

#### Типичные результаты

Определение направления

Поиск проблем, бизнес-цели, текущее состояние данных

Определение проблемы, оценка ценности, анализ условий данных, выявление границ риска

Резюме направления, гипотезы о ценности, условия начала эксперимента

Проведение эксперимента

Краткое изложение направления, выборка, пакет данных, гипотезы о возможностях

Проведение экспериментов по оценке возможностей, продукта и коммерческой модели, выявление моделей неудач

Сигналы сходимости, примеры неудач, рекомендации по принятию решений в отношении продукта

Построение системы

Конвергирующие возможности, границы неудач, цепочки данных

Разбиение системы, организация контекста, интеграция знаний, мониторинг и откат

Инженерно-реализуемая система, интерфейсы управления, наблюдаемая структура

Аудит и одобрение

Структура системы, результаты оценки, доказательства «Shadow»

Определение границ разрешения, способы утверждения, ручное управление, условия отката

Заключение «go / no-go», стратегия запуска, ограничения при вводе в эксплуатацию

Производственная эксплуатация

Одобренная система, показатели мониторинга, пользовательский трафик

Наблюдение за реальным поведением, выявление сбоев, сбор обратной связи и ее использование

Новые образцы, новые тесты, новые проблемы, новый раунд приоритезации

## 1.7. Объяснение ключевых концепций и новых терминов данной главы

- ИИ: если не указано иное, термин «ИИ» в данной книге в основном относится к генеративному искусственному интеллекту, основанному на крупных языковых моделях и соответствующих мультимодальных моделях, а не является общим обозначением всей области традиционного искусственного интеллекта.
- Методология продуктов AI Native: это не общие «руководства по разработке ИИ», а набор методологий, организующих работу вокруг выявления возможностей, продуктового управления и непрерывной обратной связи. Она наиболее подходит для полноценного применения в продуктах AI Native, но также может быть упрощена и использована в программных командах, где ИИ используется для поддержки разработки.
- Продуктовый цикл (Product Loop): основной цикл от определения проблемы до обратной связи после запуска, отвечающий на вопрос «Как продукт создаётся на постоянной основе».
- Цикл обнаружения ценности (Value Discovery Loop): цикл, проходящий от экспериментов к обнаружению ценности, проверке на рынке и корректировке направления развития продукта, отвечающий на вопрос «Как обнаруживается и проверяется ценность?».
- Цикл взаимодействия с клиентами (Customer Loop): цикл, проходящий от раннего тестирования продукта клиентами до получения обратной связи, внедрения улучшений и расширения, отвечающий на вопрос «Как клиенты способствуют росту продукта?».
- Уровень управления (Governance Layer): уровень управления, пронизывающий весь процесс и охватывающий полномочия, риски, оценку, одобрение, откат и границы ответственности.
- Уровень возможностей (Capability Layer): основа возможностей, лежащая в основе продукта, включающая модели, данные, инструменты, модули навыков, интеллектуальные системы, системы памяти, контекст и системы знаний.
- Этапные контрольные точки (Stage Gate): минимальные условия, которые команда должна выполнить перед переходом к следующему этапу; процесс не зависит от субъективных оценок.
- Краткое изложение направления (Direction Brief): краткий документ с определением этапа направления, используемый для уточнения проблемы, гипотез о ценности, условий данных и границ рисков.
- Теневая система (Shadow System): система, позволяющая ИИ работать параллельно в реальных условиях, но не принимая непосредственного управления бизнесом; используется для сбора доказательств и выявления рисков.
- Цикл (Loop): это не добавление новой главы, а механизм обратной связи в производственной среде, который возвращает реальный мир в основную линию методологии.

2. Основная линия: определение направления -> проведение испытаний -> построение системы -> аудиторское одобрение -> эксплуатация в производственной среде -> циклическая обратная связь

Рис. 4-1 Общая схема методологии

Определение направления -> Проведение испытаний -> Построение системы -> Проведение аудита и выдача разрешения -> Производственная эксплуатация -> Циклическая загрузка -> Определение направления обратного потока

Источник обратной связи: - Примеры неудач - Поведение пользователей - Сигналы из бизнес-процессов - Ручная корректировка

«Циклическая обратная связь (Loop)» в данном контексте не является отдельным крупным этапом, а означает, что мониторинг, обратная связь, случаи сбоев, поведение пользователей и бизнес-сигналы из производственной среды вновь поступают в следующий цикл определения направления и проведения экспериментов. Иными словами, производственная эксплуатация — это не конечная точка, а отправная точка для нового цикла обучения продукта.

Эти шесть этапов можно рассматривать как шесть последовательных вопросов:

•

Определение направления: что именно мы определяем как проблему, каковы её границы, выполняются ли условия для данных?

•

Проведение экспериментов: можно ли действительно решить подобные проблемы с помощью ИИ, и каким образом это следует проверять?

•

Построение системы: как превратить проверенные возможности в стабильную систему?

•

Аудит и запуск: можно ли безопасно, надёжно и контролируемо ввести эту систему в эксплуатацию?

•

Эксплуатация: как система ведет себя в реальных условиях и какие новые проблемы при этом выявляются?

•

Обратная связь: как эти отзывы используются в следующем цикле определения задачи, экспериментов и построения?

Эта основная линия составляет «цикл продукта» (Product Loop), описанный в данной книге, и является самой ключевой частью всей методологии.

## 2.1. Что на самом деле означает каждый узел на блок-схеме

Если рассматривать эту схему лишь как перечень этапов, она останется слишком абстрактной. Более понятным подходом будет рассматривать каждый узел как рабочую задачу.

- Определение направления — это не «запуск проекта», а определение границ проблемы, оценка данных и определение пределов.

- Начало тестирования — это не «проверка модели», а проверка возможностей, поиск оптимальной формы продукта и уточнение целей.

- Создание системы — это не «начало написания кода», а преобразование проверенных возможностей в стабильную систему.

- Одобрение аудита — это не «окончательное утверждение», а четкое определение границ допуска, ответственности за риски и способа запуска в эксплуатацию.

- Производственная эксплуатация — это не «уже сдано», а переход к реальному использованию, выявление реальных сигналов и реальных сбоев.

- Обратная связь — это не «анализ результатов», а преобразование этих сигналов в основания для принятия решений о следующем цикле разработки и определения приоритетов экспериментов.

### 2.2 Что на самом деле передают стрелки на блок-схеме

Стрелки на блок-схеме — это не формальная передача ответственности, а передача информации, доказательств и решений.

- Определение направления → Проведение экспериментов: передаются определение проблемы, условия данных, ограничительные границы и экспериментальные гипотезы.

- Проведение экспериментов → Построение системы: передаются доказательства возможностей после сходимости, модели сбоев, базовые показатели оценки и решения по продукту.

- Создание системы → Аудит и допуск: передаются структура системы, цепочки полномочий, мониторинг журналов, механизмы ручного управления и отката.

- Аудит и допуск → Производственная эксплуатация: передаются стратегия допуска, сценарии «Shadow»/поэтапного внедрения, требования к мониторингу и условия отката.

- Производственная эксплуатация → обратная связь: передаются данные о поведении пользователей, случаях сбоев, сигналах о затратах, ручных корректировках и подтверждении ценности.

- Циклическая обратная связь → Определение направления: передаются приоритеты следующего цикла, корректировки направления и вопросы, требующие повторного тестирования.

Таким образом, «процесс» в данной методологии по сути не является очередью задач, а представляет собой непрерывное накопление, интерпретацию и передачу доказательств вверх по цепочке.

## **2.3. Помимо основной вертикальной линии этапов, существует также горизонтальная линия взаимодействия человека и машины**

В черновике есть одно суждение, которое стоит вынести на первый план: помимо вертикальной основной линии «определение направления -> проведение экспериментов -> построение системы -> аудит и запуск -> эксплуатация -> обратная связь», в всей методологии существует ещё одна горизонтальная линия взаимодействия человека и машины.

Эта поперечная линия не означает, что «в каждый узел нужно добавить ручной этап», а означает, что на любом этапе необходимо ответить на один и тот же вопрос: что в данный момент делает ИИ, что делает человек, кто отвечает за определение направления, кто отвечает за подтверждение ключевых узлов, кто отвечает за мониторинг, вмешательство и обеспечение резервных мер во время работы.

Если свести это к одному общему принципу, то он будет следующим:

- 

Человек отвечает за направление

, границы, принятие решений, одобрение и обеспечение резервного варианта.

- 

ИИ отвечает за обработку данных, анализ проблем, генерацию решений, ускорение выполнения, помощь в мониторинге и систематизацию обратной связи.

В последующих главах — в главе 08 — это будет развернуто в модель взаимодействия человека и машины на уровне процессов, а в главе 16 — в дизайн сотрудничества человека и машины на уровне пользовательского опыта.

## **3. Что на самом деле является результатом каждого этапа**

Методология полезна не потому, что этапы в ней хорошо названы, а потому, что она чётко определяет, что должно быть результатом каждого этапа.

### 3.1. Результаты определения направления

Цель определения направления заключается не в том, чтобы расширить список требований, а в том, чтобы сформулировать определение направления, готовое для эксперимента. Оно должно, как минимум, отвечать на следующие вопросы:

- 

Каковы пользователи и бизнес-задачи.

- 

Почему эту проблему стоит решать с помощью ИИ, а не традиционного программного обеспечения.

- 

Достаточно ли внешних и внутренних бизнес-данных для проведения эксперимента?

- 

Определены ли требования по обезличиванию данных, правам доступа, соблюдению нормативных требований и пределы рисков?

- 

целесообразнее ли в данный момент проверить возможности, или лучше отказаться от этого направления.

Поэтому типичным результатом определения направления обычно является не PRD, а документ, больше напоминающий краткое изложение направления (Direction Brief): определение проблемы, гипотезы о ценности, условия данных, ограничения и обоснование начала эксперимента.

### 3.2. Результаты проведения эксперимента

Цель проведения эксперимента заключается не в создании демо-версии, похожей на продукт, а в накоплении достаточно убедительных доказательств возможностей. В результате должны быть получены:

- Результаты экспериментов по ключевым сценариям и примеры неудач.
  - Сравнительные выводы по промптам, рабочим процессам, агентам или способам подачи данных.
  - Основные критерии оценки и базовые показатели для
  - Рекомендации по сведению целей и принятию решений о продуктах.
  - Решение о целесообразности начала разработки системы.
- Иными словами, результатом проведения экспериментов должны быть «доказательства» и «выводы», а не просто «шум».

### 3.3. Результаты построения системы

Цель построения системы заключается в преобразовании доказательств возможностей в системные возможности. После перехода на этот этап приоритеты команды изменятся:

- Как способности инкапсулировать в навыки (Skill), рабочие процессы (Workflow), агенты (Agent) или сервисные модули.
  - Как обеспечить стабильную подачу данных, контекста, памяти и знаний.
  - Насколько совершенны механизмы авторизации, ведения журналов, мониторинга, обработки ошибок и отката.
  - Завершена ли переработка кода из экспериментальной версии в производственную.
- Таким образом, результатом построения системы становится не «работающий эксперимент», а система, готовая к аудиту и принятию решения о вводе в эксплуатацию.

### 3.4. Результаты аудита и одобрения

Цель аудита и одобрения заключается не в том, чтобы поставить окончательную подпись, а в том, чтобы четко определить, можно ли запускать систему в эксплуатацию. В ходе аудита рассматриваются следующие вопросы:

- права доступа, превышение полномочий, утечка данных и операционные риски;
  - Надёжность вывода данных, обработка сбоев и механизмы ручного управления.
  - тестовые среды «Shadow» и «Grey», ручная проверка, объем разрешения к эксплуатации и условия отката;
- Заключение «go / no-go» и его обоснование.
- Настоящим результатом аудита перед запуском является решение о запуске, а не формальный перечень.

### 3.5. Результаты эксплуатации в производственной среде

Цель эксплуатации в производственной среде — не просто «запуск в эксплуатацию», а получение реальных сигналов от системы в реальных условиях. Производственная среда обеспечивает:

- данные о поведении пользователей и обратную связь;
- реальные случаи сбоев и сценарии с высоким риском;
- Информацию о затратах, задержках, стабильности и цепочках вызовов.
- Какие возможности действительно создают ценность, а какие лишь кажутся умными.

Вся эта информация в конечном итоге станет входными данными для цикла (Loop) и, в свою очередь, повлияет на определение направления и приоритетов экспериментов в следующем раунде.

### **3.6. Понимание этой основной линии в одном предложении**

Если сжать всю методологию в одно предложение, то она организует не «создание ИИ с нуля», а «то, как высокочастотная бизнес-задача постепенно определяется, постепенно проверяется, постепенно воплощается в инженерные решения, а затем безопасно внедряется в реальную среду». В последующих главах, посвященных примерам, эта основная линия будет раскрыта с трех точек зрения: управление высокими рисками, сложный эффект в эксплуатации сервисов и конкурентные преимущества продукта.

## **4. Переход между этапами осуществляется через контрольные точки, а не на основе интуиции**

Рис. 4-2. Схема контрольных точек этапов

Определение направления -> Контрольный пункт 1: Наличие необходимых данных  
Начало испытаний -> Шлюз 2: поступил сигнал сходимости  
Построение системы -> Шлюз 3: доказательства системы полны  
Одобрение аудита -> Шлюз 4: Условия разрешения определены  
Производственная эксплуатация

Проблема многих команд заключается не в отсутствии этапов, а в том, что переход от одного этапа к другому происходит исключительно на основе интуиции. Истинная ценность методологии заключается в том, что она устанавливает «этапные контрольные точки» для каждого шага вперед.

## **4.1. «Шлюз» «Определение направления — развертывание эксперимента»**

Не все проблемы заслуживают немедленного экспериментирования. Перед переходом к этапу проведения экспериментов должны быть выполнены как минимум три условия:

- Проблема должна быть достаточно четко сформулирована и заслуживать применения ИИ.

- Должны быть обеспечены основные условия для работы с данными, включая внешние данные, внутренние бизнес-данные и схему обезличивания.

- Пределы риска уже предварительно определены, и команда не окажется с самого начала в явно неконтролируемой зоне.

## 4.2. Переход от экспериментальной фазы к построению системы

Этот этап чаще всего становится предметом ошибочных суждений. Хорошие результаты эксперимента не означают, что следует немедленно приступить к внедрению в производство. Перед переходом к построению системы необходимо получить более убедительные сигналы о сходимости:

- ключевые сценарии должны быть неоднократно проверены, а не просто случайно сработать один раз.
  - Причины неудач в основном поддаются объяснению с помощью метода , и команда знает, что можно делать в пределах определённых границ, а что — за их пределами.
  - Форма продукта и целевая аудитория начинают консолидироваться, и проект перестает быть просто «интересным с технической точки зрения».
  - Команда сформировала четкое решение относительно продукта, а не продолжает просто накапливать эксперименты.
- Другими словами, эксперименты достигли порога, позволяющего приступить к созданию системы: речь идет не о том, что «модель впечатляет», а о том, что «возможности уже достаточно стабильны и заслуживают воплощения в продукте».

### 4.3. Этап «Создание системы» → «Одобрение аудита»

Перед началом аудита и одобрения команда должна, как минимум, ответить на следующие вопросы:

- 

Ясны ли границы системы?

- 

Являются ли права доступа и цепочки поставки данных понятными?

- 

Проведена ли базовая оценка ключевых возможностей и обеспечена ли их отслеживаемость?

- 

Возможен ли откат, остановка системы или переход на ручное управление в случае сбоя?

- 

Если эти базовые условия не выполнены, то система ещё не соответствует даже критериям «готовности к аудиту».

## **4.4. Шлюз «Одобрение аудита» → «Производственная эксплуатация»**

Этот этап — не просто «прохождение тестирования». Более полное решение о допуске обычно включает:

- Сначала тестирование в тестовой среде (Shadow) или сразу постепенное внедрение (Grey Release).
- На каких этапах сохраняются ручная проверка и ручное управление.
- Каких пользователей  
, какие задачи и какие сценарии следует разрешить в первую очередь.
- При появлении каких показателей или событий необходимо выполнить откат.  
Поэтому вывод аудита о запуске скорее заключается в том, «каким образом запустить», а не просто «можно ли запускать».

## **5. На самом деле в процессе происходит не линейное продвижение, а сближение целей**

Если смотреть только на схему, эта методология кажется пятиэтапным процессом. Но на самом деле она реализуется не по линейной траектории, а путем постоянного сближения целей внутри каждого этапа.

Наиболее типичный пример наблюдается на этапе развертывания эксперимента:

- вначале, возможно, проверяется способность модели.
- Затем выясняется, что действительно ключевым фактором является способ подачи данных.
- Еще позже становится ясно, что проблема заключается не в качестве ответов, а в реорганизации процессов и форме продукта.
- В конечном итоге наступает этап «конвергенции целей / принятия решений о продукте», на котором определяется, какие направления стоит продолжать, какие отложить, а какие прекратить.

Иными словами, развертывание эксперимента — это не один большой экспериментальный бассейн, а процесс постоянного сближения проблем, возможностей, формы продукта и оценок ценности. Создание системы также не завершается за один раз: часто возникающие новые проблемы заставляют вернуться к предыдущему этапу.

Поэтому методологию, описанную в этой книге, точнее назвать «циклической системой со шлюзами», а не строго последовательным «водопадным» процессом.

## **6. «Теневая система» — важный мост от эксперимента к производству**

Shadow System играет очень важную роль во всей книге, поскольку предоставляет более надежный механизм перехода, чем «непосредственная замена исходной системы».

Под термином «Shadow» подразумевается не то, что ИИ сразу же берет на себя управление реальными бизнес-процессами, а то, что он работает параллельно в реальных условиях, постоянно сравнивая результаты и выявляя проблемы. Например:

- позволить ИИ давать рекомендации по реальным заявкам в службу поддержки, но пока не отправлять их автоматически;
  - Пусть ИИ анализирует реальные оповещения, но сначала сравнивает свои выводы с выводами операторов.
  - Включить ИИ в бизнес-процессы, но сначала использовать его в качестве «теневого уровня принятия решений» для сбора доказательств.  
Его ценность заключается в трёх аспектах:
  - Позволяет команде получать в реальной среде данные, более достоверные, чем результаты экспериментов.
  - Предоставить аудиторам возможность принимать решения об одобрении на основе реальных рисков, а не гипотетических предположений на бумаге.
  - Превратить переход к производственной эксплуатации из «резкого переключения» в «контролируемую эволюцию».
- Таким образом, «теневая система» — это не просто отдельная концепция, а связующий механизм в рамках всей методологии, объединяющий проведение экспериментов, построение системы, одобрение аудита и запуск в производственную эксплуатацию.

## **7. Три цикла вместе составляют полную методологию**

До этого момента в книге речь шла о самом основном цикле — цикле продукта (Product Loop). Однако одного цикла для понимания продукта ИИ недостаточно. Ведь продукт ИИ включает в себя не только «как его создать», но и «как обнаружить его ценность» и «как клиенты стимулируют его рост».

Поэтому в данной книге методология далее структурирована в виде трёх циклов:

Продуктовый цикл (Product Loop): Определение направления -> Проведение испытаний -> Построение системы -> Аудит и одобрение -> Производственная эксплуатация -> Возврат в цикл

Цикл обнаружения ценности (Value Discovery Loop): Проведение испытаний -> Выявление ценности -> Тестирование на рынке -> Определение цены -> Корректировка направления развития продукта

Цикл взаимодействия с клиентами (Customer Loop): Возможности для тестирования → Первые клиенты → Получение обратной связи → Улучшение продукта → Расширение клиентской базы

Эти три цикла отвечают на три категории вопросов:

- Цикл продукта (Product Loop): как продукт постоянно совершенствуется.
  - Цикл обнаружения ценности (Value Discovery Loop): как ценность обнаруживается, проверяется и преобразуется в направление развития продукта.
  - Цикл взаимодействия с клиентами (Customer Loop): как реальные клиенты способствуют росту продукта, а не просто пассивно принимают его в конце.
- Самый важный момент заключается в том, что запуск экспериментов — это не только отправная точка для технической проверки, но и отправная точка для обнаружения ценности и изучения поведения ранних клиентов. Бизнес-модель появляется не тогда, когда продукт полностью созрел, а зачастую начинает формироваться уже на этапе экспериментов.

## **8. Два уровня поддержки: уровень управления (Governance Layer) и уровень возможностей (Capability Layer)**

Помимо трех циклов, в этой книге также используется концепция «двух уровней» для объяснения того, почему продукты AI Native отличаются от традиционного программного обеспечения.

Уровень управления (Governance Layer) Основной цикл продукта Уровень возможностей (Capability Layer)

### **8.1 Уровень управления (Governance Layer)**

Уровень управления (Governance Layer) — это не просто окончательная проверка безопасности, а часть методологии, пронизывающая весь процесс. Он охватывает:

- риски, полномочия, соответствие нормативным требованиям и границы данных.
- оценку, объяснимость  
, мониторинг и принятие решений о допуске;
- ручное вмешательство, откат, реагирование на инциденты и непрерывное управление.

Иными словами, управление — это не дополнение к аудиту и одобрению, а логика принятия решений, пронизывающая каждый этап.

## 8.2. Уровень возможностей (Carability Layer)

Уровень возможностей (Carability Layer) представляет собой фундамент возможностей, лежащий в основе продукта. Обычно он включает:

- Модель
- Data
- Инструмент
- Навык
- Агент
- Память
- Контекст
- Система знаний

Эти способности сами по себе не являются продуктом, но именно они определяют, обладает ли продукт действительно стабильной, повторно используемой и управляемой структурой способностей. Проблема многих команд заключается в том, что на первый взгляд они занимаются разработкой продукта, но на самом деле базовый уровень способностей еще не сформирован.

## 9. Как читать всю книгу

Если рассматривать эту главу как общую схему всей книги, то последующие главы фактически последовательно разворачивают эту схему.

- В главе 5 обсуждается определение границ направления, даются ответы на вопросы о том, как определять границы и как оценивать условия данных.
- В главах с 06 по 10 описывается проведение экспериментов, включая исследование форм продукта, реорганизацию процессов, сближение целей и тестирование в тестовой среде.
- В главах с 11 по 16 рассматривается построение системы и обсуждается, как способности превращаются в систему.
- В главе 17 обсуждается процесс аудита и утверждения, а также дается ответ на вопрос, как система проходит утверждение.
- Глава 18 посвящена производственной эксплуатации и отвечает на вопрос, как осуществлять мониторинг, обратную интеграцию и переход к следующему циклу.
- В главах 22–24

рассматриваются циклы обнаружения ценности и циклы взаимодействия с клиентами.

•

В Приложении А приведены графические представления односторонней структуры, модели «трех колец» и двухуровневой структуры.

Иными словами, цель данной главы не в том, чтобы заменить последующие разделы, а в том, чтобы создать единую систему координат для них.

## 10. Конечная цель данной методологии

Конечной целью методологии AI Native не является то, чтобы команда быстрее нагромождала функции, а то, чтобы она быстрее обнаруживала правильные способности, более уверенно воплощала их в продукты и более устойчиво преобразовывала продукты в обратную связь и сложные проценты.

Эта «сложная прибыль» в основном формируется за счёт трёх факторов:

•

Экспериментальная сложная прибыль: команда всё лучше понимает, что стоит пробовать, а что нет.

•

Композиционный эффект продукта: возможности преобразуются в многократно системы, а не остаются одноразовыми результатами проектов.

•

Сложный доход от данных: обратная связь, неудачи и пользовательское поведение в производственной среде, в свою очередь, улучшают принятие решений о продукте в следующем цикле.

Таким образом, эта методология на самом деле направлена не на решение столь узкой задачи, как «как ускорить исследования и разработки с помощью ИИ», а на то, как в эпоху ИИ команда может на основе компетенций, данных, управления и обратной связи непрерывно создавать растущие продуктовые системы.

Также следует добавить, что эта методология не предполагает своей вечной актуальности. По сути, она представляет собой этапный ответ на текущие реалии в области генеративного ИИ, агент-инжиниринга и продуктового инжиниринга. Если в будущем создание агентов (agent building) достигнет достаточной зрелости и сможет интегрировать в более стабильные платформенные возможности те задачи, которые сегодня в значительной степени требуют ручного выполнения — такие как определение направлений, проведение экспериментов, построение систем, аудит, проверка и управление эксплуатацией, — то эта методология, скорее всего, будет значительно упрощена или даже заменена новыми подходами. Иными словами, то, что действительно хочет сохранить эта книга, — это не какой-то фиксированный процесс, а способность постоянно переосмысливать подход к разработке продуктов в свете появления новых технологических парадигм.

## **Конец ознакомительного фрагмента.**

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, [купив полную легальную версию](#) на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.