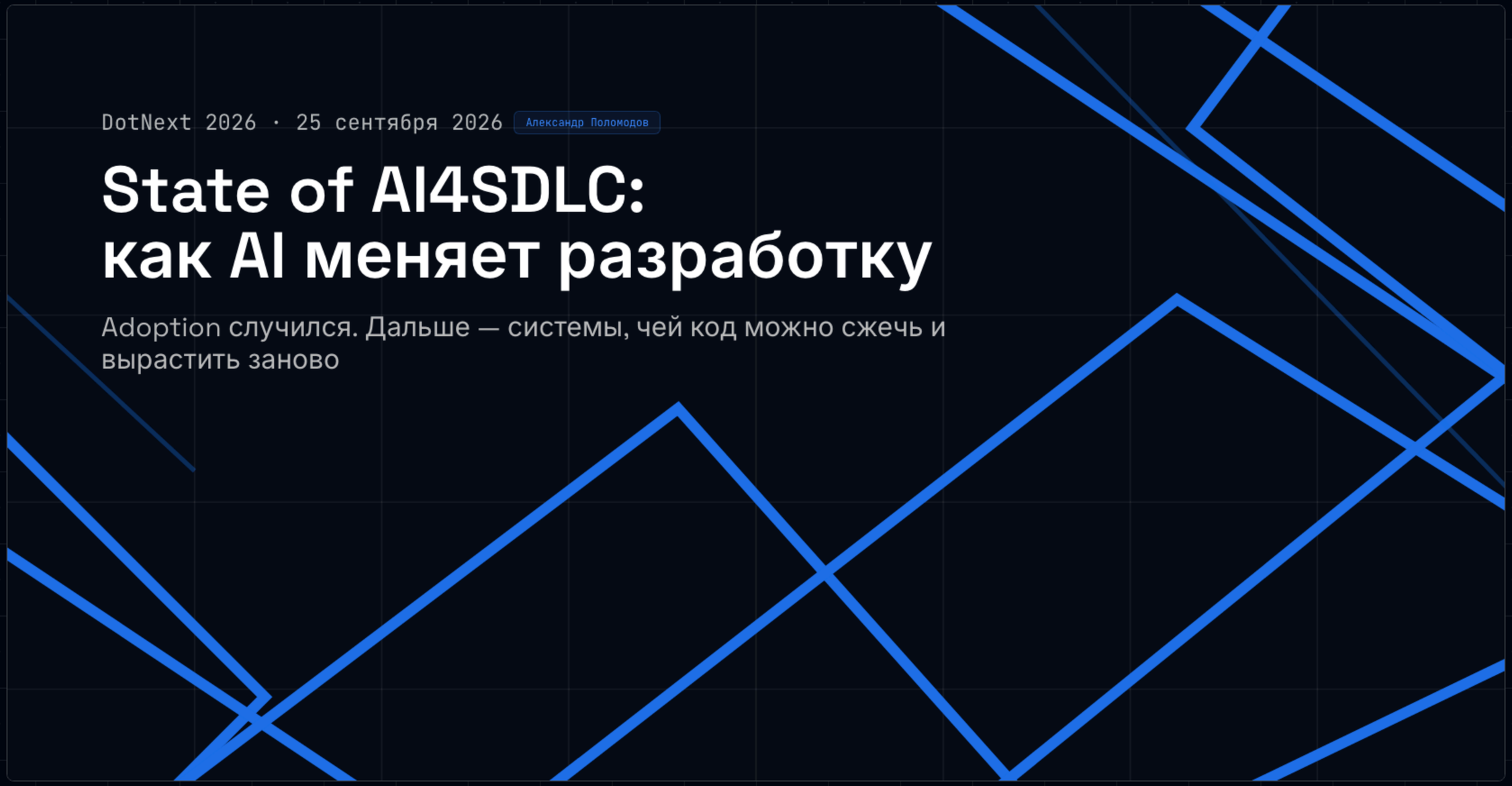
DotNext 2026 · 25 сентября 2026

Александр Поломодов

State of AI4SDLC: как AI меняет разработку

Adoption случился. Дальше — системы, чей код можно сжечь и вырастить заново



Код больше не сложная часть

Инструменты менялись быстрее процессов

Четыре части одного разговора

План 45 минут

- Индустрия: внедрение есть, системного эффекта нет
- Финтех 10 000+: платформа, агенты, метрики
- Regenerative Software: код, которому суждено умереть
- Инженер: намерение, контекст и проверка

Часть 1

01

Что происходит с индустрией

Срез 2026: Sonar, DX, Harness, Perforce и наше AI4SDLC Research

Внедрение выше доверия

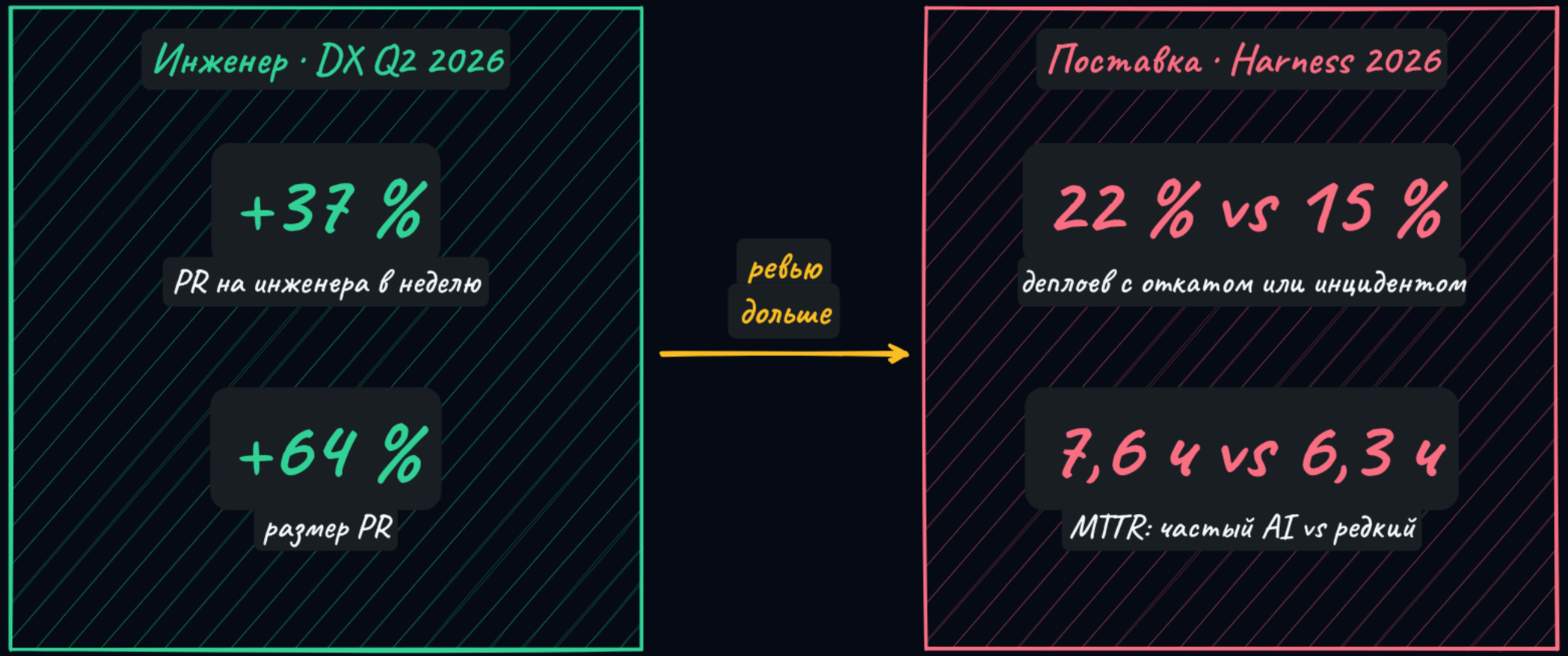
Внедрение

JetBrains 2026 · SO pulse 04.2026

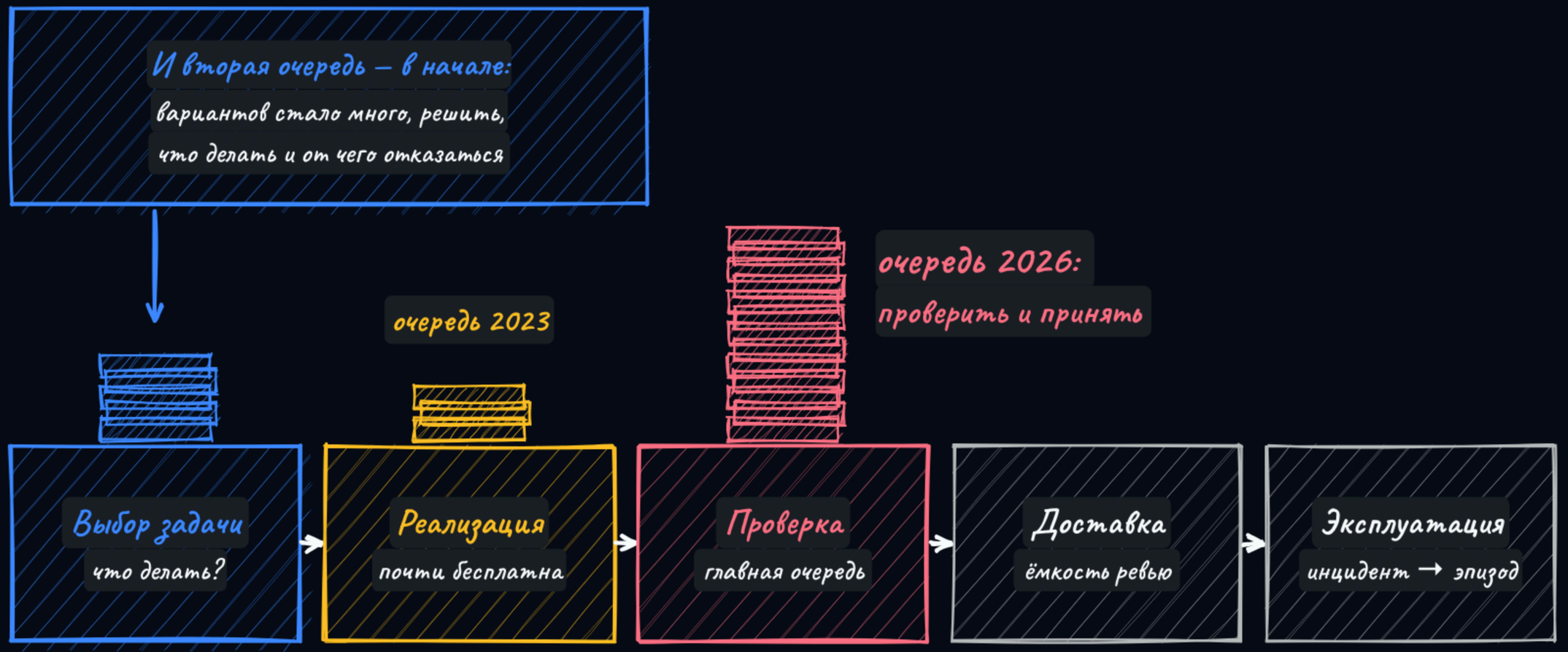
Доверие и контроль

SO pulse 04.2026 · Sonar 2026

Код быстрее, поставка — нет



Узкое место переезжает вправо



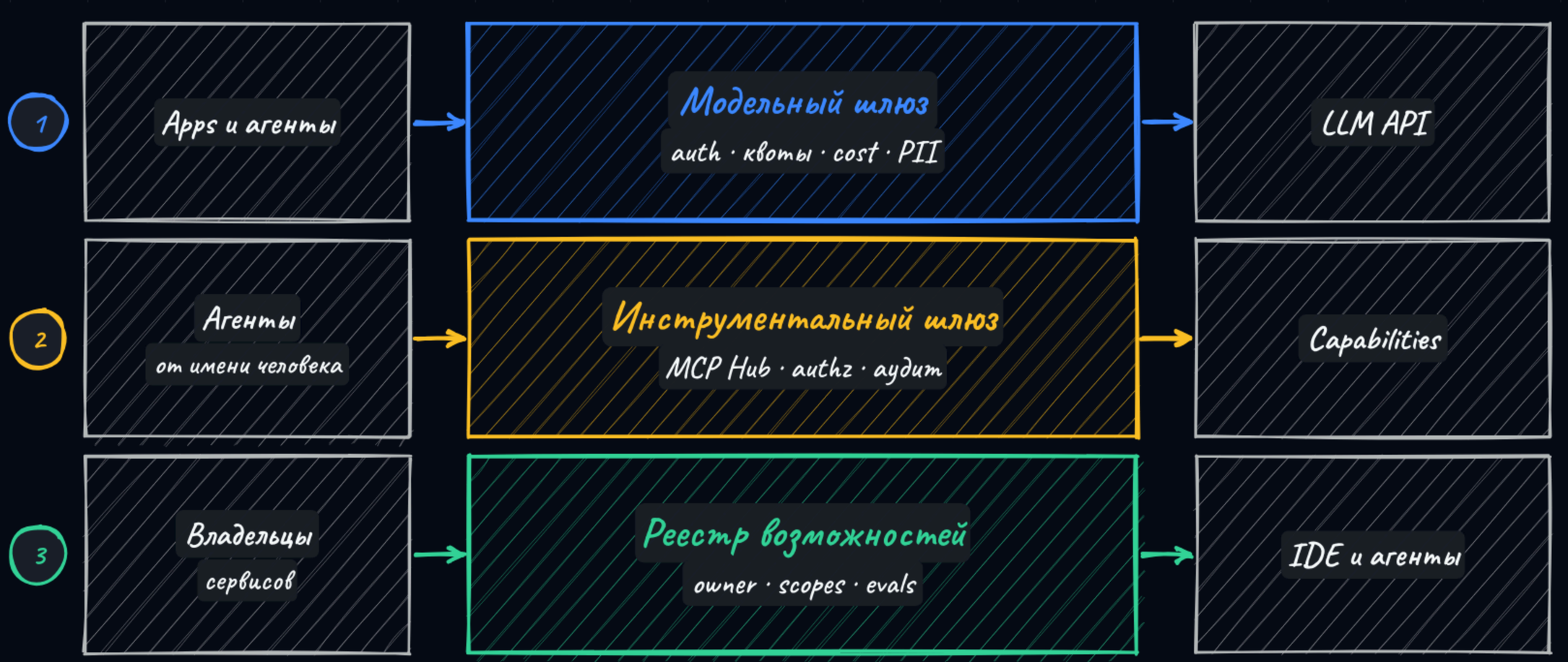
Часть 2

02

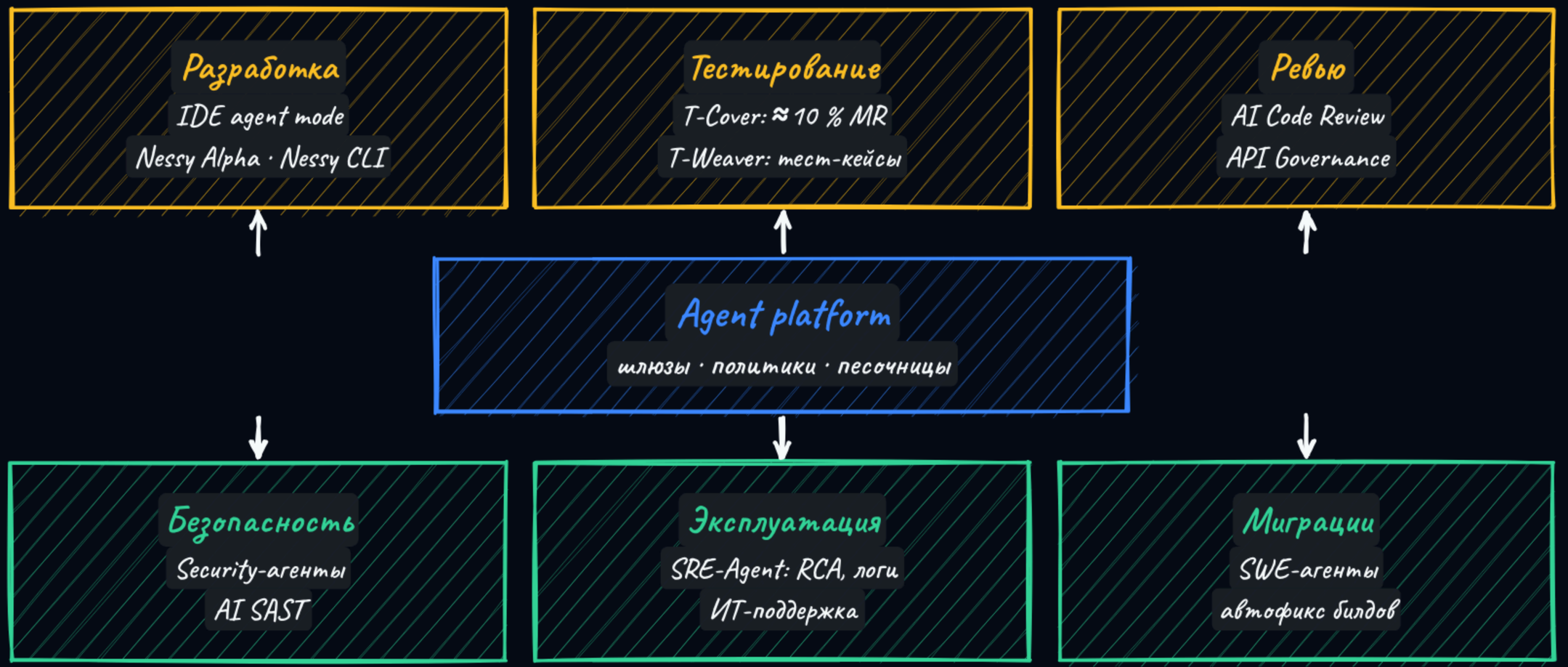
Как это выглядит в финтехе

10 000+ инженеров, тысячи сервисов, один периметр контроля

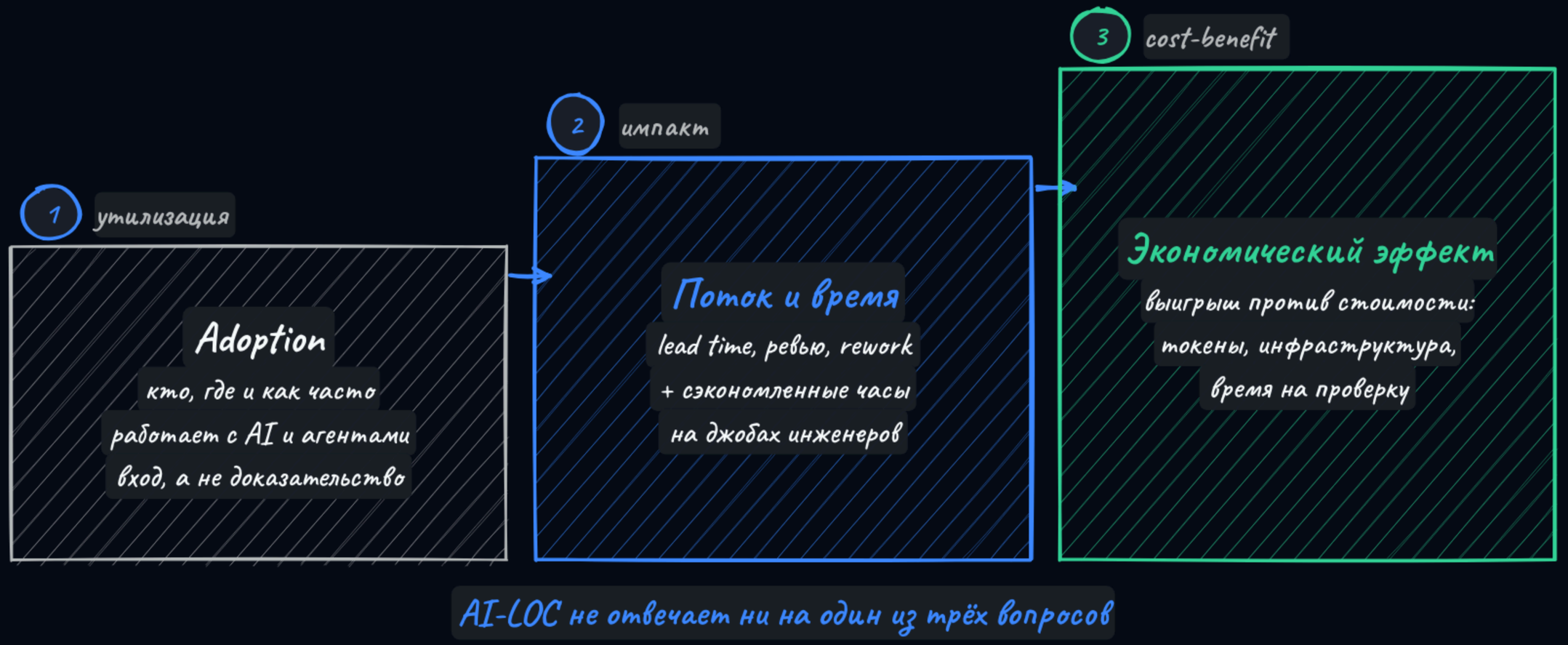
Три слоя агентной платформы



Агенты по всему SDLC



Что считать вместо AI-LOC



AI-LOC не отвечает ни на один из трёх вопросов

Часть 3

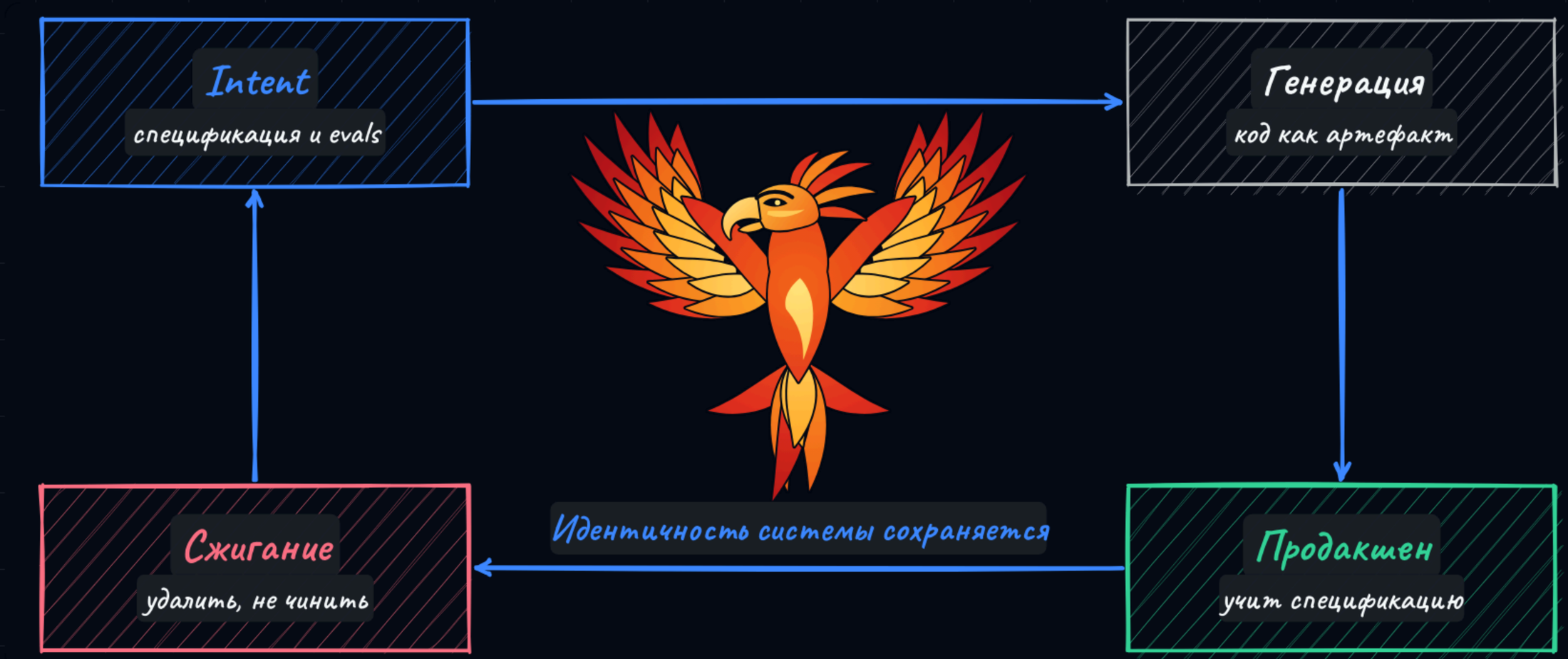
03

Regenerative Software

По книге Чада Фаулера (O'Reilly): что должно пережить реализацию?

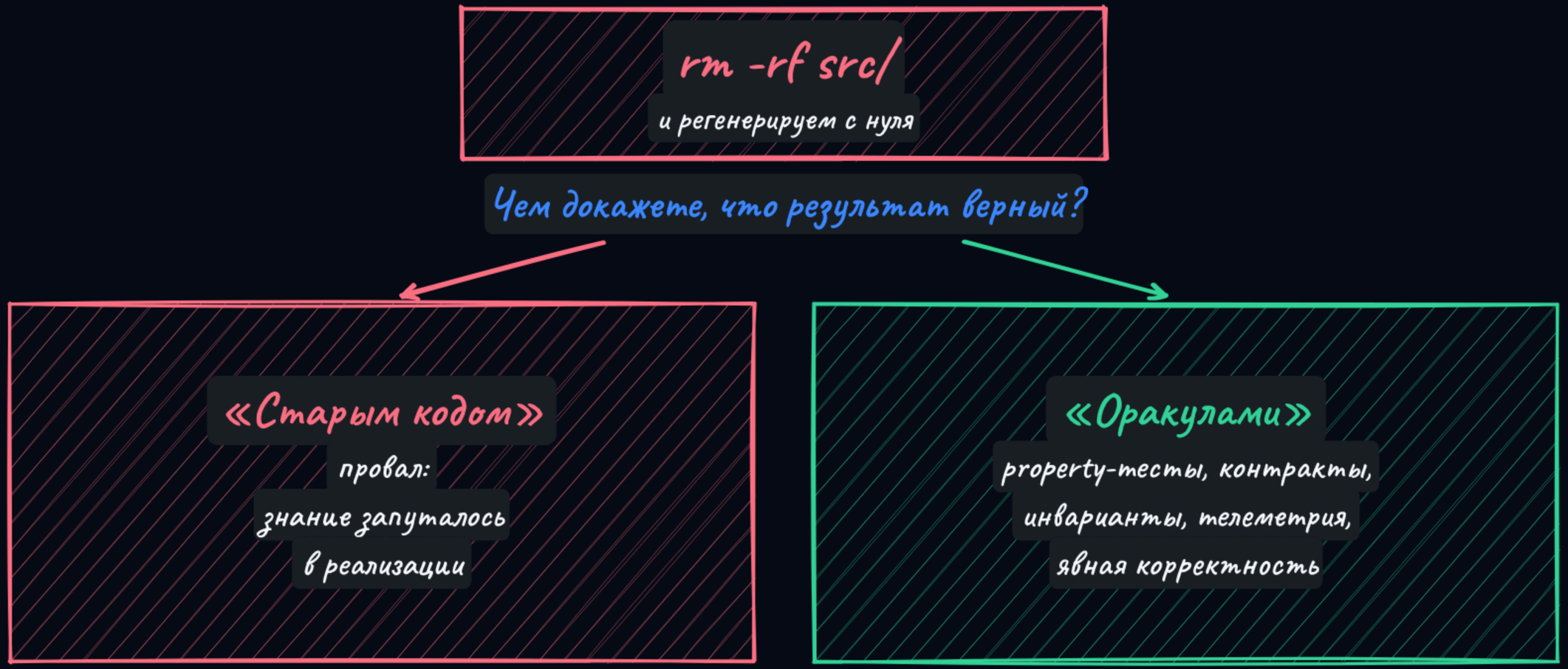
Код, которому суждено умереть

По Чаду Фаулеру



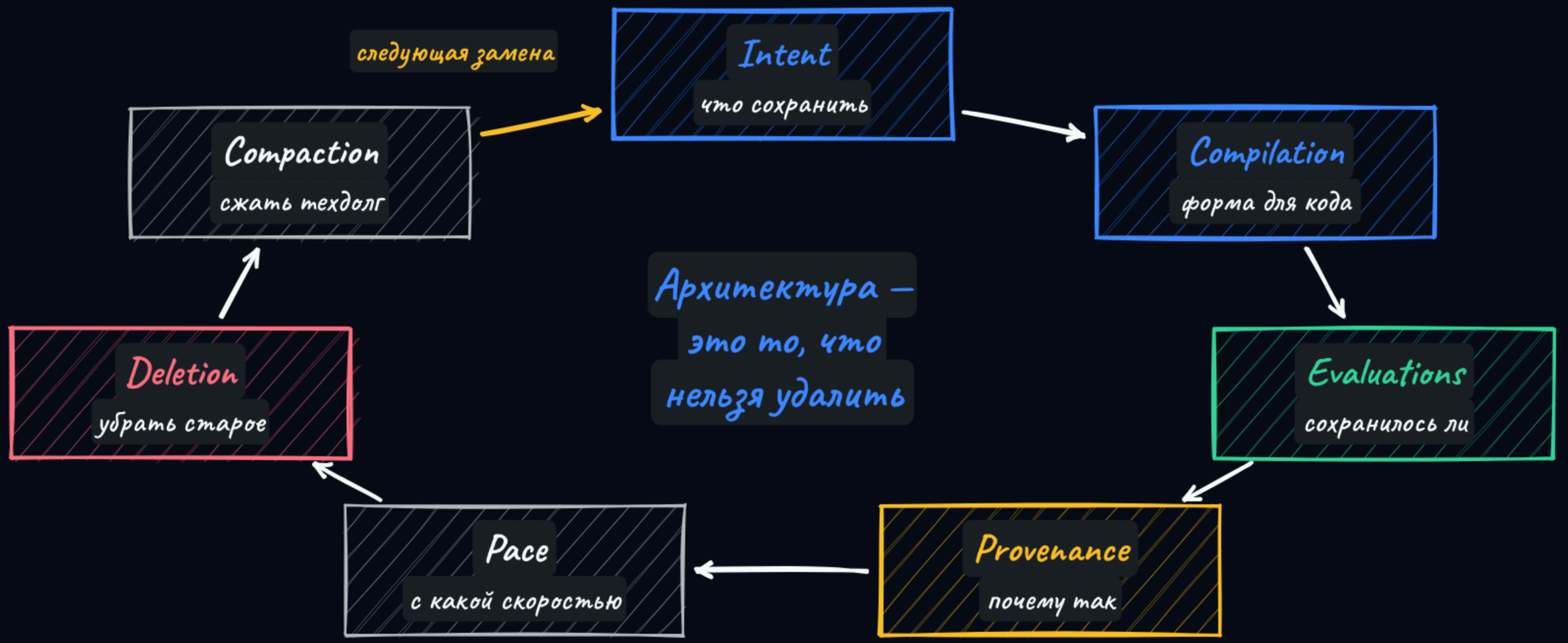
Тест на удаление

По Чаду Фаулеру



Семь неудаляемых примитивов

По Чаду Фаулеру



Intent: обещание, не механизм

По Чаду Фаулеру

Механизм

OrderService.validatePayment()
проверяет inventory
до PaymentGateway.authorize()

Намерение

платёж — только после
подтверждения остатков;
не списывать за отсутствующее

Scope
где действует

Invariant
что истинно

Reason
почему важно

Негативное пространство: не списать дважды, не уйти в минус по остаткам

Знакомые практики acceptance criteria · BDD · design by contract · SLO · ADR

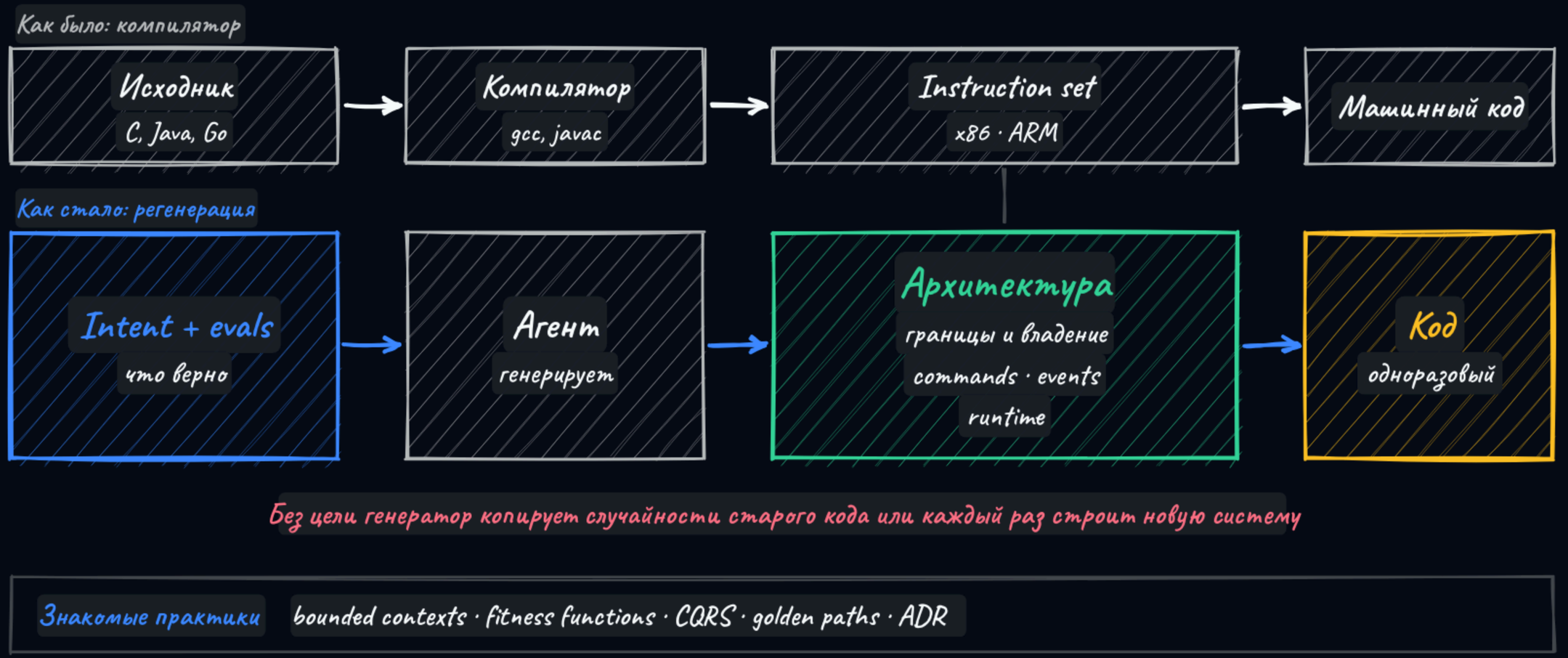
Ни единого разрыва

Что должен был сказать intent

Выход: записать инвариант или сделать вызовы идемпотентными

Архитектура — цель компиляции

По Чаду Фаулеру



Фреймворк стал архитектурой

Compilation: чёткие границы

Узкий API сделал реализацию заменяемой: Rust уже обслуживает 95 % запросов

Проверяем поведение, а не код

По Чаду Фаулеру

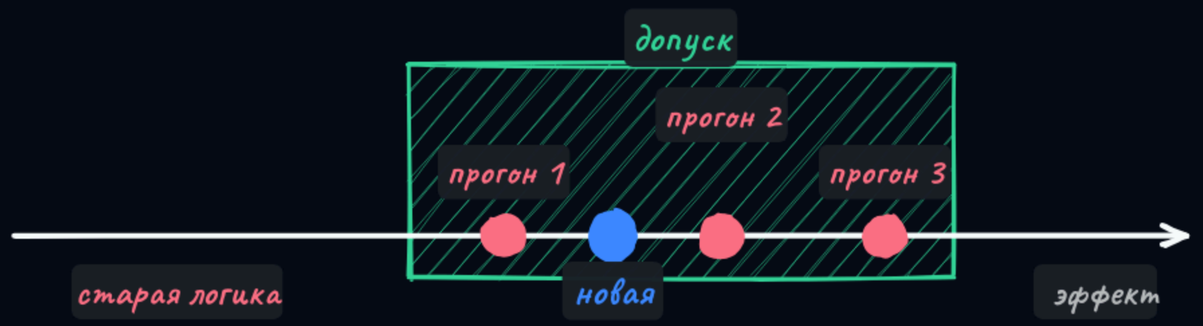
Новая A/B-платформа
другой сплит трафика
и другой расчёт итогов

Проверка «1-в-1»
итоги должны точно
совпасть со старыми

Старый расчёт
недетерминирован:
прогоны дают разное

Совпадение в допуске
новое решение принято,
старое заменили

Итог одного и того же эксперимента



Оракул — поведение
а не старый код:
сравниваем итоги
с допуском

Знакомые практики TDD/ATDD · property-based · contract tests · регрессии из инцидентов · canary · chaos

Чем проверять поведение

Способ	Что проверяет	Пример оценки
Контракты	интерфейс ведёт себя, как обещано	«коды ошибок и поля ответа не меняются»
Property	инварианты на любых входах	«последний товар: ровно один заказ»
Регрессии	уроки инцидентов и багов	«двойное списание не повторится»
Скорость	задержки, ресурсы, цена	«p99 ответа не больше 50 мс»
Отказы	таймауты, ретраи, рестарты	«падение ДЦ не рвёт сессии»
Безопасность	authz, изоляция, утечки	«чужой счёт недоступен»
Canary	реальное поведение в проде	«новая версия ведёт себя как старая»

Каждый инцидент становится новой оценкой

Provenance: почему, а не что

По Чаду Фаулеру

Лог решений вместо памяти

2020 · ArchDays
как мы принимаем архитектурные решения: RFC и ADR

2023 · прижилось
документ до решения, итог — в логе решений в большинстве управлений

2026 · агенты
находят по ADR историю изменений: почему система такая

Зачем писать
«информация не замыкается на одном человеке, а доступна всем заинтересованным»

Без лога
«через год никто уже не помнит, почему команды были нарезаны именно так»

Лог решений кажется бюрократией, пока не нужно ответить «почему»

Pace: у слоёв разная скорость

По Чаду Фаулеру

Почти не движется

фундаментальные слои

лишь при неопровержимых фактах

Медленный слой

auth · сессии · billing

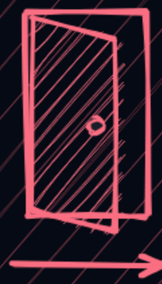
staged rollout · наблюдение

Быстрый слой

рекомендации · эксперименты

регенерация еженедельно

Безос: два типа решений

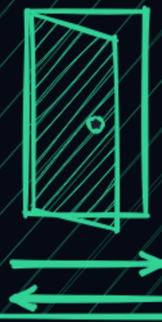


One-way door

необратимо:

медленно,

с доказательствами



Two-way door

обратимо:

быстро,

откат дешёвый

Знакомые практики — pace layering · change management · semver · feature flags · staged rollout

Aerospike: решает слой, а не diff

Плановый апгрейд версии Aerospike

Новая версия Aerospike
под сервисом аутентификации

та же схема данных, казалось безопасным

скрыто глубоко в движке БД

Скрытый медленный слой
1+1 ДЦ: раскол кластера по сети
логика вторичных ключей в новой версии — другая

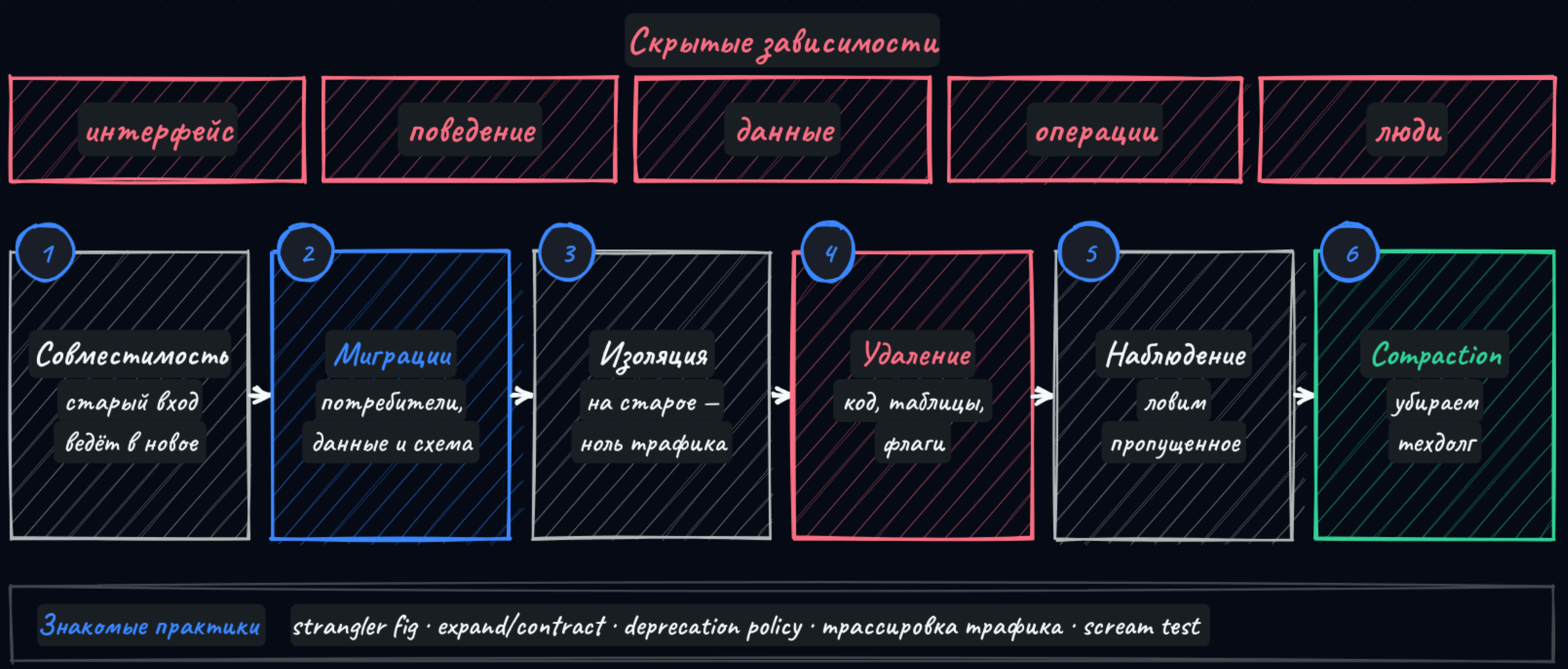
всплыло только при сбое сети

Радиус: открытые сессии SSO
большой отказ при сбое одного ДЦ

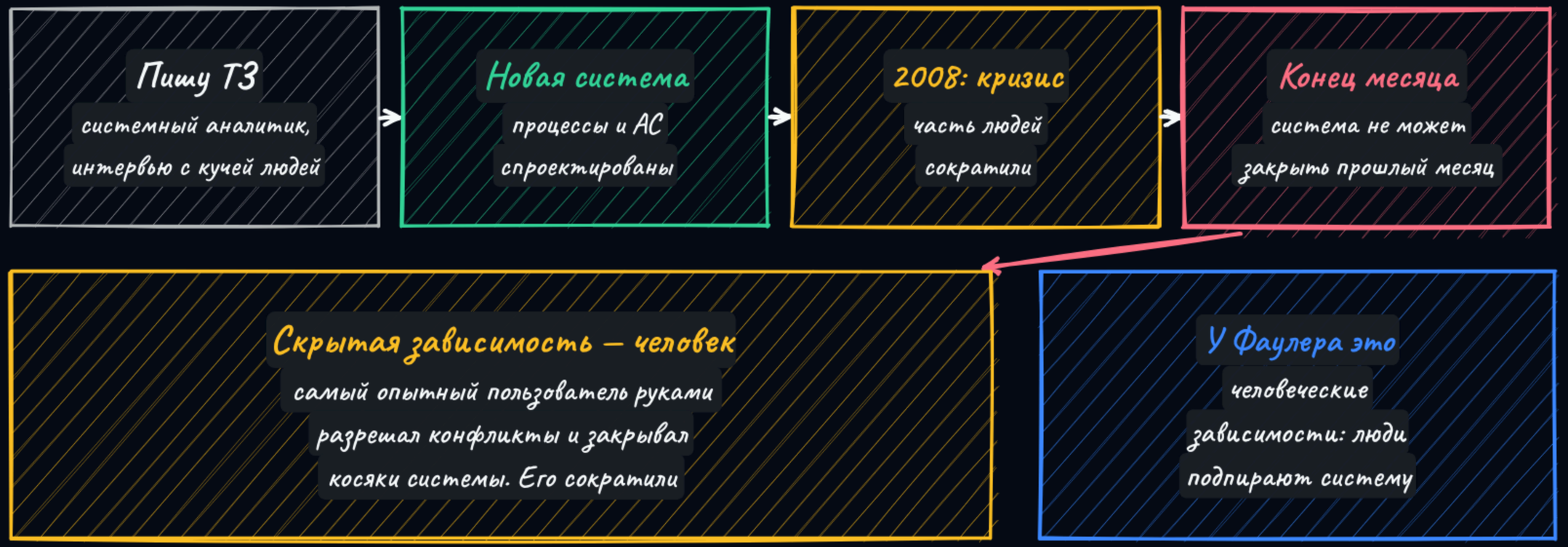
Слой, а не размер diff, решает, что будет дальше

Deletion: удалить безопасно

По Чаду Фаулеру



2008: месяц, который не закрылся



Кто заметит, если это исчезнет? И что исчезнет вместе с человеком?

Compaction: пять уровней

По Чаду Фаулеру

Compaction на потоке

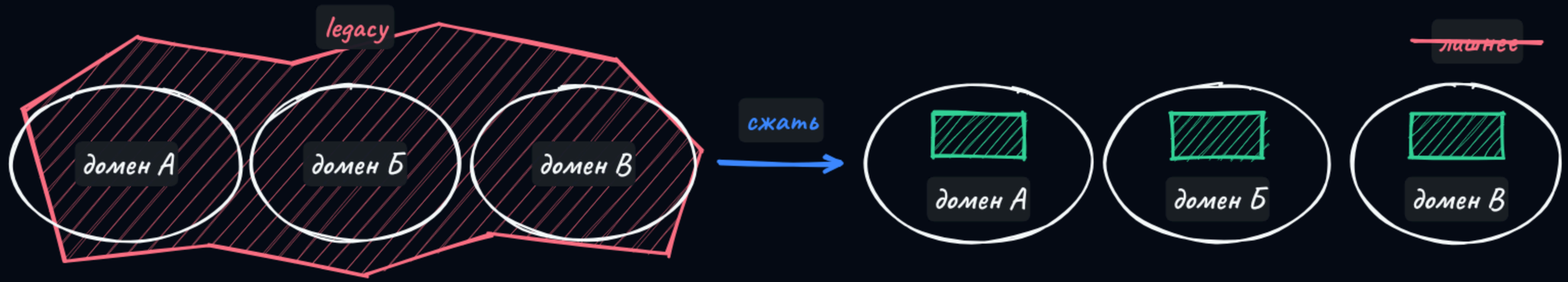
2022 · «Прометей»
стоп-слова в стеке:
Oracle и не только

Археология
legacy разрослось
за границы доменов

Не переносить 1:1
иначе уедет
и весь техдолг

Было: одна система на три домена

Цель: каждая функция — в своём домене



Импортозамещение — шанс вернуться меньше, а не перенести всё как есть

Примитивы — знакомые практики

По Чаду Фаулеру

Код помнит, организация — нет

По Чаду Фаулеру

2016 · переезд tinkoff.ru на новую платформу: год разработки, 50+ человек

Что умел старый сайт

страницы видны поиску (SEO)

веб-аналитика и трекинг

механики форм привлечения

за 3 месяца до переезда в новой версии этого нет

Цена переезда «as is»

сотни миллионов ₽

доп. затрат на привлечение

Что сделали

проектная команда и план доработок · релизы 2 раза в неделю вместо ~3 недель

Извлеките, что помнит старая система, до того как её заменить

Где регенерация ломается

По Чаду Фаулеру

Антипаттерны

- ✗ Регенерировать приложение целиком
- ✗ Пропустить предпосылки и evals
- ✗ Владеть кодом, а не системой

Предпосылки

- ✓ Стабильные границы и контракты
- ✓ Evals снаружи реализации
- ✓ Один владелец у каждого состояния

Вероятностно внутри, детерминированно на краях

Часть 4

04

Как меняется работа инженера

Одинаковая платформа даёт разный эффект в разных командах

Что отличало успешные команды

Команды Разный эффект

Без эффекта

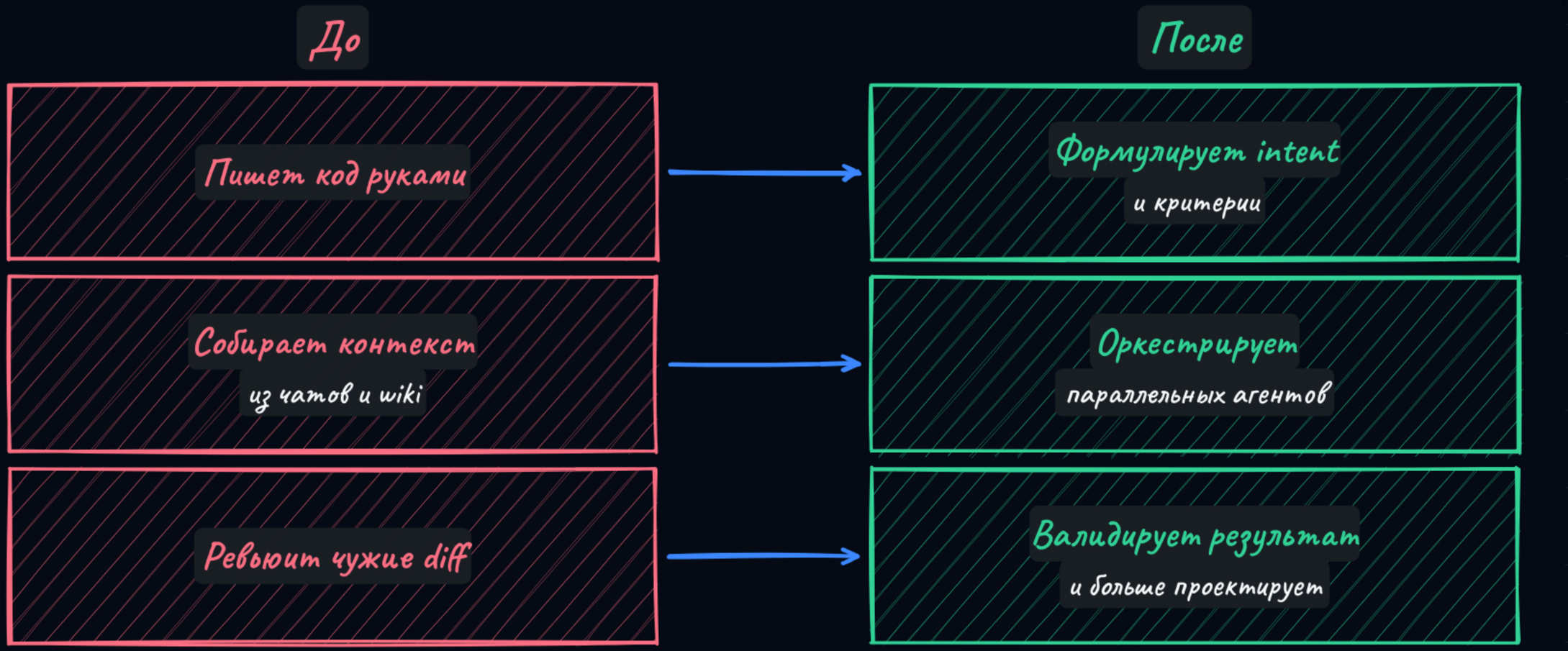
- Включили ассистента и ждали
- Задачи ставятся «по чату»
- Review и тесты без изменений

С эффектом

- Requirements стали agent contracts
- Review адаптирован к AI-изменениям
- Context и checks в pipeline

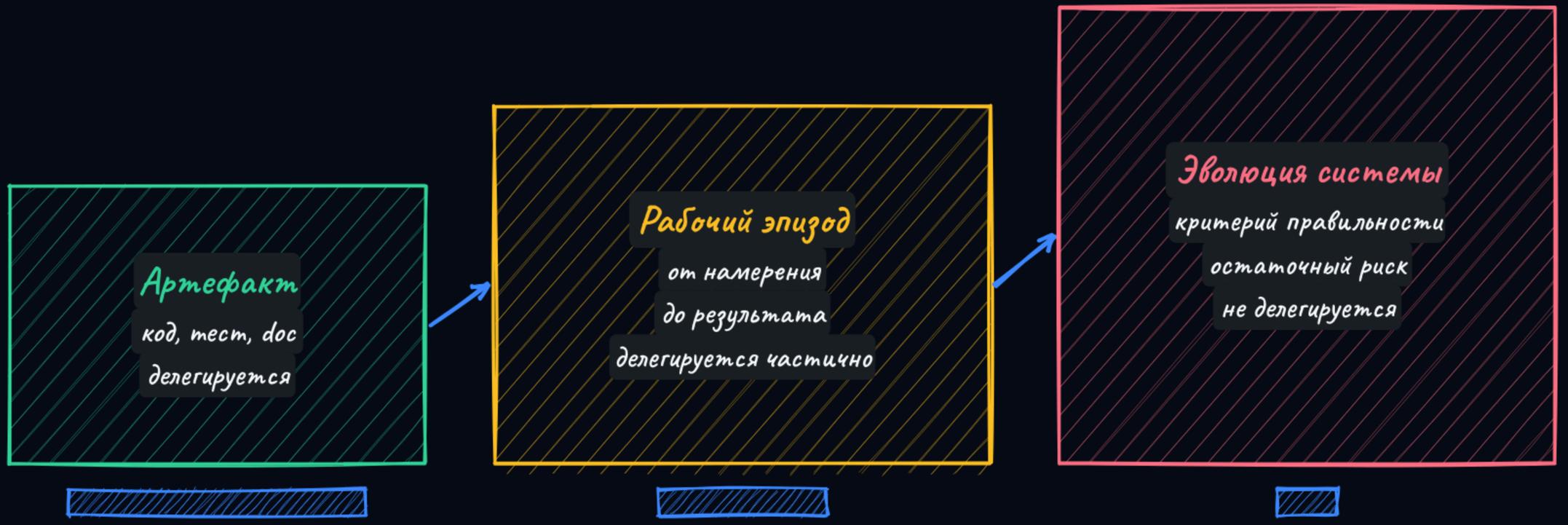
День инженера: до и после

Инженер

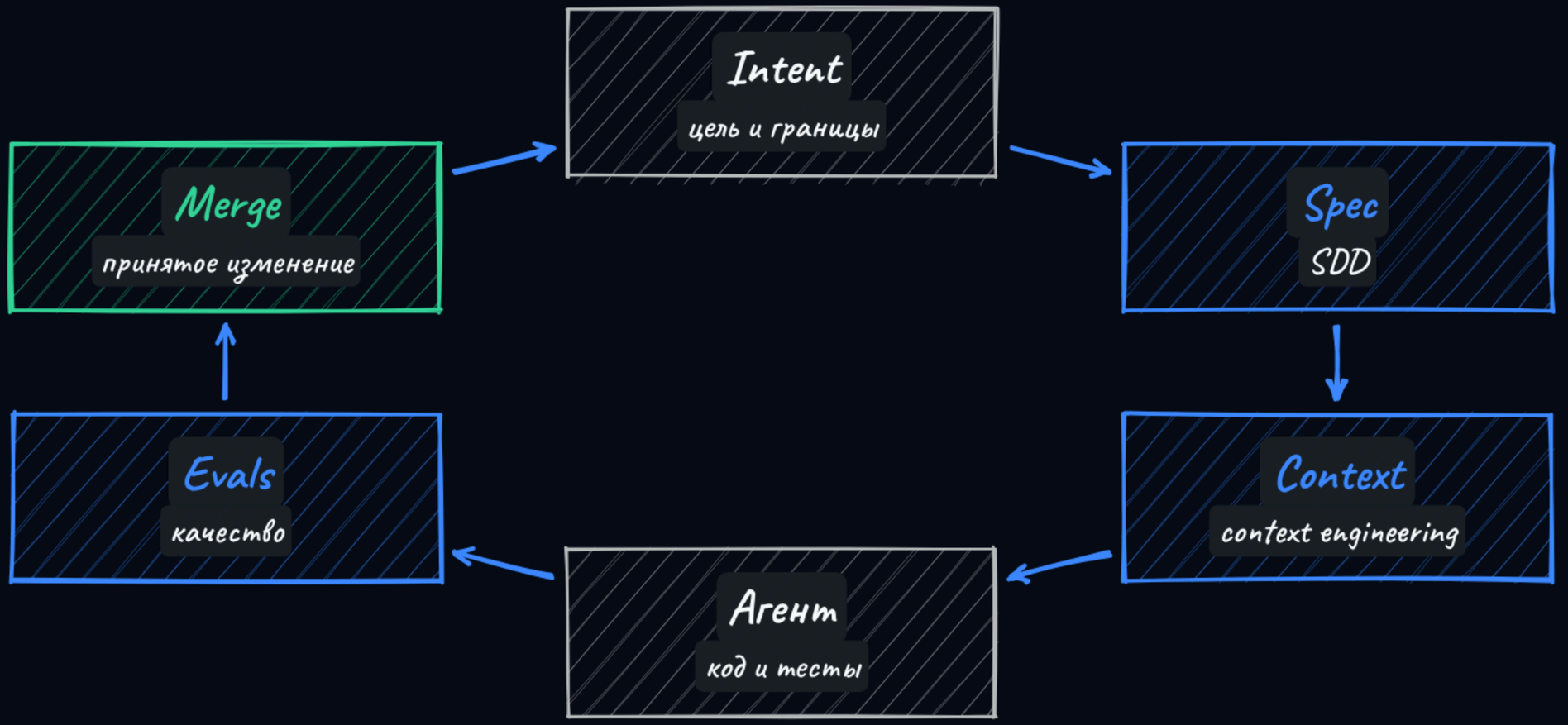


Что остаётся человеку

Инженер



Три навыка замыкают контур



Мифы, которых не стоит бояться

- Инженеры не исчезают: judgment важнее
- Джуны нужны, меняется обучение
- Ревью проверяет contracts и proof
- Ответственность остаётся на человеке

Пять сдвигов AI4SDLC

- Кодинг — больше не главное ограничение
- Очередь переехала в постановку и проверку
- Знание уходит из кода в примитивы
- Evals и provenance переживают реализацию
- Работа — у инженеров, а не у кодировщиков

- Код сгорит. Система возродится — по Чаду Фаулеру

AI4SDLC

Спасибо!

Слайды и источники — в Telegram-канале; книга Чада Фаулера «Regenerative Software» — на O'Reilly

Александр Поломодов

CTO & Technical Fellow,
polomodov.tech

@ai4sdlc