

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

Контекст

2024

Разрешать ли Copilot?

автодополнение

вопрос к безопасности

2025

Ассистент по умолчанию

чат и правки в IDE

в каждой команде

2026

Агент берёт тикет

draft PR, ревьюит человек

90 % с AI (JetBrains)

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

Индустрия

Внедрение

JetBrains 2026 · SO pulse 04.2026

используют AI-инструменты



специализированные dev-tools



используют агентов на работе



Доверие и контроль

SO pulse 04.2026 · Sonar 2026

редко включают автопилот



блокируют несогласованное



не доверяют AI-коду полностью



всегда проверяют перед коммитом



Код быстрее, поставка — нет

Индустрия

Инженер · DX Q2 2026

+37 %

PR на инженера в неделю

+64 %

размер PR

ревью
дольше

Поставка · Harness 2026

22 % vs 15 %

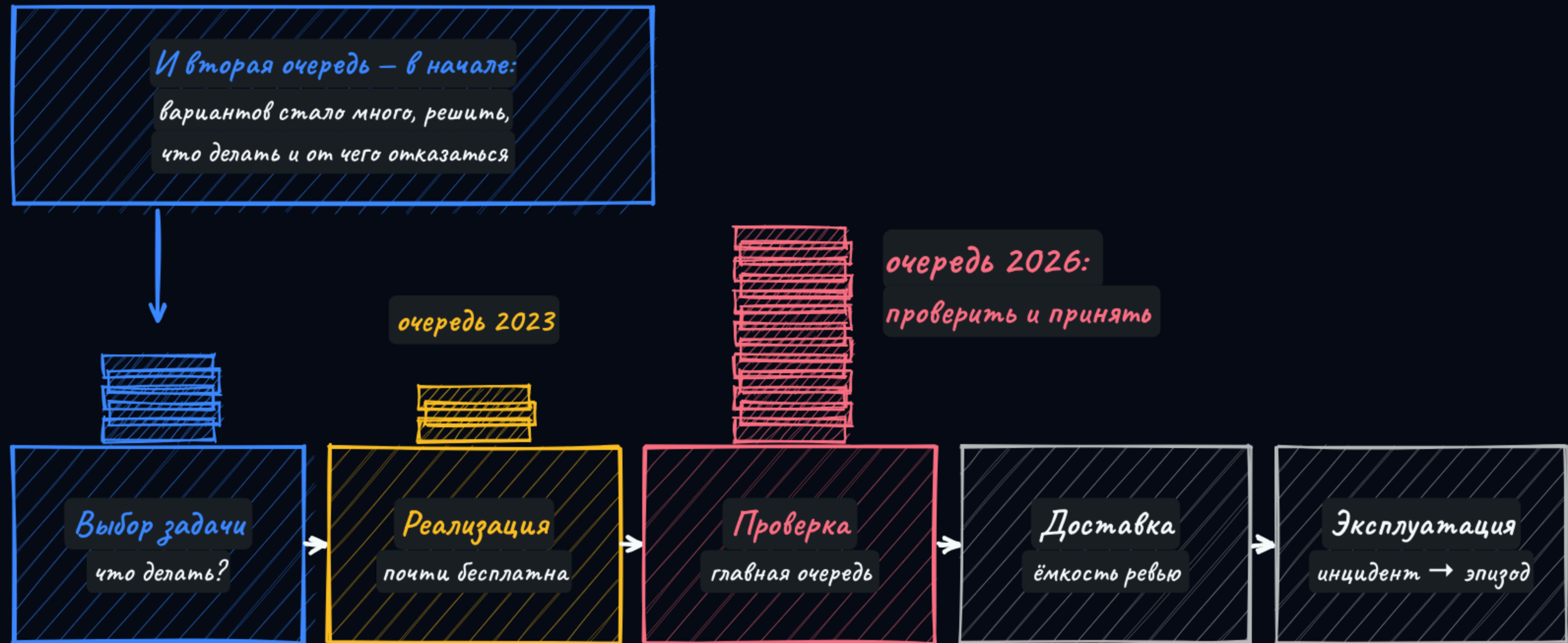
деплов с откатом или инцидентом

7,6 ч vs 6,3 ч

MTTR: частый AI vs редкий

Узкое место переезжает вправо

Сдвиг



Часть 2

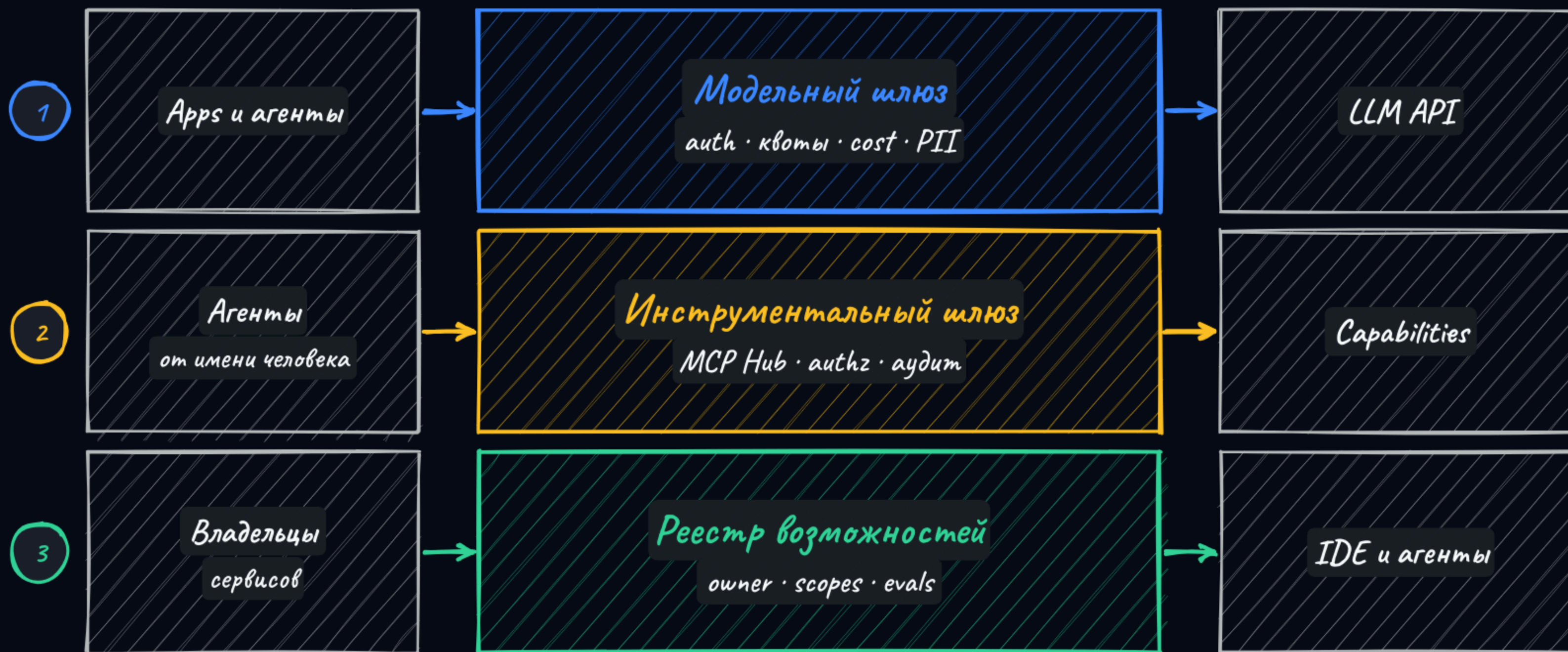
02

Как это выглядит в финтехе

10 000+ инженеров, тысячи сервисов, один периметр
контроля

Три слоя агентной платформы

Платформа



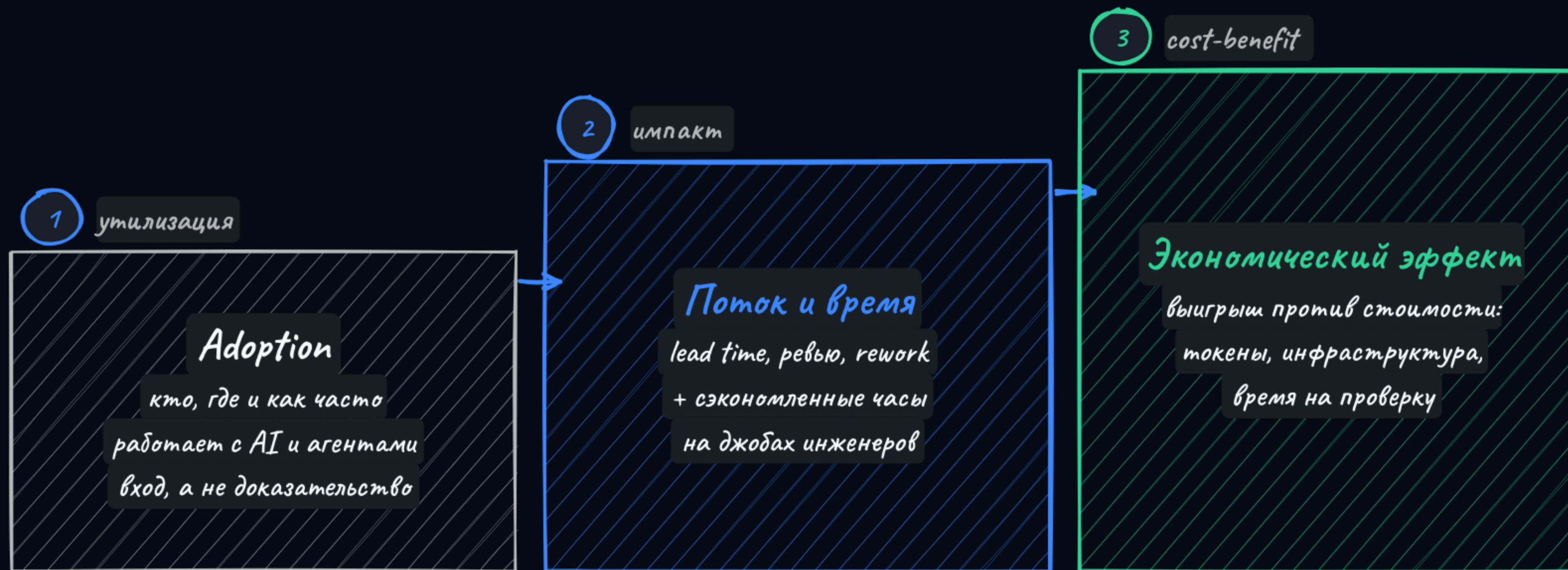
Агенты по всему SDLC

Агенты



Что считать вместо AI-LOC

Измерение



AI-LOC не отвечает ни на один из трёх вопросов

Часть 3

03

Regenerative Software

Код стал артефактом сборки. Что должно пережить
реализацию?

Код, которому суждено умереть

Regenerative Software



Тест на удаление

Regenerative Software

`rm -rf src/`

и регенерируем с нуля

Чем докажете, что результат верный?

«Старым кодом»

провал:

знание запуталось

в реализации

«Оракулами»

property-тесты, контракты,

инварианты, телеметрия,

явная корректность

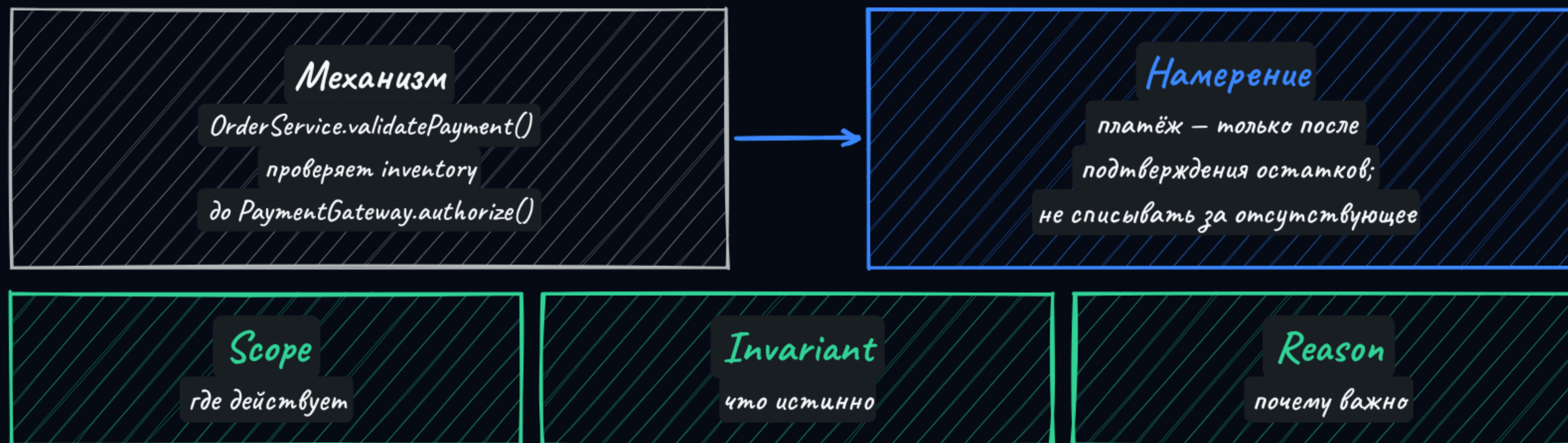
Семь неудаляемых примитивов

Regenerative Software



Intent: обещание, не механизм

Примитив 1 · Intent

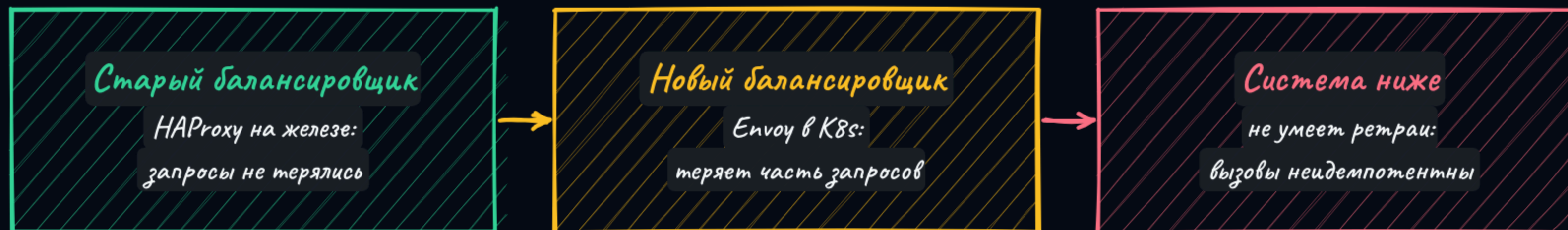


Негативное пространство: не списать дважды, не уйти в минус по остаткам

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

Ни единого разрыва

Примитив 1 • Intent



Что должен был сказать intent

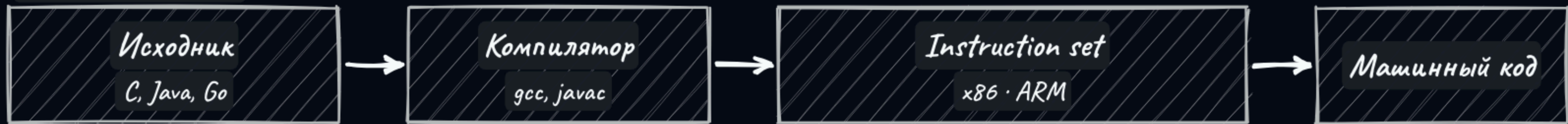


Выход: записать инвариант или сделать вызовы идемпотентными

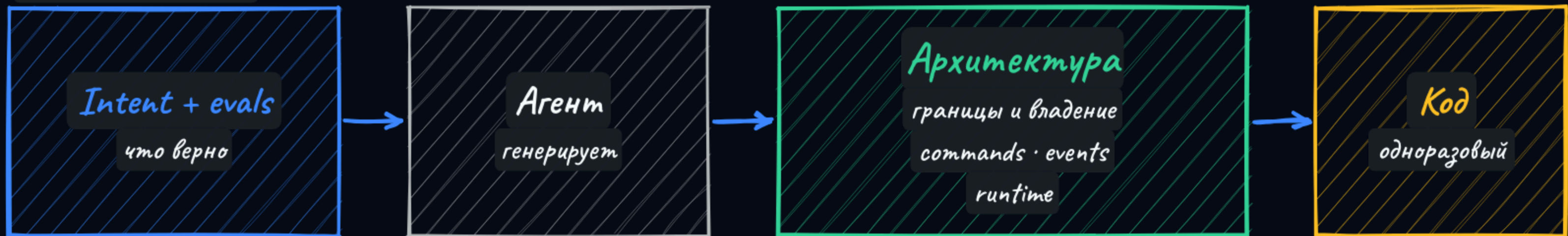
Архитектура — цель компиляции

Примитив 2 · Compilation

Как было: компилятор



Как стало: регенерация

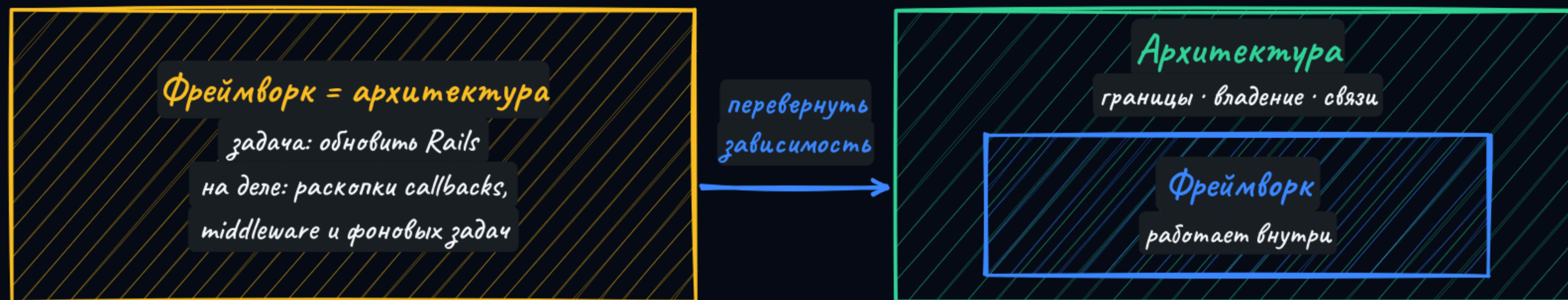


Без цели генератор копирует случайности старого кода или каждый раз строит новую систему

Знакомые практики bounded contexts · fitness functions · CQRS · golden paths · ADR

Фреймворк стал архитектурой

Примитив 2 · Compilation



Тогда замена — просто рекомпиляция



Intent и evals не меняются — меняется цель компиляции

Compilation: чёткие границы

Примитив 2 · Compilation



Простой API: объекты и связи, без SQL и обходов графа

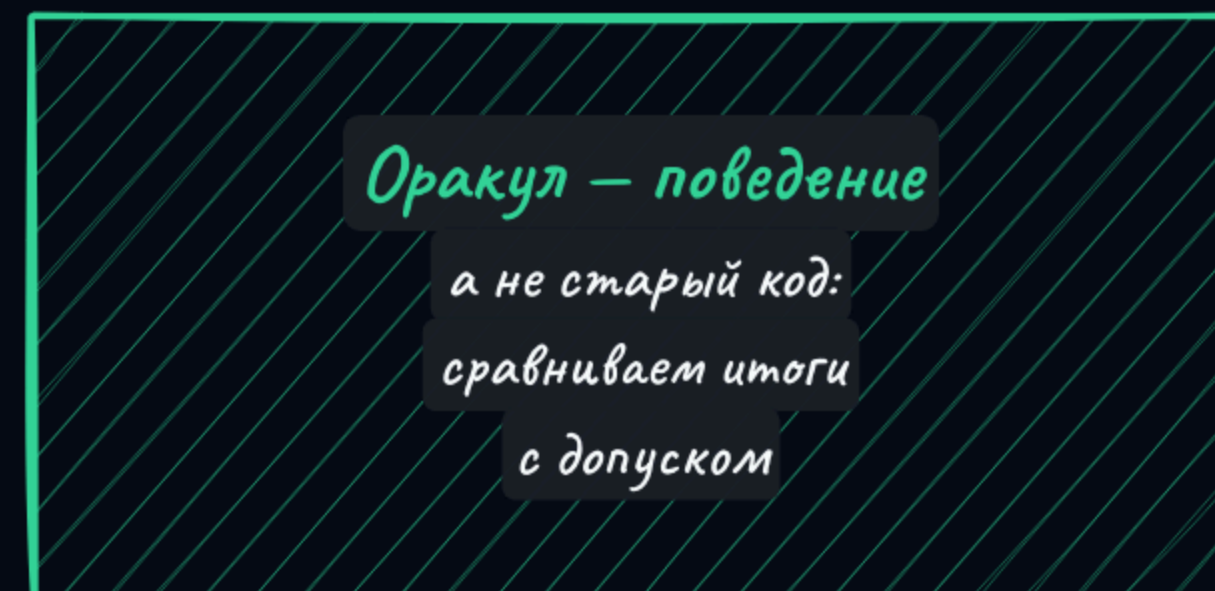
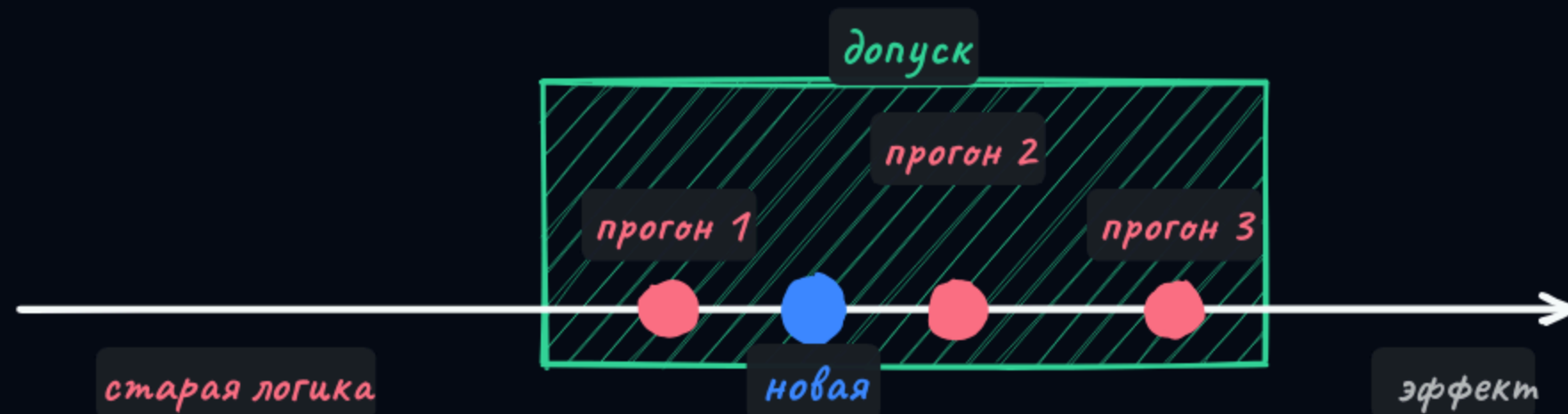
Узкий API сделал реализацию заменяемой: Rust уже обслуживает 95 % запросов

Проверяем поведение, а не код

Примитив 3 · Evaluations



Итог одного и того же эксперимента



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

Чем проверять поведение

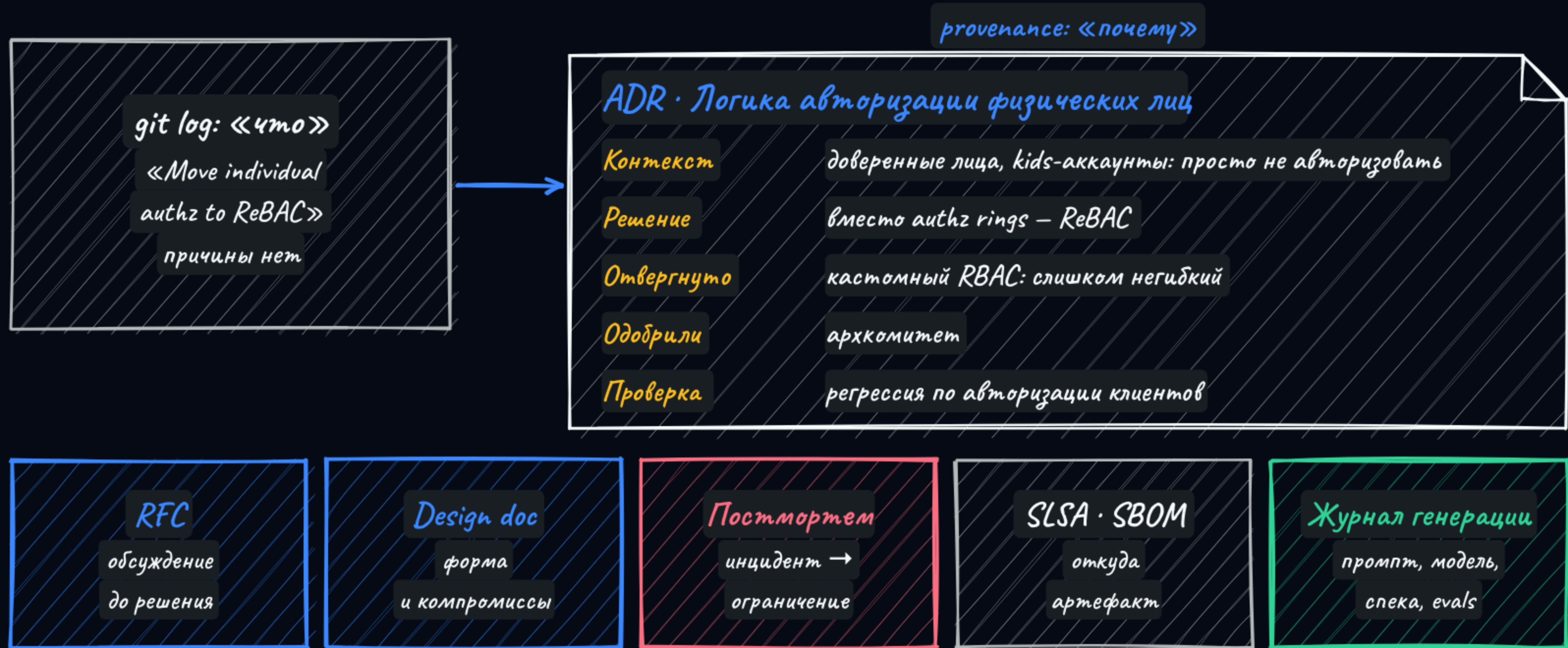
Примитив 3 · Evaluations

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	«р99 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

Примитив 4 • Provenance



Лог решений вместо памяти

Примитив 4 • Provenance

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»

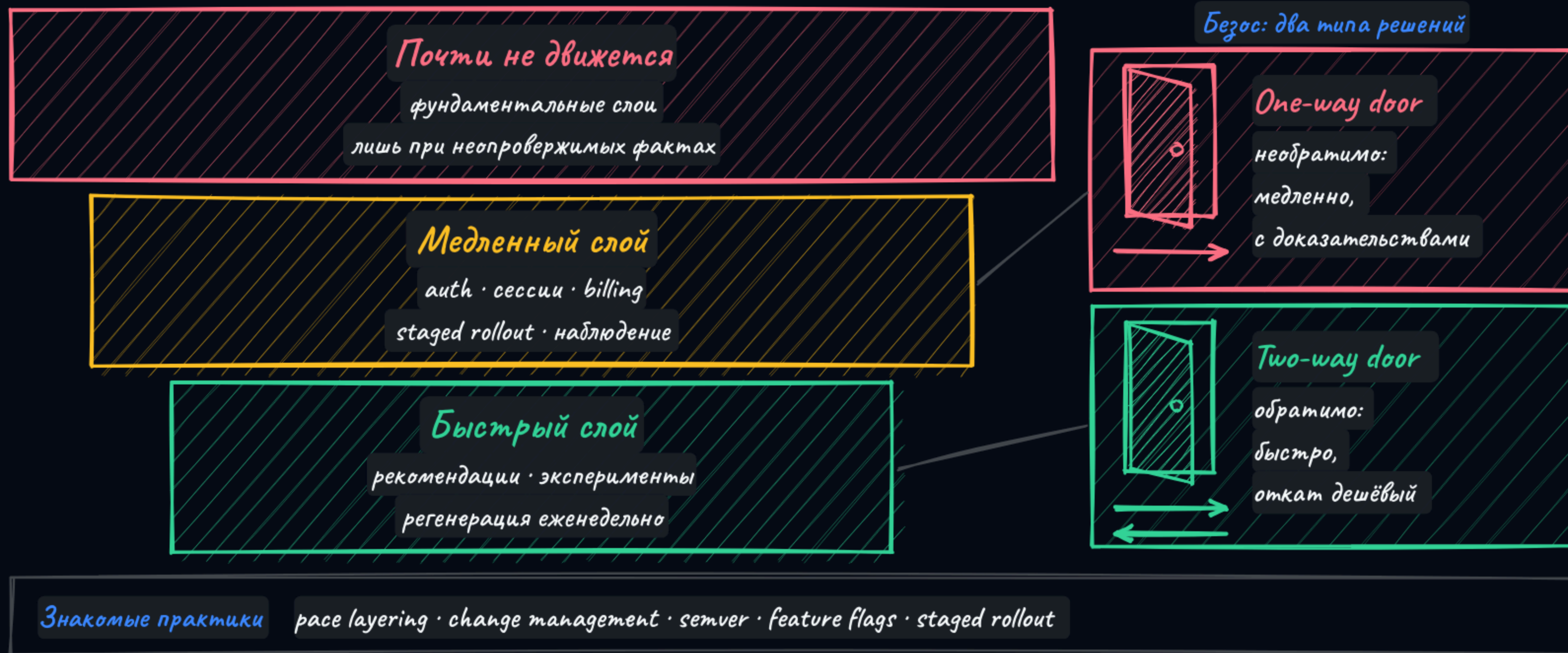
Без лога

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

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

Расе: у слоёв разная скорость

Примитив 5 · Расе



Aerospike: решает слой, а не diff

Примитив 5 · Race

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

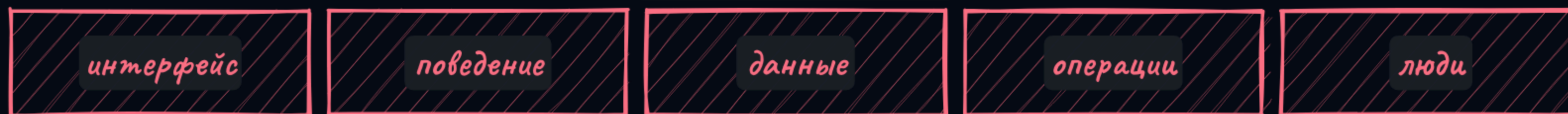
большой отказ при сбое одного ДЦ

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

Deletion: удалить безопасно

Примитив 6 · Deletion

Скрытые зависимости



Знакомые практики strangler fig · expand/contract · deprecation policy · трассировка трафика · scream test

2008: месяц, который не закрылся

Примитив 6 · Deletion



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

Contraction: пять уровней

Примитив 7 · Contraction

Сервис аутентификации после шести регенераций

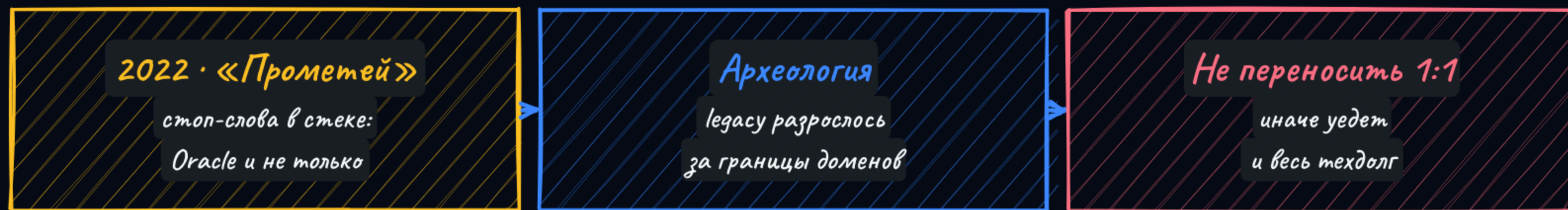
Уровень	Техдолг	После contraction
Интерфейсы	endpoint ради одного партнёра	→ одна версия API, старое — sunset
Конфигурация	флаг: «а что будет при false?»	→ флаг проверен и удалён
Зависимости	очередь: вдруг её читает nightly job?	→ потребитель найден — или очереди нет
Evaluations	тесты на внутренности старой версии	→ проверки поведения на границе
Понятийный	4 формата токенов, 3 потока логина	→ одна модель, которую можно объяснить

Сложность — это сколько нужно знать, чтобы безопасно менять

Знакомые практики · рефакторинг · tech-debt backlog · dead code removal · API sunset · когнитивная нагрузка

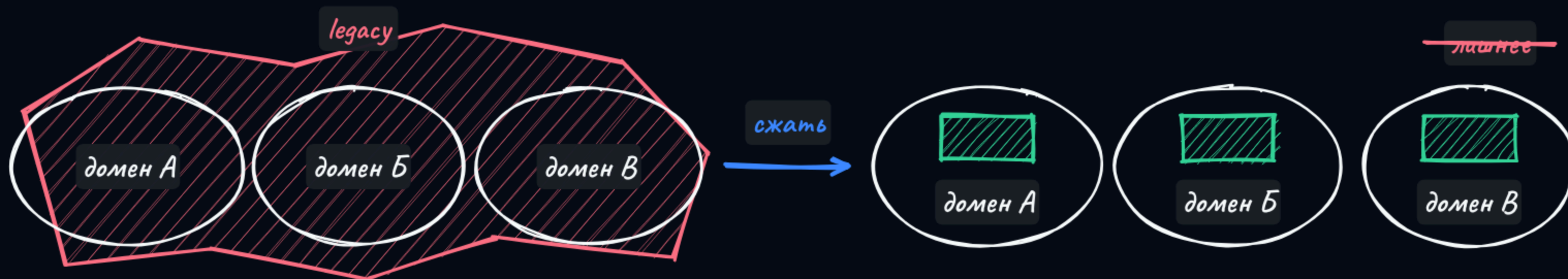
Compaction на потоке

Примитив 7 • Compaction



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

Regenerative Software

Intent

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

Compilation

bounded contexts · fitness functions · CQRS · golden paths

Evaluations

TDD/ATDD · property-based · contract tests · canary · chaos

Provenance

ADR · RFC · design docs · постмортемы · SLSA · lineage

Pace

pace layering · semver · feature flags · staged rollout

Deletion

strangler fig · expand/contract · deprecation · scream test

Compaction

рефакторинг · tech debt · dead code · API sunset

Практики нужны как никогда: работа будет у инженеров, а не у кодировщиков

Код помнит, организация — нет

Regenerative Software

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

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

Вероятно 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

До

Пишет код руками

Собирает контекст
из чатов и wiki

Ревьюит чужие diff

После

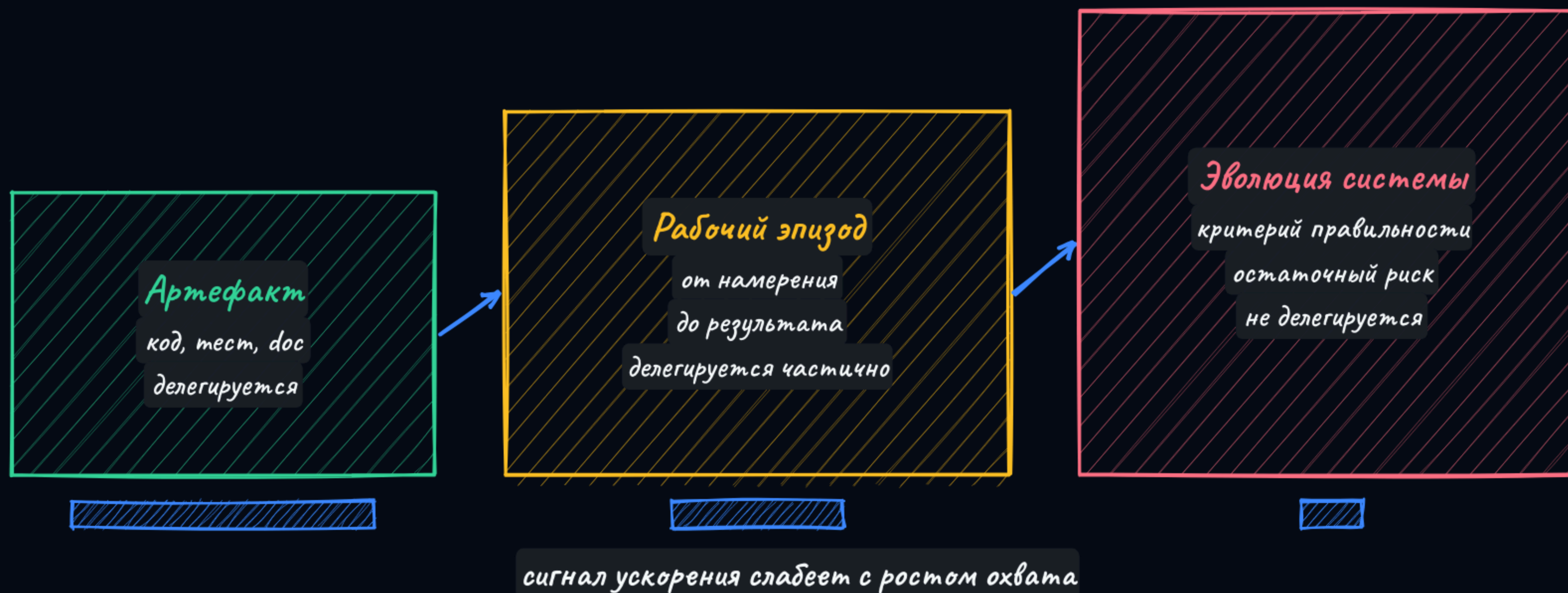
Формулирует intent
и критерии

Оркестрирует
параллельных агентов

Валидирует результат
и больше проектирует

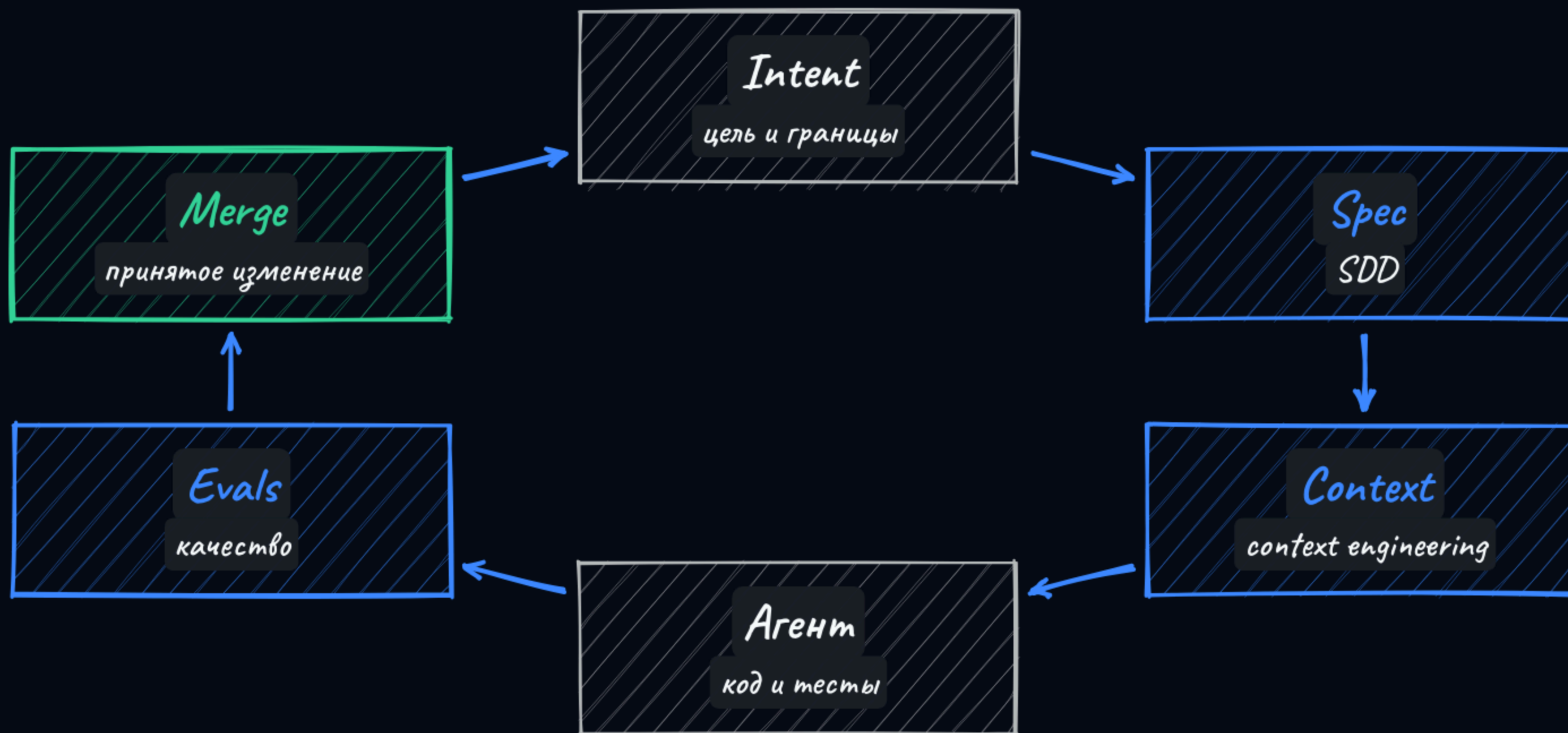
Что остаётся человеку

Инженер



Три навыка замыкают контур

Навыки



Мифы, которых не стоит бояться

Мифы

Чего не происходит

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

Пять сдвигов AI4SDLC

Итоги

Пять сдвигов

- 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 знает, как вернуться

AI4SDLC

Спасибо!

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

Александр Поломодов

CTO & Technical Fellow,
polomodov.tech



@ai4sdlc