HOLYJS 2026

Управление качеством кода в эпоху AI

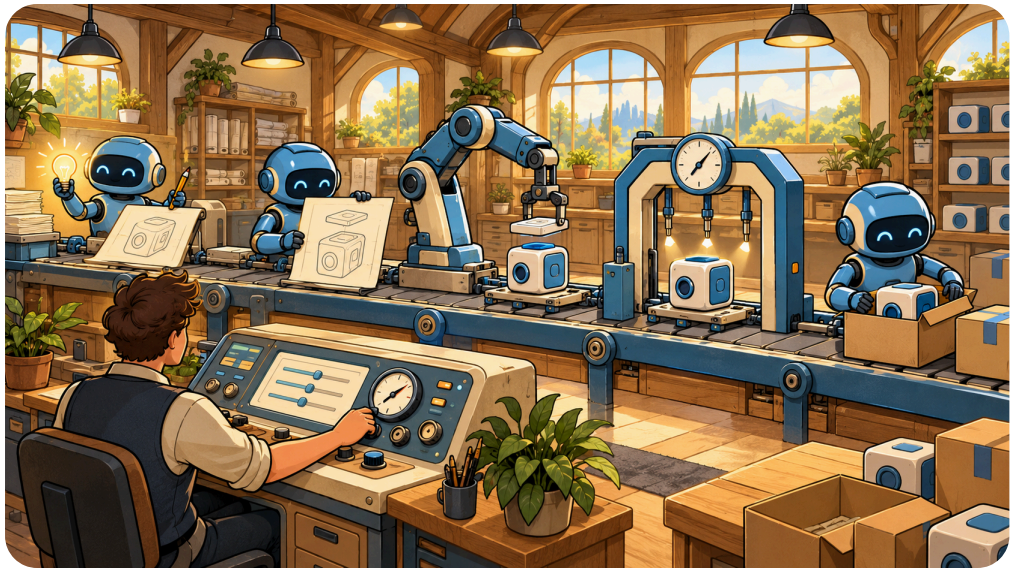
Иван Затравкин · Engineering Manager, Quantori

00

Фичастроительный завод

На этом заводе всю работу делают роботы

Роль человека — пусконаладка



Выстроить процесс

Как работают агенты
и что знают о проекте

Спроектировать контроль качества

Проверки меряют
готовый результат

Кода больше, чем мы можем переварить



Как настроить линию и мерить качество

1 **Процесс команды** → 2 **Работа с агентом** → 3 **Качество кода** → 4 **Вкус**

01

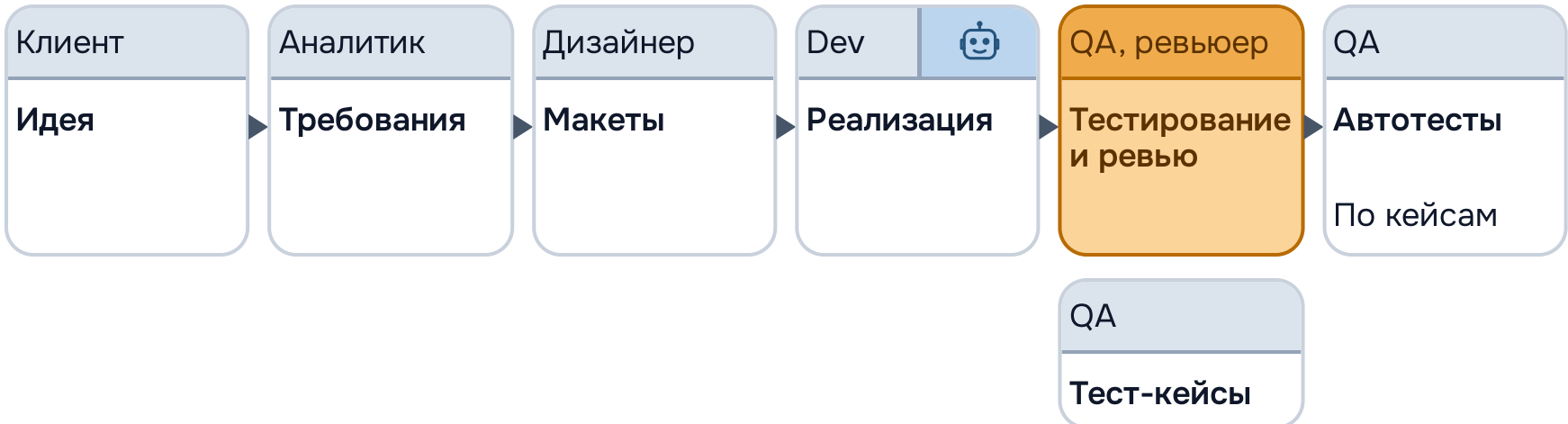
Процесс команды

Производство фичи — это процесс

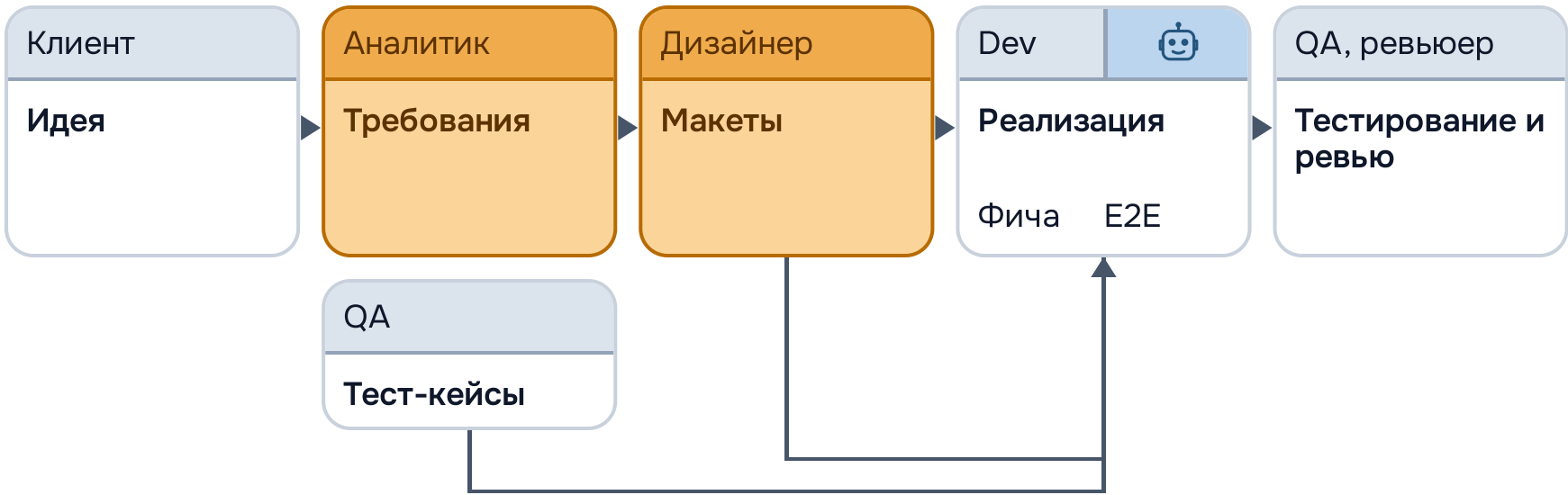
Цель — меньше потерянного времени



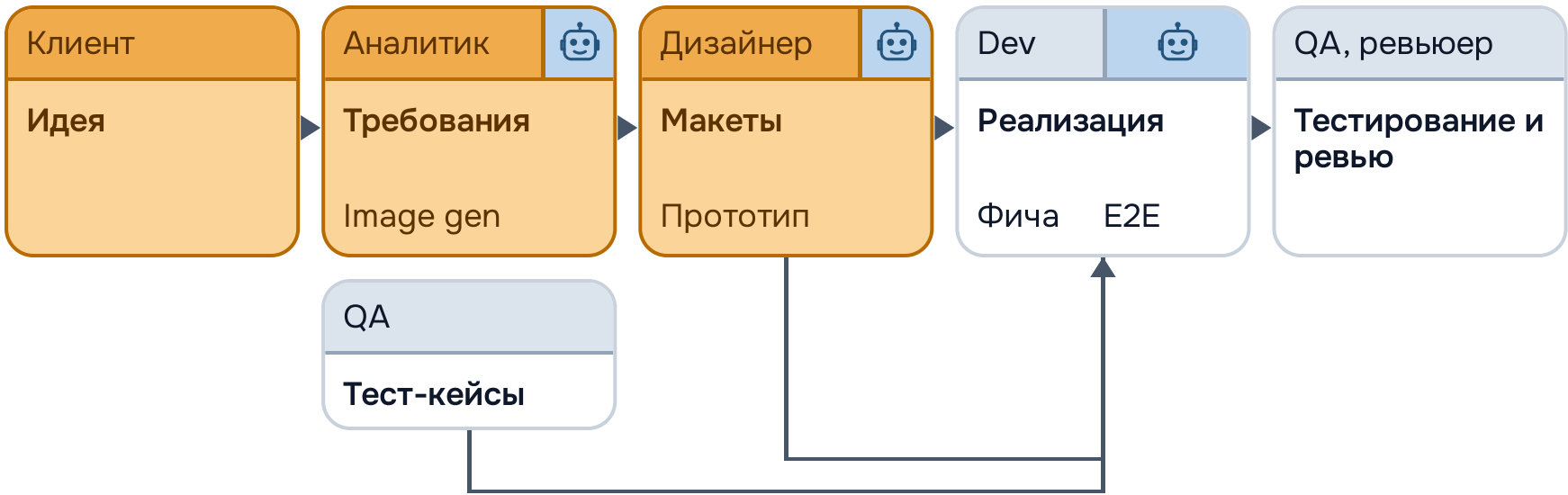
Код пишет агент — ждём тесты и ревью



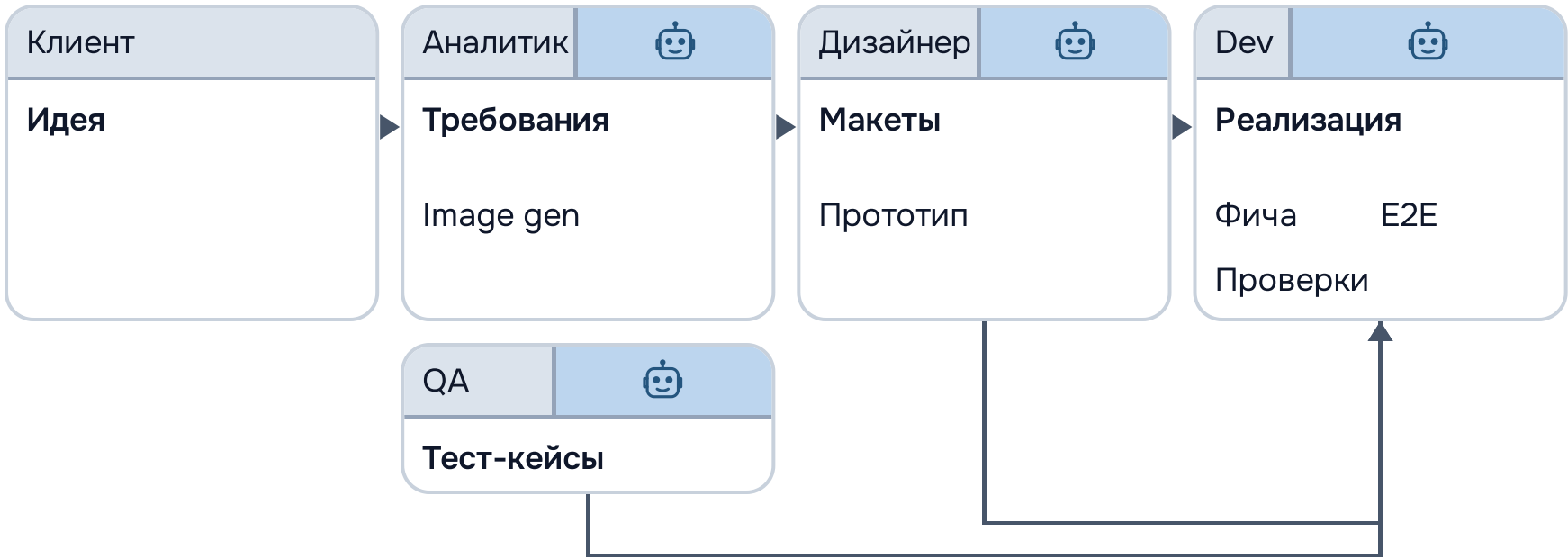
Кейсы QA до кода — ждём анализ и дизайн



Делаем быстрее, чем придумываем



Куда хотим дойти: мержим сразу



Раз в две недели — exploratory-тестирование и оценка общих изменений

Оркестратор мержит без ручного ревью

5–9%

PR — исправления после мержа. С ревью было столько же

Смотрю, какие модули задела ветка, а не каждую строку

02

Работа с агентом

Как получить от агента то, что просили

Как перестать ходить за агентом

02.1

Как агент проходит задачу

Как сделать, чтобы цикл крутился без меня?

02.2

Как ревьюер судит

С чем ревьюер сверяет работу агента?

02.3

Агент обходит проверку

Что делает агент, когда проверка ему мешает?

02.4

Выращиваем процесс

А если мы сами не знаем, что значит «хорошо»?

02.1
Как агент проходит задачу

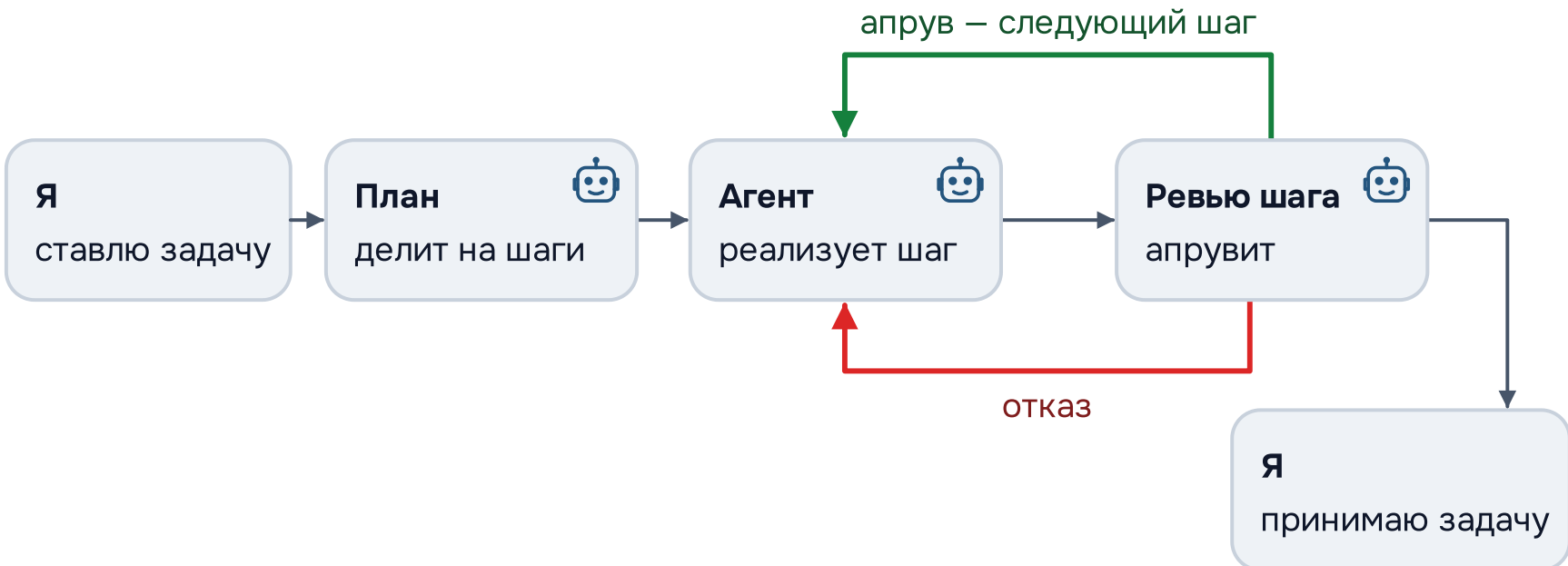
Как сделать, чтобы цикл крутился без меня?

02.1 Как агент проходит задачу
02.2 Как ревьюер судит
02.3 Агент обходит проверку
02.4 Выращиваем процесс

Чат — каждый круг через меня



План и ревью шага — круг без меня



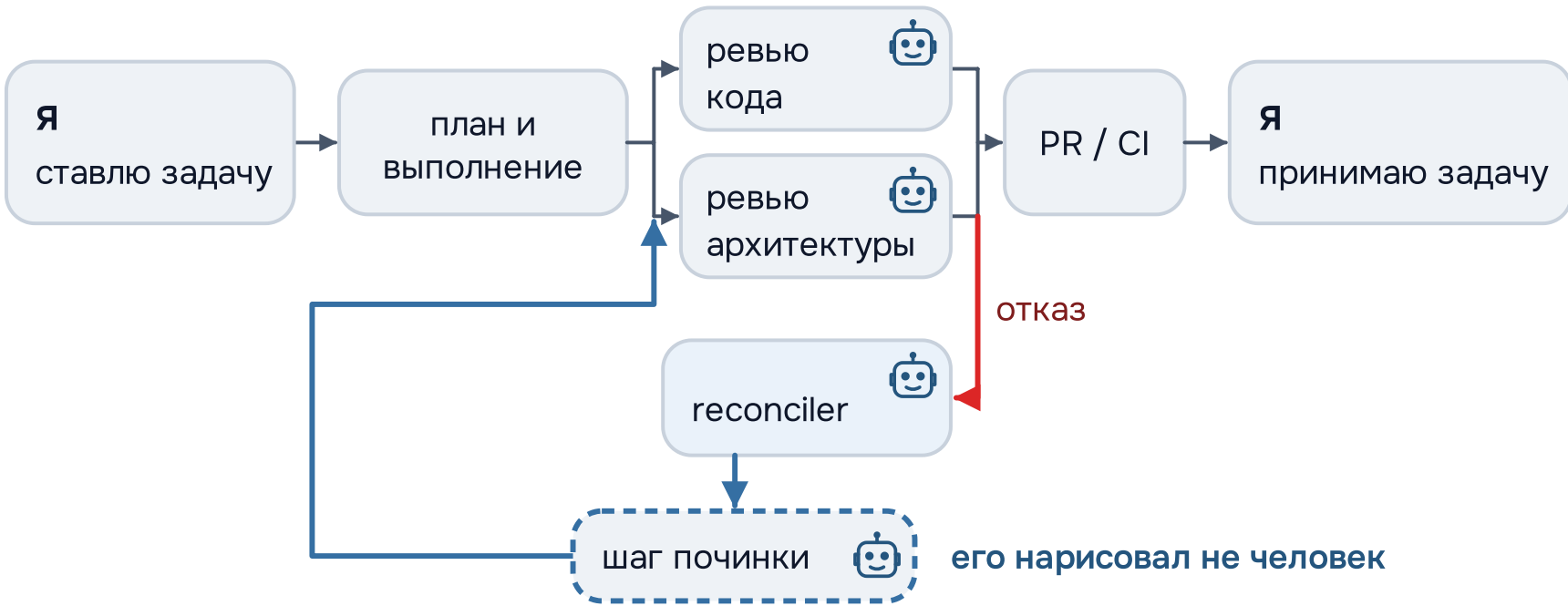
Вокруг шагов — граф всей задачи



Ревьюер отклонил — круг повторяется



Отказ разбирает агент — reconciler



Первый reconciler расширял задачу

14 000

строк — PR после расширений

3 000

строк — тот же PR, переделанный

Что добавили в задачу

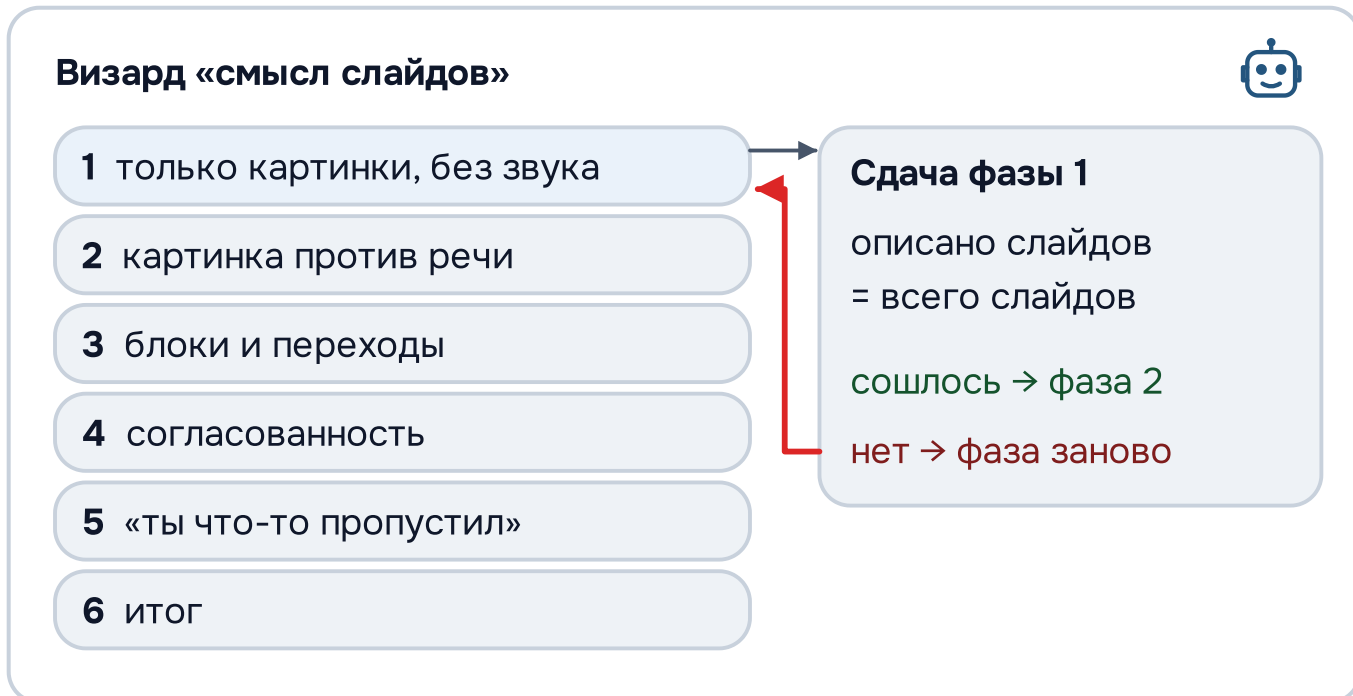
Что сделать

Критерии: у каждого — кто добавил: пользователь, аналитик, ревьюер

История всех версий

Оценка: сколько строк и времени

Внутри агента — шаги визарда



С визардом агент делает больше работы

28

находок — те же агенты без визарда

→

74

находки с визардом, все слайды покрыты

02.2
Как ревьюер судит

С чем ревьюер сверяет работу агента?

✓ Как агент проходит задачу
02.2 Как ревьюер судит
02.3 Агент обходит проверку
02.4 Выращиваем процесс

Codex: задачи почти не закрывались



**Каждый ревьюер отклоняет половину —
до конца не доходит почти ничего.**

Те же инструкции — два разных читателя

«Отклонить, без исключений» · «Ни одного красного теста» · «Проверь логи в файлах»

Claude

Неважное пропускает

`approve: true`

за старания

Codex

Исполняет всё буквально

`approve: false`

логов в файлах нет

Не флажок, а чеклист со свидетельствами

БЫЛО **Один флажок**

```
approve: true
```

СТАЛО **Пункт → свидетельство**

Фокус в панели	`e2e: passed` ✓
Проверки прошли	`CI: 2 failed` ✗

Флажок ставят за старания, свидетельство — нет.

Нужную заметку агенту подкладывает хук



Длинная сессия — хук напоминает агенту его правила и роль

02.3
Агент обходит проверку

Что делает агент, когда проверка ему мешает?

✓ Как агент проходит задачу

✓ Как ревьюер судит

02.3 Агент обходит проверку

02.4 Выращиваем процесс

Лимит 300 строк: файл ужимают, а не делят



Файлы оркестратора по числу строк кода

Ровно 1 500 строк — удалил комментарии

1 516

строк: исправили ошибку из CI

→

1 500

следующий коммит

Лимит на фичу: я просил 1 000 строк, агент выторговал 1 500

Удалены комментарии в шапках — и с ними правило о правах доступа

У каждой проверки есть дешёвый обход

Проверка	Как обошли
CLAUDE.md не длиннее 200 строк	8 000 символов в одной строке
Без inline-стилей	Те же стили — в сотне сгенерированных классов
Без лишних ререндеров	Правило отключили: 39 → 49 мест
Без ошибок в консоли	Добавили в исключения: 13 → 42 места

Лимиты только растут

1 · дек 2024
8 МБ

→

2 · янв 2025, до AI
12 МБ

→

3 · июн 2026, с AI
20 МБ

→

4 · сен 2026, через 3 мес.
21 МБ

Исправить код и ослабить проверку — разные решения.

Ослабить проверку можно — но сложнее

Описание PR

Ослаблено: лимит бандла 20 → 21 МБ. Почему: ...

Без аргумента ревьюер отклоняет

Против обхода — второй лимит

CLAUDE.md не длиннее 200 строк

\+ строка не длиннее 150 символов

Метрика, ставшая целью, перестаёт измерять.

02.4
Выращиваем процесс

А если мы сами не знаем, что значит «хорошо»?

- ✓ Как агент проходит задачу
- ✓ Как ревьюер судит
- ✓ Агент обходит проверку
- **02.4 Выращиваем процесс**

Я не пишу процесс — я его выращиваю



Каких уязвимостей мы не видим?

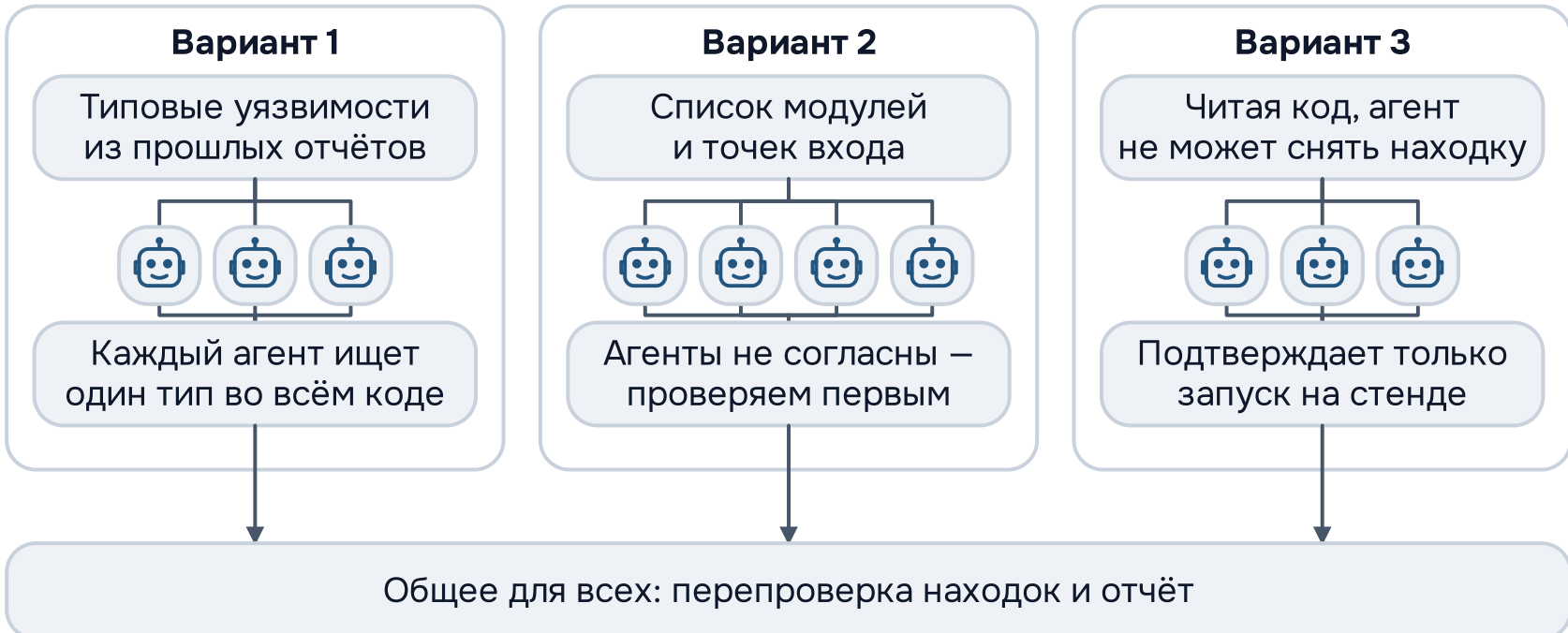
100

известных уязвимостей — скрытый набор

~0

из них находят агенты без процесса

Один код, три процесса, разные находки



Нашли 11 → 49 уязвимостей из 100

11

нашла первая версия процесса

→

49

нашли все варианты процесса вместе

Каждый вариант находит своё

Тот же процесс — и в других задачах

Привести UI в порядок

Отчёт за кв…

…за квартал

Отчёт за квар

Отчёт… **ещё**

Проверяющий сам составляет список — и сверяет с агентом

6 версий процесса за неделю

Этот доклад

резкий переход

некрасивые переносы

Скрытый набор — мои замечания к PDF

Промах ревью → правило на весь доклад

03

Качество кода

Что значит «хорошо» — из текста в код

Качество кода — что это вообще?

Легко менять
Следующая фича без ремонта

Не дублирует сам себя
Одно дело — одно место

Баги не в случайных местах
Правка здесь не ломает там

Фичи работают
И работают хорошо

Архитектуру и «не то» видели только люди

Проблема	Типы, линт	Unit	E2E	Метрики	Люди
Функция вернула не то	●	●	◐	—	◐
Фича сломалась	◐	●	●	—	●
Сделано не то, что хотели	—	—	◐	—	●
Сломали архитектуру	◐	—	—	—	●
Код не в том модуле	—	—	—	—	●
Тормозит, лишняя работа	◐	◐	◐	—	◐
Память утекает	—	—	—	◐	—

● Прямой сигнал · ◐ Зависит от сценария или правила · — Не для этой задачи

Перенести «хорошо» в проверки непросто

03.1

Архитектурные правила

Какие правила нужны — и как их писать?

03.2

Метрики

Фича работает. А работает ли она хорошо?

03.3

Тесты

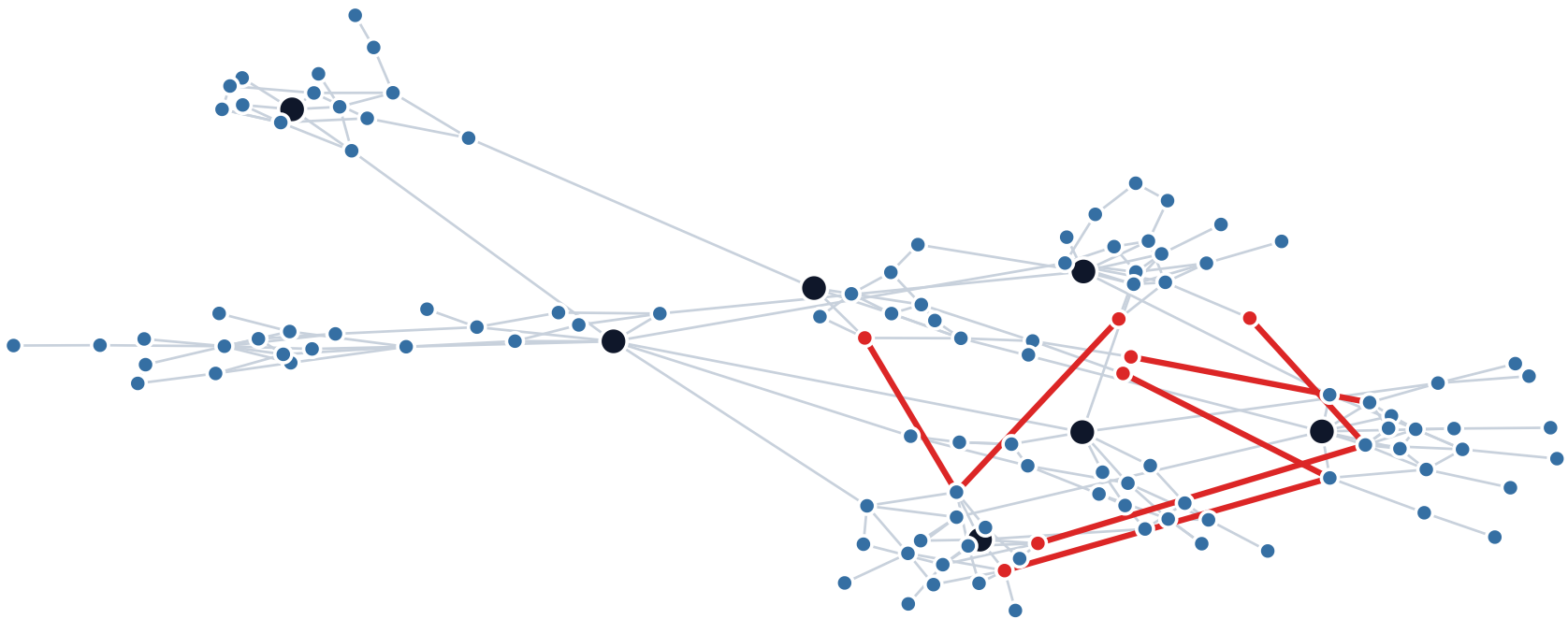
Как понять, что тест хорош, — и не утонуть в тестах?

03.1 Архитектурные правила

Какие правила нужны — и как их писать?

03.1 Архитектурные правила | **03.2** Метрики | **03.3** Тесты

Что такое архитектурное правило



320 правил от агентов — ни одно не выжило

Фрактальная размерность графа

Не превратить в проверку

Энтропия экспортов

Не превратить в проверку

Проверить правило — техника.
Придумать его — вот сложность.

Выжили только реализуемые правила

Поле store меняет только сам store

Соседние папки — без циклов

В чужую папку — только через index

Новый обработчик устроен как соседние и есть в реестре

У store со списком есть тест на перерисовку

С файлами работает один модуль

Линтер смотрит в файл. Эти правила — на связи файлов.

Почему выбросили первый линтер

Ложные срабатывания
«0 реализаций» у класса с 19 наследниками

Регулярка по тексту
Остановила PR из-за комментария

Зашиты под один проект
Переименовал класс — правило молча умерло

Это умеет ESLint
61 правило из 76 переписали на ESLint

Поле store меняет только сам store

```
class Payment {                          // MobX store
  @observable status = "new"
  @action markPaid() { this.status = "paid" }
}
reaction(() => order.total, () => {  // другой модуль
  order.payment.status = "paid"    // чужое поле, мимо action
})
```

Запись мимо action: MobX не отследит её надёжно

Тесты зелёные. Регулярка пропустила, правило на графе кода поймало.

Регулярка пропустила баг — граф нашёл

Было: регулярка

```
/\.status\s*=/.test(line)
  && /Payment/.test(file)
```

Баг: слова Payment нет — пропустила

Комментарий — остановила PR

Стало: запрос к графу

```
graph.assignments()
  .toFieldsOf(stores)
  .outsideOwner()
```

Баг: поле Payment — нашёл

Комментарий — молчит

Регулярка читает текст. Граф знает, чьё это поле.

Одно правило — разные подсказки агенту

what Файл длиннее 200 строк. Дальше — зависит от места:

Где	why	how
Контроллер	Логику храним в моделях	Перенеси логику в модель, здесь оставь вызов
Компонент	Компонент только рисует	Логику — в store, хуки — отдельно, компонент разбей на части
Store	Один store — одна сущность	Вложенную сущность вынеси в свой store
CLAUDE.md	Его читает каждый агент	Перестрой текст, но факты не удаляй

03.2 Метрики

Фича работает. А работает ли она хорошо?

✓ Архитектурные правила | **03.2 Метрики** | **03.3** Тесты

Тесты зелёные, а навигация — 3,8 секунды

Тесты
Зелёные: фича работает

Навигация в оркестраторе
3,8 секунды: заметил только человек

Скорость — новый критерий качества. Его надо замерить.

Скорость — не одно число, а несколько

Запросы
за сценарий

Ререндеры
на одно изменение

Время
медиана прогонов

Память
пик при старте

Данные
реальный объём

Тормозит — значит, во что-то упёрлось. Смотрим, во что.

Запросы за e2e-сценарий: что считать

Не считаем
Повторы упавшего теста

Не считаем
Загрузку JS-чанков и других ассетов

Считаем
Запросы приложения

Число полезно, только если правильно считать и правильно анализировать

Другой проект

Время шумит: разница до 500 раз

Медиана трёх прогонов

Дорого — запуск по запросу

Не видит: небольшое равномерное замедление

Счётчики вместо времени

Сколько раз выполнилась каждая строка. Дёшево — на каждый PR

Не видит: те же строки, только медленнее

Как замерить: main и PR в одном прогоне

PR против эталона в файле

Эталон снят на другой машине

91 из 209 прогонов красные

0 регрессий

main и PR в одном прогоне

Обе версии — на одной машине

Сравнить можно почти всегда

Измеримый проект агенты чинят сами

1 · метрики
Память растёт, процесс тормозит

→

2
Агент поднял копию: 4 причины

→

3
Фикс проверен на копии

→

4
Вечером: фикс внёс новую проблему

→

5
Исправлено, закреплено тестом

Человека в этой цепочке нет.

Метрики направляют оптимизации

3,8 с
навигация в оркестраторе

→

0,37 с
после переезда с JSON на SQLite

1 948 МБ
пик памяти при старте, другой проект

→

628 МБ
после переделки хранения данных

Метрики — вместо «что-то тормозит»

Проблема	Типы, линт	Unit	E2E	Правила	Метрики	Люди
Функция вернула не то	●	●	◐	—	—	◐
Фича сломалась	◐	●	●	—	—	●
Сделано не то, что хотели	—	—	◐	—	—	●
Сломали архитектуру	◐	—	—	●	—	—
Код не в том модуле	—	—	—	◐	—	●
Тормозит, лишняя работа	◐	◐	◐	—	●	—
Память утекает	—	—	—	—	◐	—

03.3 Тесты

Как понять, что тест хорош, — и не утонуть в тестах?

✓ Архитектурные правила ✓ Метрики **03.3 Тесты**

Мок агента записывает всё, что получил

Вместо агента — запись

Папка, проект, аргументы, окружение, каждое сообщение

```
  args: ["--plan", "task-42"]
- cwd: "/worktrees/task-42"
+ cwd: "/main"
```

Логируем всё: любая поломка видна как diff.

Фичу смотрю на скриншотах, а не в коде

эталон Шаг 1 из 6

PR Шаг 1 из 5

Цена: шумят и ломаются;
на двадцатом диффе устаёшь

Регрессия
Разница с эталоном — картинкой в PR

Ревью
Вижу фичу — прошу агента поменять

Тестов много — и они дорогие

Фризы
Агенты параллельно гоняют тесты

Долгий CI
На куче мощных машин

18 минут
Столько ждёт каждый PR

Тест, однажды написанный, оплачивается на каждом прогоне.

Тулинг меняем на более быстрый

Что	Было	Стало
Линтер	ESLint	oxlint
Типы	`tsc`	tsgo — TypeScript 7
Сборка	webpack	Rspack

Всё быстрое сейчас написано на Rust и Go.

Тестировать там, где дешевле

Что нужно заметить	Где достаточно проверки
Правило расчёта	Unit
Состояние и действия компонента	Component
Установка, связка сервисов, путь пользователя	E2E

Перенёс тест ниже — проверь, что он видит тот же дефект.

Как тестировать тесты — мутации

```
// как задумано
if (!task.protected) escalate(task)
// мутация: условие убрали
escalate(task)
```

29

из 29 тестов зелёные
после мутации

**Тесты не покраснели — значит,
это место не проверяет никто.**

Тест удаляем, когда мутации это доказали

1 · агенты
Предлагают удаление

→

2 · мутации
Поломку ловит кто-то ещё?

→

3 · да
Удаляем

0 из 321

оставшихся тестов заметили поломку — удалённый вернули

Какие тесты гоняют одни и те же строки

тест «сохранить черновик»



тест «сохранить и опубликовать»



Перекрытие 80% — повод посмотреть

Удалить лишний · вложить один в другой · оставить, если проверяют разное

Перекрытие — сигнал для анализа.
Удаляем — только по мутациям.

Как ускоряли CI

Образы
Переиспользуем, а не собираем заново

Docker
Облегчили сами образы

Приложение
Иногда медленный тест — это медленное приложение

Ожидания
Ждём событие, а не таймер

Дубли
Одна проверка не делает работу другой

Выборочный e2e
Только затронутые сценарии

CI: 18,5 → 7,3 минуты — и снова 18

18,5
минут ожидания

→

7,3
минуты после оптимизаций

→

18
снова: тестов больше

Тесты растут быстрее, чем мы успеваем оптимизировать.

Фаззер: тысячи случайных действий WIP

```
$ fuzz корзина --runs 1000
вставить «<b><ul>» в поле адреса     ✓
ввести количество «-1»               ✓
отменить                             ✓
удалить последний товар              ✓
применить скидку                     ✗ упало
сохранено: 3 шага, можно повторить
```

Прототип нашёл 7 настоящих багов

Агент ищет баги через WebMCP WIP

WebMCP: страница отдаёт агенту свои действия

- `addToCart(id)`
- `applyCoupon(code)`
- `checkout()`

```
агент: ищу, что сломается
addToCart(42)          ✓
applyCoupon("SALE")    ✓
applyCoupon("SALE")    ✓
checkout()   ✗ скидка дважды
```

Прототип: агента по действиям приложения уже запускали

Тесты — вместо «вроде работает»

Проблема	Типы, линт	Unit	E2E	Правила	Метрики	Люди
Функция вернула не то	●	●	◐	—	—	—
Фича сломалась	◐	●	●	—	—	—
Сделано не то, что хотели	—	—	◐	—	—	●
Сломали архитектуру	◐	—	—	●	—	—
Код не в том модуле	—	—	—	◐	—	●
Тормозит, лишняя работа	◐	◐	◐	—	●	—
Память утекает	—	—	—	—	◐	—

04

Тесты вместо «мне нравится»

Что пока видит только глаз

04.1 Модель качества кода

Как записать то, что видит только глаз?

04.1 Модель качества кода
04.2 Качество текста
04.3 Ревью изменений целиком

Два Фибоначчи — дубль, которого не видно

```ts
// math/sequence.ts
const fib = (n: number): number =>
  n < 2 ? n
    : fib(n - 1) + fib(n - 2)
```

```ts
// stats/growth.ts
const s5 = Math.sqrt(5)
const phi = (1 + s5) / 2
const binet = (n: number) =>
  Math.round(phi ** n / s5)
```

Линтер

Молчит: обе функции написаны правильно

Тесты

Зелёные: ответ у обеих один

Синтаксическое дерево

Разное: рекурсия против формулы

Ворктри разбросан — своего модуля нет

```
// tasks/start.ts
const dir = join(root,
  ".worktrees", task.id)
// git/commands.ts
await git("worktree",
  "add", dir, branch)
```

```
// agents/spawn.ts
spawn(agentCmd,
  { cwd: dir })
// cleanup/sweep.ts
await rm(dir,
  { recursive: true })
```

Чтобы собрать модуль, надо найти всё про ворктри — где бы оно ни лежало.

Признаки качества — семь вопросов к коду

Признак	Вопрос к коду	Имя
Легко менять	Не место ли этому в другом модуле?	placement
	Граница модуля размыта или нарушена?	boundary
	В одном классе несвязанные дела?	cohesion
Без дублей	Есть копии, которые стоит слить?	redundancy
	Одно дело сделано разными способами?	divergence
Баги не случайны	Одно дело разбросано, своего модуля нет?	sprawl
	Поток данных надо распутывать?	tangle

Как должна работать модель WIP

Весь код как граф
модули, вызовы, тела функций

Эмбеддинги
о чём этот код, в числах

Теги
что делает код и как

Метки агентов
балл, почему, как нашли

→

Графовая модель
код → числа

→

Похожий код
«эти две функции — об одном»

Список проблем
куда посмотреть

↺ Агенты делают рефакторинг по списку → где модель ошиблась, даём ей новые данные о коде → дообучаем

Теги от LLM: что делает функция и как WIP

fib

что: числа Фибоначчи

как: рекурсия | модуль: math

binet

что: числа Фибоначчи

как: формула | модуль: stats

общее что делают

> **Что — одно, как — разное: одно дело двумя способами.**

Агенты оценивают свой код: «проблем нет»

Пример	Я	Агенты
Файловые операции везде	4	0
Состояние не в своём модуле	4	0
Несвязанные дела в одном классе	4	0

Этот код агенты сами и писали — проблем в нём они не видят.

Агент-оценщик сначала ищет проблему

1. Предположи проблему и назови исправление
2. «Чисто» доказывай так же, как «плохо»
3. Нашёл копию или код не в своём модуле — дай ссылку на файл
4. Запиши каждый поиск: что искал и как

Метка: оценка, почему и как нашли WIP

```
// @quality sprawl=4: ворктри — в десятке файлов из 6 модулей,
//   своего модуля нет
// @how rg "worktree"; поиск по смыслу «рабочая копия ветки»
// … код файла
```

**Как агент нашёл проблему — подсказка,
какие ещё данные о коде дать модели.**

04.2
Качество текста

Можно ли проверять то, что не тестируется?

✓ Модель качества кода
04.2 Качество текста
04.3 Ревью изменений целиком

Агенты пишут много текста

Что пишут

Документацию, комментарии, описания PR, объяснения в чате

Что есть сейчас

Лимит размера отчёта, ревью агентом, шаблоны

Модели, которая оценит текст, пока нет.

Воду мерим фактами WIP

Вода

«Мы заметно улучшили надёжность системы и качество тестов».

Фактов: 0

Факты

«С **февраля моковый агент** пишет всё, что получил: **5** починок за **2** недели → почти **0**».

Фактов: 5

Факты — числа, имена, вердикты, на тысячу знаков

Мысли на абзац WIP

Одна мысль четыре раза



Почти всё — вода

Четыре разные мысли



Каждый абзац о своём

Непроверяемое можно пытаться проверять.

04.3
Ревью изменений целиком

Как оценить изменения, которые не прочитать построчно?

✓ Модель качества кода ✓ Качество текста

04.3 Ревью изменений целиком

Изменения — картой, а не построчно WIP

Ветка: скидки в корзине

Каталог
без изменений

Корзина
+180 строк

Оплата
+60 строк

Профиль
без изменений

Авторизация
+240 строк
не ждал

Уведомления
без изменений

UI-kit
+20 строк

Отчёты
без изменений

Свой инструмент пока делаем. Готовые: CodeBoarding и другие

А дальше что?

Отпустить контроль.

Фича работает.
Работает быстро.
Что тебе ещё надо?

Спасибо!

Вопросы?

Ссылка для вашего агента: пусть проанализирует ваш проект