

Heisenbug 2026 · 16–17 октября

Никита Осипов | Senior QA Automation Engineer

AI-агенты для Auto QA: от хаотичных запросов к системной работе с навыками

Java · Spring · JUnit 5

Никита Осипов

Senior QA Automation Engineer

Автоматизация тестирования бэкенда в банке



Больше шести лет в тестировании. Начинал с ручного



Автоматизация: сначала фронтенд, потом бэкенд.
Java · Spring



Следующий этап — разработка с AI-агентом.
Полгода: он пишет тесты, я ставлю задачи и проверяю

Мои вводные



Агент: OpenCode

Консольный агент с открытым кодом. Подойдёт Claude Code, Codex CLI или то, что разрешено у вас



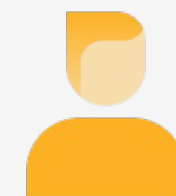
Модель: Qwen 3.5 397B

Развёрнута внутри компании. Подойдёт сопоставимая модель по API



Стек: Java · Spring · JUnit 5

Задачи в Jira, сценарии в Allure TestOps, агент ходит в них по MCP



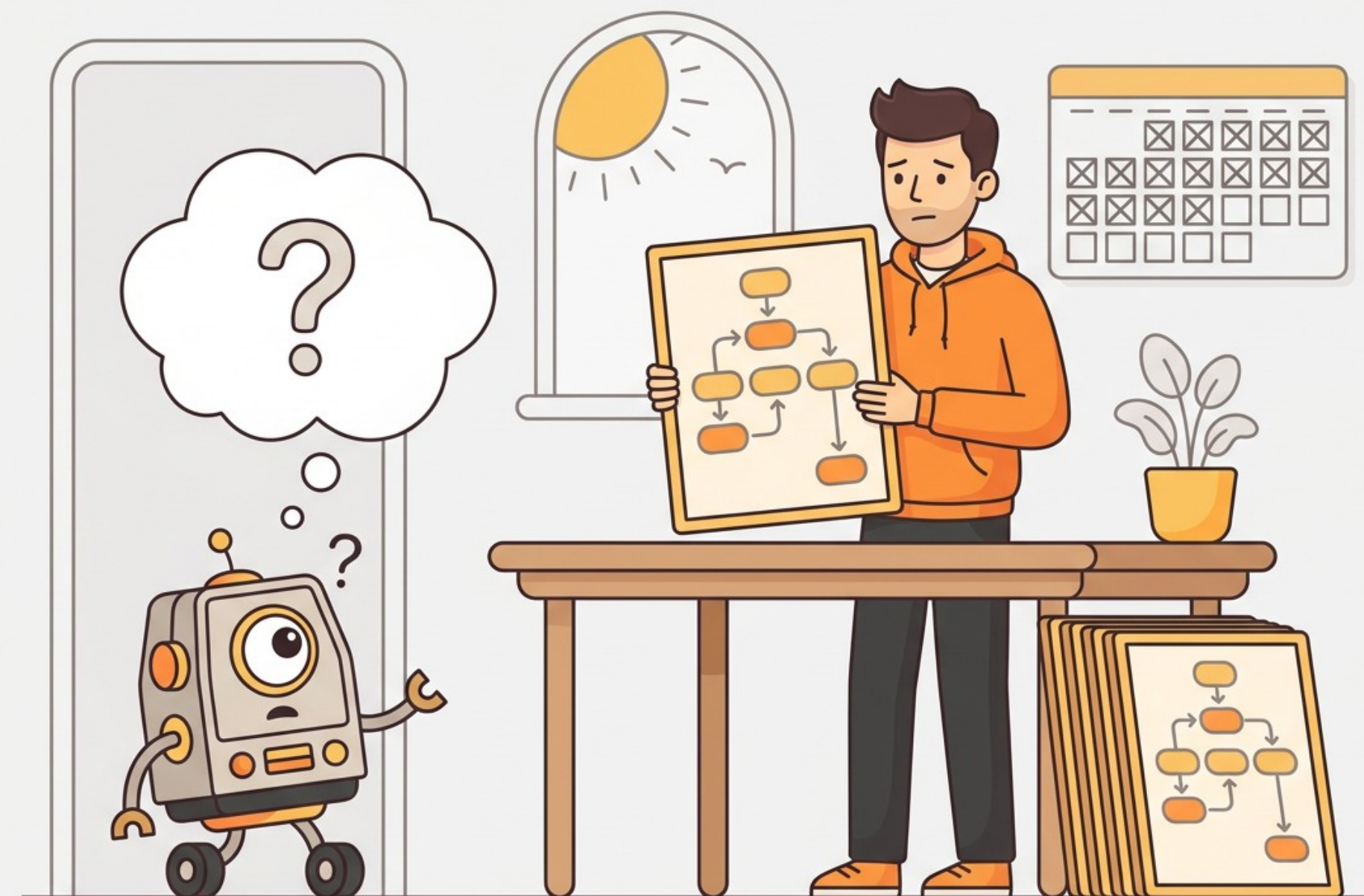
Команда: один AQA-инженер

Автоматизацией в команде занимаюсь я один. Все цифры в докладе — мои, за полгода

Никаких внутренних инструментов, только подход. На Kotlin, Python или Go всё то же самое

Агент — новичок в команде, который каждое утро приходит с амнезией

Онбординг, постановка задачи, ревью и допуски — как для человека, только каждый день



О чём поговорим

01



Онбординг

Память: как агент узнаёт проект и почему её нужно обновлять

02



Постановка задачи

План, порции и контекст: как ставить задачу, чтобы получить рабочий код

03



Ревью и допуски

Кто бьёт агента по рукам: проверки, мутации, права

04



Первый рабочий день

С чего начать в понедельник и что забрать с собой

Стадия хаоса: «напиши мне все тесты»



≡ opencode — chaos ×

> Напиши все интеграционные тесты для этого сервиса

- Сгенерировано 10 тестов, запускаю `mvn test`...

✗ не компилируется

✗ импорты из несуществующих пакетов

✗ Spring-контекст не поднимается

✗ выдуманные методы и утилиты, которых нет в проекте

✗ структура из учебника, а не из репозитория

Tests run: 10, Failures: 0, Errors: 10 BUILD FAILURE

+ Готово: 10 интеграционных тестов в `CompanyIntegrationTests`. Запускаю сборку, чтобы проверить компиляцию.

✓ Shell `mvn -q clean test-compile 2>&1 | tail -40` (Проверка компиляции тестов)

... first 22 lines hidden ...

[ERROR] src/test/java/.../company/CompanyIntegrationTests.java:[7,52] package org.springframework.kafka.test.utils does not exist

[ERROR] src/test/java/.../company/CompanyIntegrationTests.java:[9,41] package io.company.testsupport.kafka does not exist

[ERROR] src/test/java/.../company/CompanyIntegrationTests.java:[38,13] cannot find symbol

symbol: class KafkaTestHelper

location: class CompanyIntegrationTests

[ERROR] src/test/java/.../company/CompanyIntegrationTests.java:[57,32] cannot find symbol

symbol: method createTestCompany(java.lang.String)

location: variable companyService of type CompanyService

[ERROR] src/test/java/.../company/CompanyIntegrationTests.java:[71,6] cannot find symbol

symbol: class EmbeddedKafka

[INFO] 12 errors

[INFO] BUILD FAILURE

+ Похоже, в проекте нет утилит для Kafka. Создам `KafkaTestHelper` и `CompanyTestDataFactory` и повторю сборку.

Три слова: агент, скилл, МСР

01



Агент

Модель плюс инструменты:
читает файлы, запускает сборку,
ходит в Jira и TestOps

02



Скилл

Инструкция в markdown под
задачу: правила, шаги, стоп-
сигналы

03



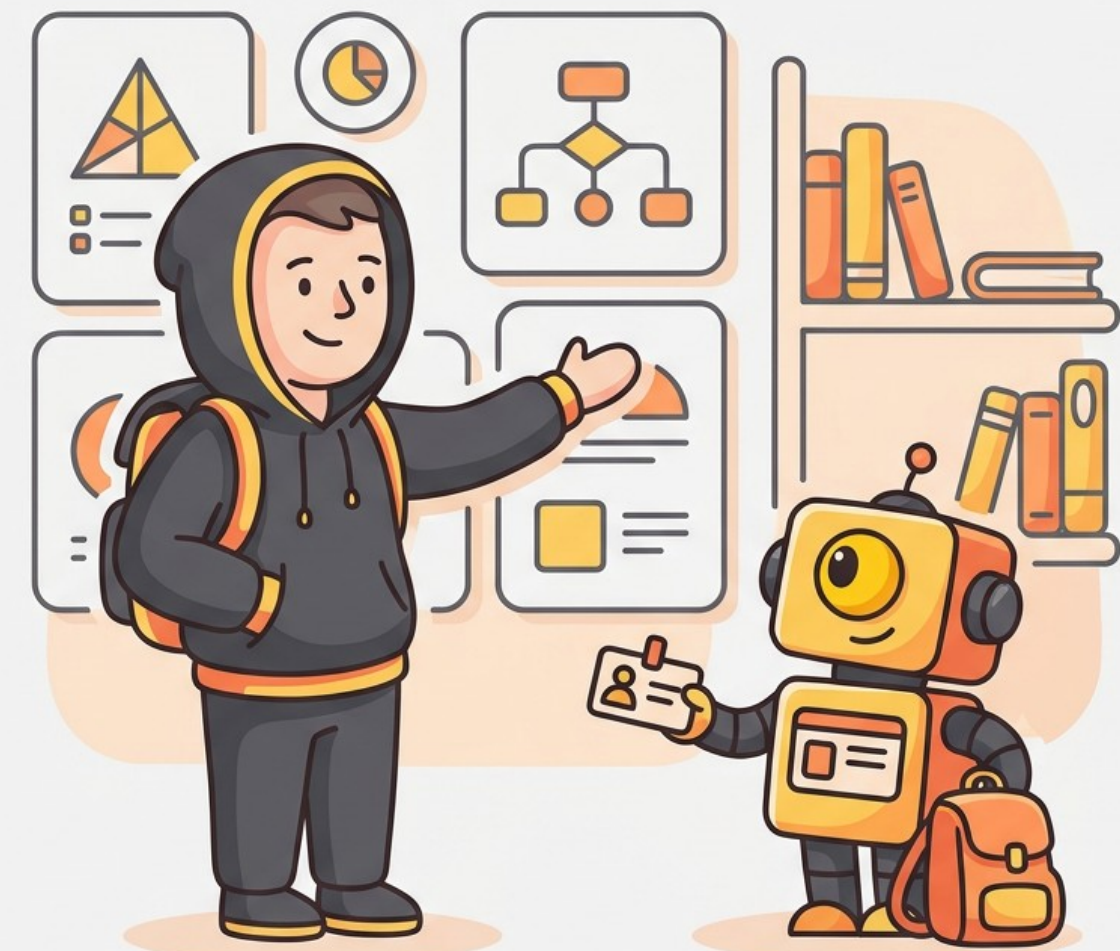
МСР

Руки агента: контракт, через
который он ходит в Jira,
Confluence, GitLab, TestOps

01

Онбординг: память

Первый день нового коллеги



Первый день: что делает ТИМЛИД

Четыре вещи, которые получает новичок

- ➔ Показывает проект: структуру, базовые классы, соглашения
- ➔ Говорит, где лежат сценарии и что мы вообще проверяем
- ➔ Даёт одну задачу, а не «напиши всё»
- ➔ Смотрит первый код внимательно

Агенту нужно то же самое. Только он забывает всё каждое утро, поэтому онбординг живёт в репозитории



Кто решает, что тестировать

Если агент сам придумывает сценарии, тесты проверяют фантазии модели, а не требования

- ✓ Сценарии живут в Allure TestOps, проверяет человек
- ✓ Каждый автотест ссылается на кейс по ID
- ✓ Покрытие считаем по сценариям, а не по строкам
- ✓ Проверка на CI: удалили тест — покрытие покраснело

Кейс в TestOps и тест с его ID

активный

#3914947 nonSme комиссия в пользу банка ежедневно в процентах

...

Обзор

История

Вложения

Карантин

Дефекты

Журнал изменений

Описание

Нет описания

Сценарий

из тестового результата

> 1 генерация данных по nonSme платежу

> 2 генерация данных по nonSme платежу

3 Вызов GET https://mock.gitlab.local/dc-ui-back/api/integration/v1/provider/directSettlements/mduumzzZHf_b_daily_percent

4 отправка в топик

5 отправка в топик

> 6 ожидание вычитки платежа в бд

> 7 ожидание вычитки платежа в бд

> 8 ожидание создания аутворда с prvCode = [mduumzzZHf_b_daily_percent]

9 Вызов POST https://mock.gitlab.local/dc-ag-pay/api/v1/aggregate/nonSme

> 10 ожидание перехода аутворда в статус [INITIAL]

> 11 проверка комиссий

Integration Tests

30/05/2024

10:08:59

История

успешный

успешный

успешный

успешный

успешный

Еще...

Ожидаемая длительность

5с

Теги

ISOLATED

MOCK

POSITIVE

SYNTH

Связанные задачи из баг-трекера

Нет связанных задач из баг-трекера

CalculateFeeTest.java

×

@AllureId("3914947")

@Test(description =

"nonSme комиссия в пользу

банка ежедневно в процентах")

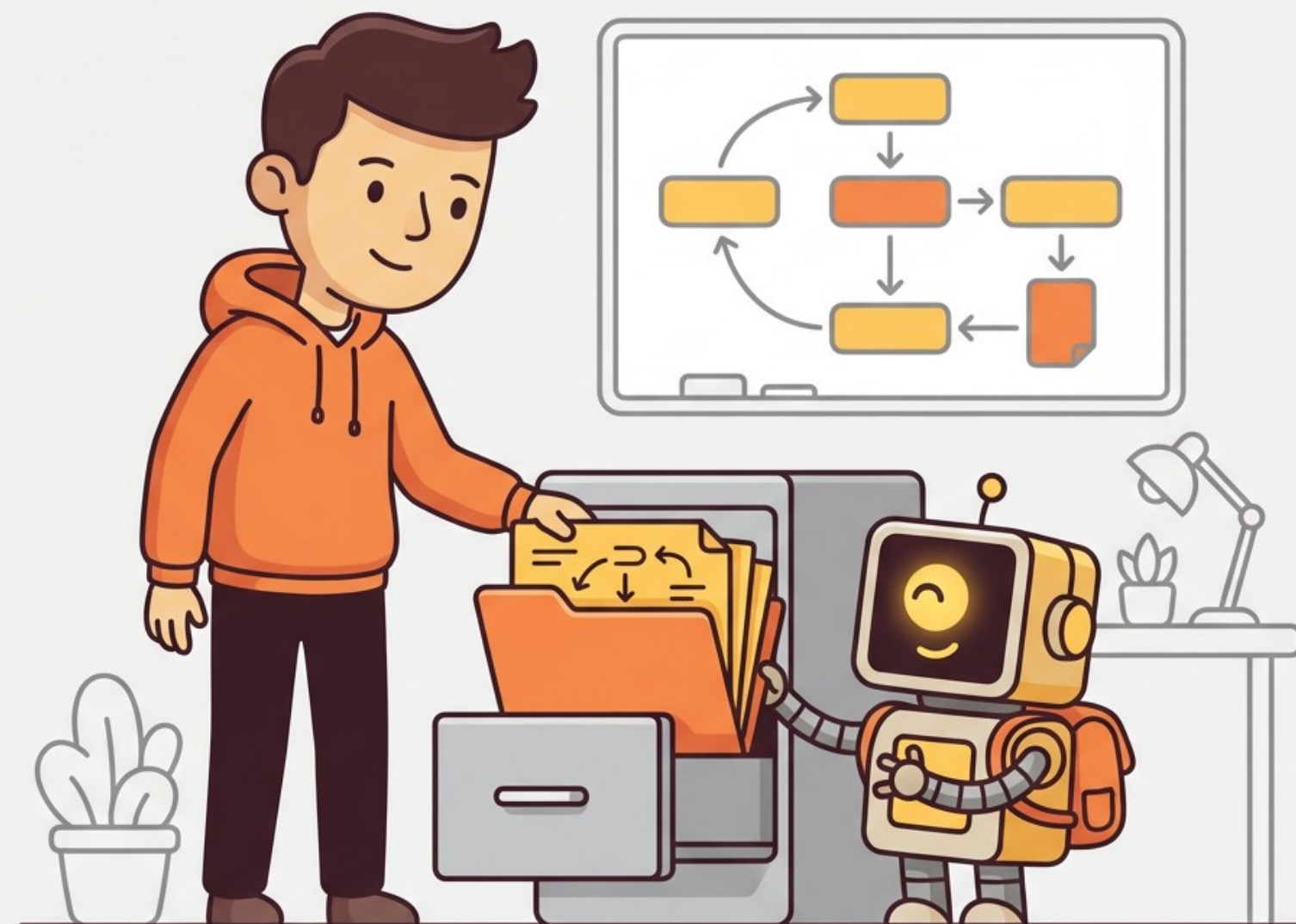


ID кейса в аннотации, название из карточки.

По этой паре агент понимает, что проверять, а CI видит, что тест на месте

Память — онбординг, записанный в репозиторий

Всё, что тимлид рассказал бы, лежит рядом с тестами. Агент читает это сам, каждое утро



Что создаёт memory-init: AGENTS.md



AGENTS.md

Корневой файл, не больше 150 строк. Агент читает его при старте каждой сессии

- ✓ Стек и версии
- ✓ Карта памяти со ссылками
- ✓ Жёсткие правила: что читать перед кодом, что запрещено
- ✓ Стоп-сигналы: когда остановиться и спросить



AGENTS.md ×

Стек

Java 17 · Spring Boot 3 · JUnit 5 · Maven. На Java 20 не собирается

Карта памяти

memory/architecture.md — слои и базовые классы

memory/examples.md — эталонные тесты, читать первыми

... ещё шесть файлов

Правила

- Перед кодом прочитай examples.md и test-patterns.md
- Новый хелпер — только если нет похожего
- Запрещено: Thread.sleep, System.out, моки в интеграционных

Стоп-сигналы: остановись и спроси

- Нужен новый базовый класс
- Нужна новая зависимость
- Правка вне src/test

Что создаёт memory-init: папка memory



memory/ — 8 файлов

project-overview.md	что тестируем, окружения, точки входа
architecture.md	слои tests → steps → clients, базовые классы, Spring-контекст
conventions.md	именование, стиль, запреты: Thread.sleep, System.out, адреса в коде
test-patterns.md	шаблоны: REST, база, Kafka, WireMock, параметризованный
frameworks-stack.md	библиотеки и версии. «Java 17, на 20 не собирается»
workflow.md	семь шагов работы по задаче
examples.md	3–5 эталонных тестов из репозитория как есть
common-pitfalls.md	грабли именно этого проекта



Прав всегда код

Разошлись — правь память, не код



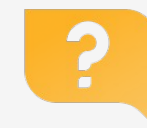
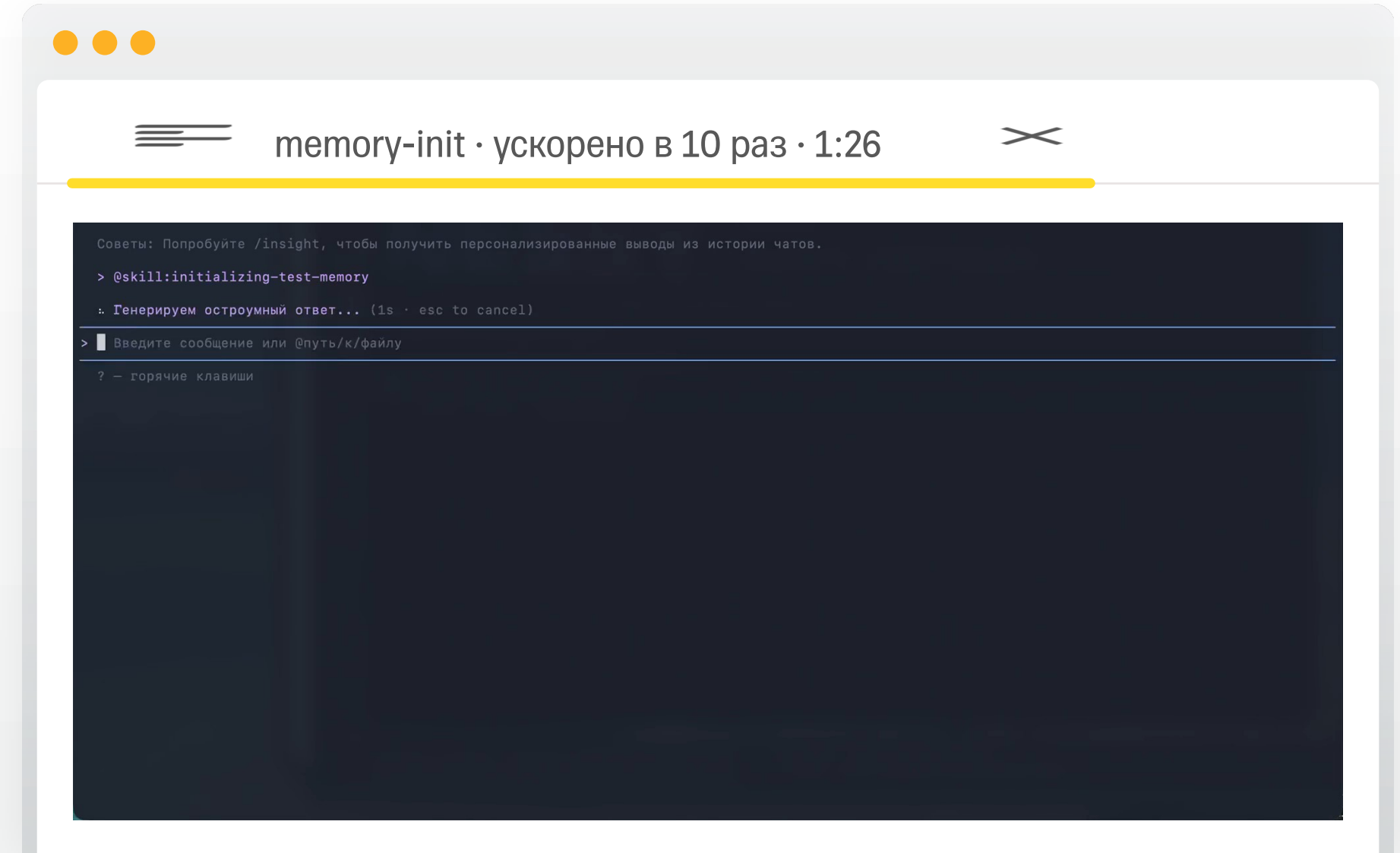
Вся память — не больше четверти окна

Иначе режьте примеры, а не правила

Как память появляется: семь шагов

Скилл memory-init: семь шагов, каждый с вашим подтверждением

- 01 Разведка репозитория без записи файлов
- 02 10–15 уточняющих вопросов. Не додумывает
- 03 Корневой файл AGENTS.md
- 04 Восемь файлов памяти, по одному, с показом
- 05 Самопроверка: классы существуют, нет противоречий
- 06 Боевая симуляция: реальная задача по памяти, без кода
- 07 Правки и коммит. Память едет в репозиторий



Каждый шаг заканчивается вопросом. Агент ждёт ответа и только потом пишет файлы



Самопроверка нашла ошибку в памяти. Написал «Java 8», в коде var и Map.of. Исправил сам

Без памяти и с памятью

Без памяти

- ✗ Выдумывает базовый класс
- ✗ Пишет Spring по учебнику
- ✗ Собирает на «какой-то» Java
- ✗ Каждую сессию проект объясняешь заново
- ✗ Из десяти тестов не работает ни один

С памятью

- ✓ Наследуется от реального AbstractTest
- ✓ Берёт конфигурацию из application.yml проекта
- ✓ Собирает на Java 17 из памяти
- ✓ Новая сессия знает проект за секунды
- ✓ Заготовка компилируется с первого раза

Одна оговорка, и она важная: память нужно поддерживать свежей →

Память устареваает

Устаревшая память опаснее отсутствующей



Без памяти

Агент хотя бы изучит код заново



С устаревшей

Не сомневается в том, что «помнит», и уверенно пишет по вчерашним правилам

Скилл memory-update



Когда запускать

- ✓ После влитой фичи
- ✓ После смены библиотеки или базового класса
- ✓ По симптому: агент уверенно ссылается на то, чего нет



Что делает

- ✓ Сверяет память с кодом, вики и задачами
- ✓ Показывает разницу, правит после подтверждения
- ✓ Пишет историю уточнений в корневой файл



Сейчас и дальше

- ✓ Сейчас запускаю руками, это минуты: сверяется только дельта
- ✓ Следующий шаг: хук после merge, чтобы память обновлялась сама

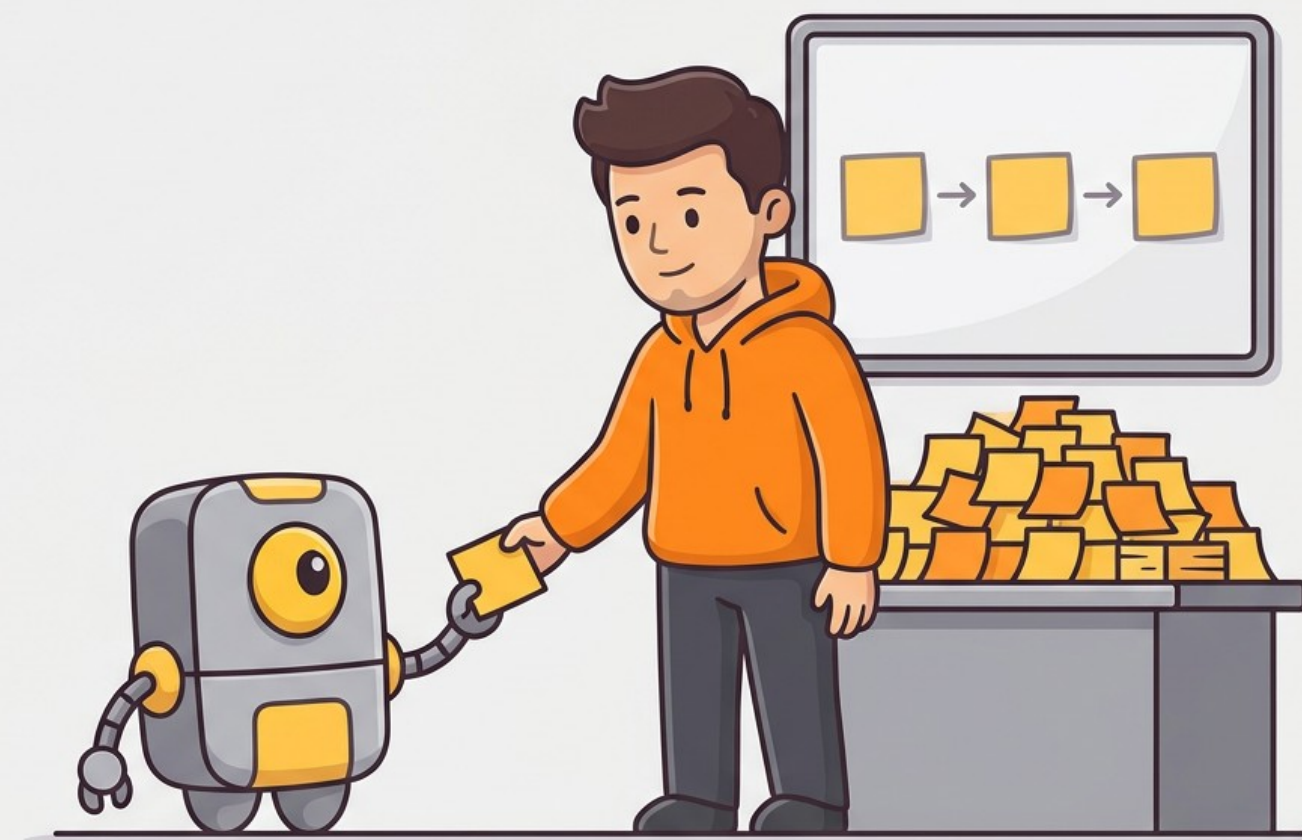


Это как сборка мусора: можно реже или чаще, но никуда не деться

02

Постановка задачи

План, порции, контекст



Одна задача: скилл `write-tests-by-task`

Семь шагов, стоп после каждого

Внутри SKILL.md: когда применять · шаги со стопами · стоп-сигналы · самопроверка теста · что запрещено

01

Загрузить память

05

Прогнать только новые тесты. Падения чинить, не подавляя

02

Прочитать задачу в Jira и пересказать в 5–10 строк

06

Коммит и MR с описанием

03

Найти похожие тесты и эталон. Составить план: список кейсов и целевые файлы. Ждать одобрения

07

Статус задачи и комментарий со ссылкой

04

Ветка и код строго по шаблонам и примерам

Сначала план, потом код



☰ opencode — режим Plan ✕

[Tab] режим Plan: читать и предлагать можно, менять файлы нельзя

[Tab] план одобрен: режим Build, появляется код

в Claude Code это plan mode



На шаге плана прошу составить негативные кейсы

Не попросил — получу только позитивные



Стоп-сигналы

Новый базовый класс, новая зависимость, правки вне тестов. На них агент обязан остановиться и спросить

Мой промпт поверх скилла



prompt — PROJ-1234 ✕

Напиши интеграционные тесты по задаче PROJ-1234.

1. **Тестовые данные:** возьми готовый хелпер создания компании через Kafka из CompanyTestHelper.
2. **Границы:** методы нового контроллера CompanyUserController.
3. **Тип тестов:** интеграционные, без моков.

версии окружения в промпте не нужны — они в памяти

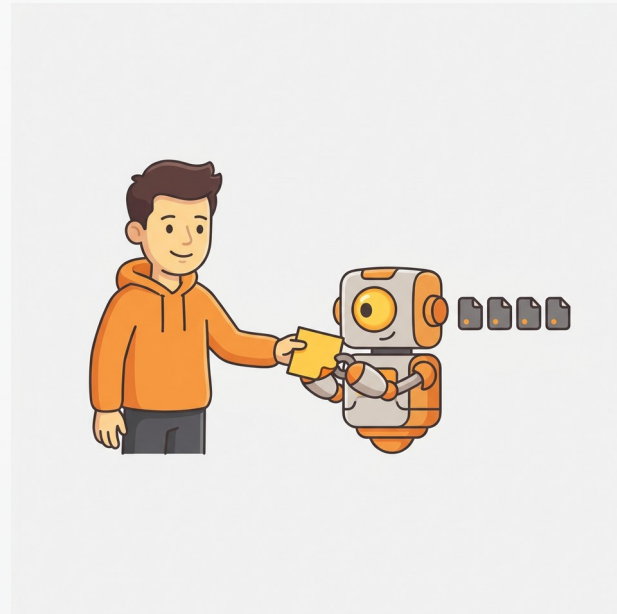
Три строки: данные, границы, тип. Всё остальное агент берёт из памяти и скилла

Размер порции — ваш рычаг

«Узкое место — ревью». А у кого ревью делают агенты — CI. Оба лечатся одним рычагом

Один контроллер

- ✓ 5–7 тестов в одном классе
- ✓ Ревью за минуты, весь контекст в голове
- ✓ Один прогон на CI



VS

«Весь сервис»

- ✗ Сотни строк по десятку файлов
- ✗ Ревью превращается в археологию
- ✗ CI гоняет всё подряд



Ломают ревью и CI не агенты, а размер порций

Что говорят данные

Faros AI, 2025

×2,5

размер PR у команд с AI

+91%

время ревью

«AI Productivity Paradox», 10 000+ разработчиков, faros.ai

LinearB, 2026

×5,3

дольше PR от агентов ждут
ревью

8,1 млн PR из 4 800 компаний, linearb.io

33% vs 84%

принимаются, против PR от
людей

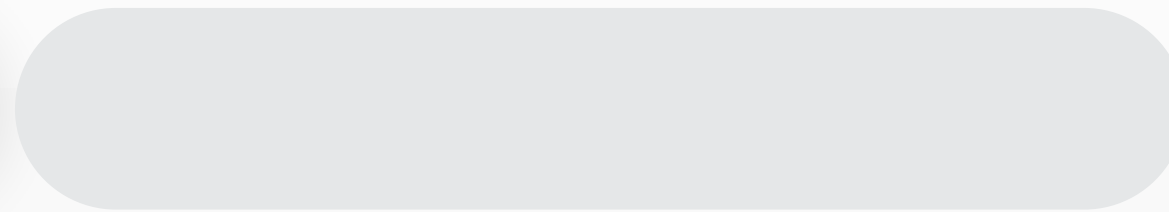


Обратная крайность: 50 MR за утро. Порция — одна задача из трекера, а не полсотни MR

Контекст: правило 50%



0–50% стабильная генерация



50–80% начинает забывать структуру проекта



80–100% галлюцинации: дописывает то, чего нет, ломает то, что работает



Половина окна — потолок

Новая задача начинается с чистого контекста. Память делает это бесплатным



/clear между задачами

Знания о проекте подтянутся из памяти за секунды



/compact с осторожностью

Если в сессии смешаны темы, после сжатия получается каша

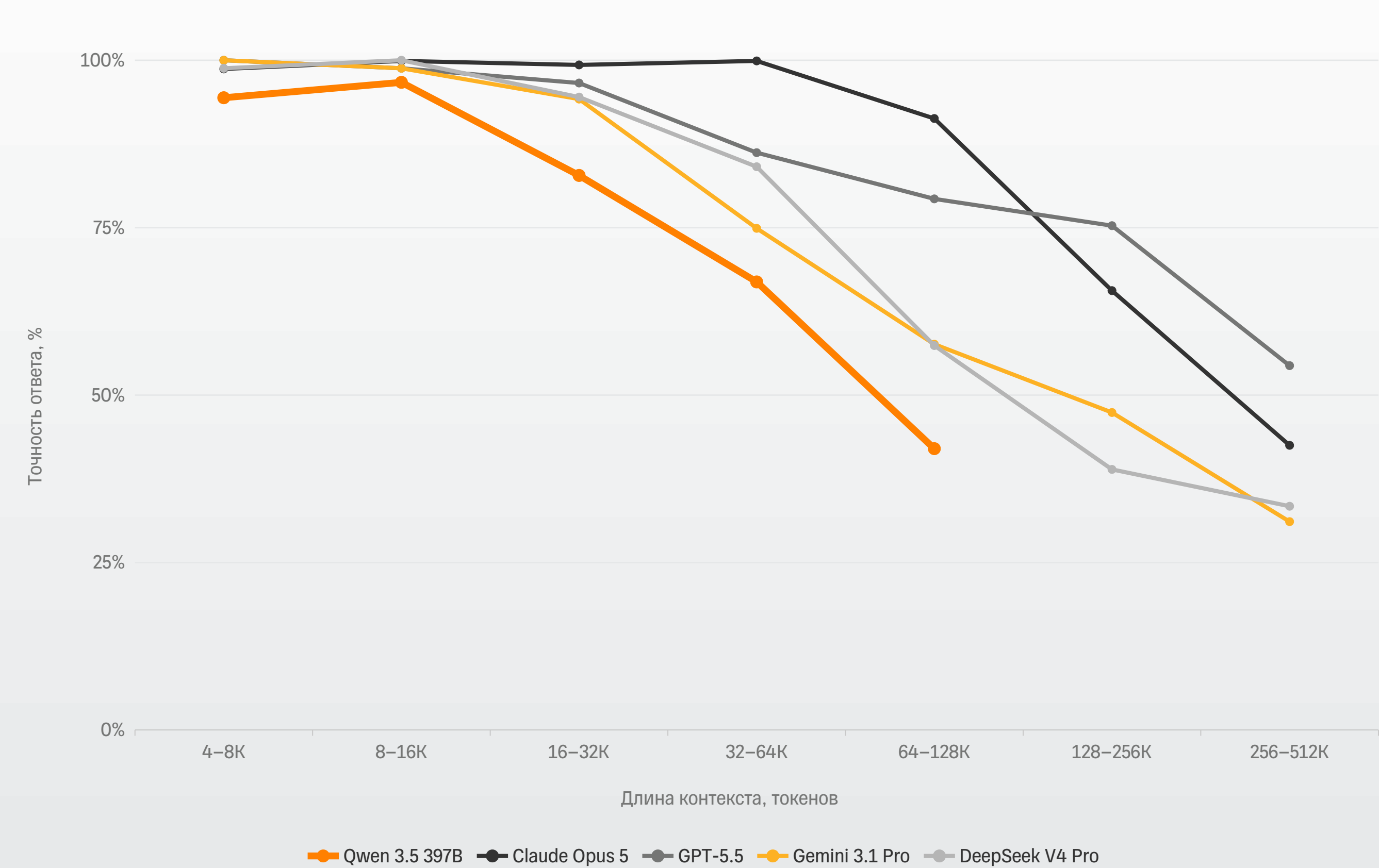


Побочные вопросы отдельно

Отдельным запросом. И не правьте код руками, пока агент работает с файлом

Контекст: что показывает рейтинг моделей

Насколько точно модель находит нужное в длинном диалоге. Чем правее, тем больше заполнен контекст



94% → 42%

Qwen 3.5 397B на половине своего окна 262K

Claude, GPT, Gemini, DeepSeek держатся дольше, но в самых длинных диапазонах теряют от 45 до 70 пунктов

Окно в миллион токенов — не миллион рабочих

Точность на MRCR v2 по диапазонам длины контекста (промпт + ответ). Источник: contextarena.ai, GDM MRCR v2 (8-needle), снимок 04.09.2026. Метод: Vodrahalli и др., «Michelangelo», 2024, arxiv.org/abs/2409.12640. Термин context rot: Chroma Research, июль 2025

Кейс Kafka: консоль

Компания создаётся через CRM и Kafka. Нужны тесты на новый контроллер пользователей. Запись 2:47, ускорена

Советы: Попробуйте `/insight`, чтобы получить персонализированные выводы из истории чатов.

- ⓘ Auto mode temporarily disabled these allow rules (they would bypass the classifier):
 - Skill(nc-helper) (from user settings)
 - Skill(project-memory-init) (from user settings)
 - Bash(node *) (from user settings)
 - Bash(npm *) (from user settings)
 - Bash(yarn *) (from user settings)
 - Skill(test-expert) (from user settings)
 - Skill(integration-guardian) (from user settings)
 - Skill(mf-developer) (from user settings)
 - Skill(bridge-expert) (from user settings)
 - Skill(memory-keeper) (from user settings)
 - Skill(writing-autotest-from-jira) (from user settings)
 - Agent(Explore) (from user settings)
 - Bash(python3 *) (from user settings)
 - Bash(mvn *) (from user settings)These will be restored when leaving auto mode.

> ⓘ skill:writing-autotest-from-jira DC-3174

Режим планирования (Shift + Tab для переключения)

Кейс Kafka: код

Итог: 4 зелёных теста и дефект сервиса, найденный по логам



CompanyUsersIntegrationTests.java



```
@Tag("POSITIVE")
@AllureId("3914952")
@Test(description = "Создание пользователя в компании")
public void createCompanyUserTest() {
    String companyId = createCompanyViaKafka();
    CreateUserRequest request = buildCreateUserRequest();
    CreateUserResponse response =
        step("Отправка запроса на создание пользователя",
            () -> companyApi.createCompanyUser(companyId, request));
    step("Проверка валидности ответа", () -> assertAll(
        () -> step("ssoId не null",
            () -> assertThat(response.getSsoId(), notNullValue())),
        () -> step("firstName совпадает",
            () -> assertThat(response.getFirstName(), equalTo(request.getFirstName()))),
        () -> step("lastName совпадает",
            () -> assertThat(response.getLastName(), equalTo(request.getLastName()))),
        () -> step("phone совпадает",
            () -> assertThat(response.getPhone(), equalTo(request.getPhone()))));
}
```



Хелперы взяты из проекта: билдер Kafka-сообщения, запросы к БД, генератор ИНН



Структура один в один с эталоном из памяти: step, assertAll, теги Allure

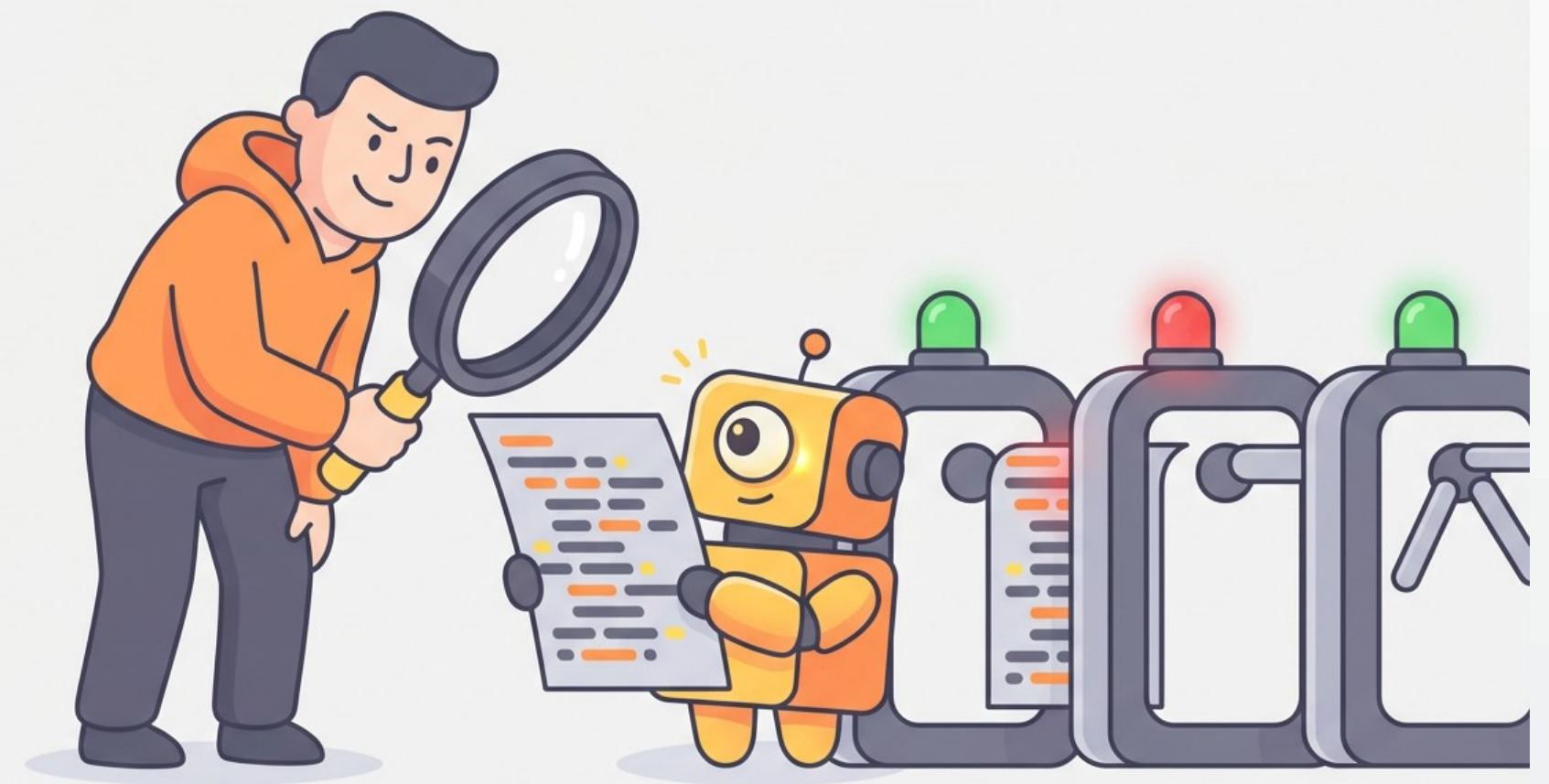


ID кейса из TestOps на месте: тест привязан к сценарию

03

Ревью и допуски

Кто бьёт агента по рукам



Кто бьёт агента по рукам

Ревью глазами нужно, но глаза субъективны и устают. Четыре ступени:

01

Проверки на CI

Сборка · Maven Enforcer:
версии · ArchUnit: слои и
запрещённые импорты ·
Sonar · ID кейса у каждого
теста

02

Ревью человеком

Работает за счёт маленьких
порций

03

Покрытие сценариев

У каждого кейса в регрессе
есть тест

04

???

А проверяют ли тесты
хоть что-то?



Правило, которое бьёт по рукам разработчика, так же бьёт и агента. Чем слабее модель, тем важнее первая ступень

Мутационное тестирование за минуту

Зелёный тест может ничего не проверять



PaymentService.java — мутант

```
// код
if (amount > limit) {
    reject(payment);
}

// мутант: инструмент поменял один символ
if (amount >= limit) {
    reject(payment);
}

$ mvn test -Dtest=PaymentServiceTest

✓ тест упал → проверка настоящая, мутант убит
✗ тест зелёный → проверка пустая, мутант выжил
```

Тест должен покраснеть. Не покраснел — тест ничего не проверяет

Юниты и интеграционные: разная арифметика



Юниты: PIT

Готовые инструменты. Тысячи мутантов, секунды на прогон



Интеграционные: перебор

Минуты на прогон × тысячи мутантов. Полный перебор не закончится никогда

Поэтому два агентских варианта

Вариант 1: самопроверка в скилле

01



Одна строка в скилле

После написания теста: сломай проверяемую логику, убедись, что тест упал, верни как было

Ломаем то, что под рукой: код сервиса, ответ стаба или входные данные. Прошёл на подменённых данных — проверка пустая

Для каждого теста и бесплатно



write-tests-by-task · консоль



Самопроверка теста

- Сломай проверяемую логику, запусти тест
- Упал: верни логику как было
- Зелёный: проверка пустая, перепиши

\$ ломаю CompanyUserService.create

✗ createCompanyUserTest: зелёный

• тест не упал, переписываю проверку

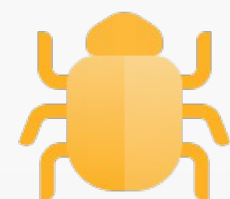
\$ прогон после правки

✓ createCompanyUserTest: упал на сломанной логике

\$ возвращаю логику как было

Вариант 2: агент-вредитель

02



Второй агент, чистая сессия

Код сервиса и тестовый набор. Задание наоборот:

«испорти код так, чтобы тесты остались зелёными».

Получилось — нашли пустой тест. Отчёт — список на доработку

Бьёт по бизнес-логике прицельно. Только начинаем пробовать



☰ opencode — вредитель ✕

> Испорти бизнес-логику так, чтобы тесты остались зелёными. Отчёт: что сломал, какие тесты не заметили.

Мутация 1: убрал проверку лимита в PaymentService

тесты зелёные ✕

Мутация 2: поменял формулу комиссии

упал CalculateFeeTest ✓

Мутация 3: закомментировал удаление ролей

тесты зелёные ✕

На доработку: PaymentLimitTest, CompanyUserRolesTest

Идея не с потолка

73%

тестов от LLM приняли инженеры в код

Meta* АСН: LLM пишет мутанты и тесты к ним. engineering.fb.com, 30.09.2025

88% vs 42%

реальных багов воспроизводят мутанты от LLM против классических инструментов

Wang, Harman и др., 851 реальный баг. arxiv.org/abs/2406.09843

Не доверяй одной LLM — поставь напротив вторую

* Meta признана экстремистской организацией, её деятельность запрещена на территории РФ

Мои цифры

Было: промпты без памяти, полгода назад. **Стало:** память, скиллы и проверки. **Один AQA-инженер**

$\infty \rightarrow 2-3$

заходов до зелёного теста

2 дня $\rightarrow$ 1 день

от задачи до MR, в среднем. Не в
десять раз: проверка никуда не делась

6 $\rightarrow$ 11

MR с автотестами в месяц, на одного
человека. Рост за счёт трёх
параллельных сессий на три сервиса

Что не попало в карточки: в показанном кейсе 4 из 7 зелёных, три падения — дефект сервиса. Покрывание сценариев в TestOps — цель 100%. Контекст больше 50% на задачу — порции слишком крупные

Не считаю строки кода: агент нагенерит их сколько угодно. Крупные исследования общего выигрыша не видят: кто работал хорошо, стал лучше, кто плохо — хуже. Измеряйте себя

Грабли 1 из 5: Модель подменилась под капотом



Что случилось

Качество упало «внезапно», я переписывал промпты, а под обвязкой подхватилась модель поменьше

Что делать

Проверять модель раньше, чем промпты

Грабли 2 из 5: Не обновил память



Что случилось

Агент уверенно писал по вчерашним шаблонам:
базовый класс переехал, а в памяти он лежит на старом
месте

Что делать

memory-update после каждой влитой
фичи

Грабли 3 из 5: /сотраст в смешанной сессии



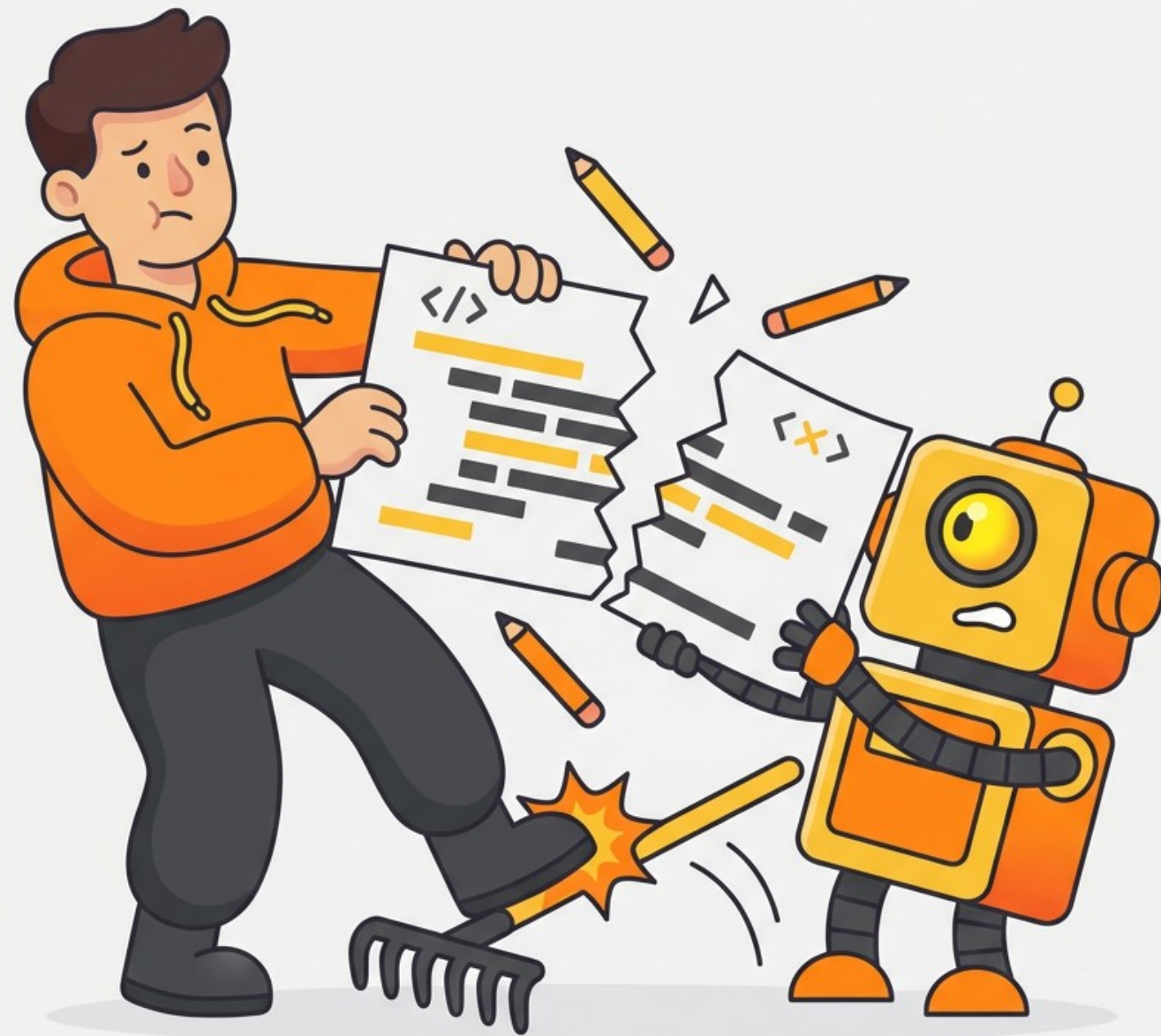
Что случилось

Каша вместо контекста: после сжатия агент отвечал как чат, а не как исполнитель

Что делать

/clear между задачами вместо сжатия

Грабли 4 из 5: Правил код руками параллельно



Что случилось

Разорвали файл пополам, и ни один из нас не понял, что произошло

Что делать

Поочерёдно: он сделал, вы посмотрели

Грабли 5 из 5: Поверил «тесты прошли»



Что случилось

Зелёная сборка «7 из 7» была на локальном стабе. На реальном сервисе упали три теста из семи

Что делать

**Зелёный только на реальном окружении.
Плюс самопроверка и агент-вредитель**

04

Первый рабочий день

С чего начать в понедельник



С чего начать в понедельник



сфотографируйте

01

Проверьте, что ваш агент по умолчанию использует самую умную модель из доступных

05

/clear между задачами. Половина окна — потолок

02

Создайте AGENTS.md: стек, версии, где лежат сценарии, что запрещено

06

Включите проверки на CI: Enforcer, ArchUnit, ID кейсов. Добавьте в скилл самопроверку теста

03

Запустите memory-init на репозитории автотестов. Шаблон по QR-коду через слайд

07

Права по мере доверия: чтение, подтверждение, автономия. Автономия только в тестовом репозитории, с проверками на CI и откатом

04

Одна задача на один контроллер. Сначала план, одобрили, потом код. Негативные кейсы просите на плане

Шаблоны скиллов в markdown: ничего специфичного для Java в них нет

**Агент не заменяет QA-инженера.
Агент заменяет ту часть работы,
где вы были принтером**

Разочарование в агенте почти всегда — разочарование в собственной постановке задач

Материалы и контакты

- Шаблоны скиллов: `memory-init`, `memory-update`, `write-tests-by-task`
- Порядок действий на понедельник: `monday.md`
- Ссылки на исследования из доклада: `research.md`

Telegram: @luckymanone

naosipov22@yandex.ru



github.com/osipovn/agents-skills