



Context Engineering — новая профессия инженера



Николай Константинов

Тимлид фронтальной разработки • Сбер



Немного обо мне

Моя работа

- 6 лет во фронтенде
- 5 лет работаю в Сбере
- 1,5+ года разрабатываю AI-агентов

Комьюнити

- Выступаю на HolyJS, Podlodka, MoscowCSS и «Стачке»
- Пишу статьи вместе с ИТМО

Кажется, я ошибся
конференцией.

HolyJS пройдёт позже :)

От инженера к тимлиду

Раньше я отвечал за код. Теперь — за delivery, знания и способы работы команды.

Что меняется, когда становишься тимлидом

Мой результат теперь зависит от всей команды

Раньше

Я сам делал feature, знал код и быстро находил нужный ответ в своей голове.

Сейчас

Я распределяю задачи, снимаю блокировки и помогаю десяти людям пользоваться общими знаниями.

10 человек. Web, iOS и Android

Сервисная команда для более чем 20 продуктов СберОнлайн

Delivery

Оценки, распределение нагрузки и выпуск изменений. Нужно видеть всю цепочку поставки, а не отдельный PR.

Engineering enablement

Создаём условия, в которых хороший подход можно повторить на любой из трёх платформ и в разных продуктах.

Как знания возникают внутри команды

Разработчики не ждут приказа на каждую маленькую автоматизацию

Web

Инженеры собирают skills для типовых UI-задач, проверок компонентов, accessibility и миграций.

iOS

Экспериментируют с агентами для повторяемой работы: тестов, шаблонов и компонентов дизайн-системы.

Android

Создают свои расширения, подключают инструменты и переиспользуют платформенные соглашения.

Одна feature — три инженерные среды

Продуктовое намерение одно, а контракты реализации разные

Web

Свои репозитории, инструменты сборки, компоненты и правила аналитики. Агенту нужен Web-контекст.

iOS

Swift, собственные библиотеки и ограничения платформы. Нельзя подставлять Web-инструкции.

Android

Kotlin, внутренние API, архитектура и циклы проверки. Общий UX не означает общий код.

Разный размер платформенной команды

Android: четыре инженера. iOS: два. Пропускная способность отличается.

Не всё решает найм

Ищем повторяемые операции, которые можно облегчить агентами: каркасы, тесты, компоненты.

Решения остаются у людей

Платформенная специфика, архитектура и ответственность за итог остаются у специалистов.

Самый дорогой ресурс — контекст



Модель знает языки программирования. Но ей нужно объяснить, как устроена именно наша система.

AI усиливает существующую команду

DORA 2025: эффект зависит от инженерных практик, а не только от модели.

Внешний ориентир

DORA изучает AI как часть системы разработки: культуру, качество процессов, обратную связь и доставку изменений.

Что вижу в команде

Если знания о библиотеках устарели, агент быстрее повторит ошибку. Если договорённости ясны — их проще масштабировать.

AI уже стал повседневным инструментом

90%

участников исследования DORA
используют AI в работе

Исследование почти 5 000 технологических специалистов. Это не показатель эффекта на доставку продукта.

Источник: DORA / Google, State of AI-assisted Software Development, 2025

<https://blog.google/innovation-and-ai/technology/developers-tools/dora-report-2025/>

Внедрение AI — это работа тимлида

Недостаточно выдать доступ и ждать роста эффективности

Доступ

Инженер получает AI-инструмент и может безопасно экспериментировать с ограниченным сценарием.

Общая практика

Мы обсуждаем удачные решения, оформляем skills, расширения и инструкции в переиспользуемые артефакты.

Обратная связь

Собираем фейлы, назначаем владельцев и исправляем контекст, инструменты и сам workflow.

У команды две скорости

Локальное ускорение — ещё не стабильный delivery

Скорость человека

«Я быстрее написал этот компонент». Важно, но само по себе не решает межплатформенные зависимости.

Скорость команды

«Второй инженер повторил сценарий без автора». Это уже вклад в совместный процесс и его предсказуемость.

От личного AI к команде

От полезного эксперимента к общему инженерному активу

Личная находка

Промпт, короткий скрипт или автоматизация рождаются в реальной задаче разработчика.

Командный skill

На встрече обсуждаем применимость, выделяем ограничения и пример воспроизводимого результата.

Marketplace

Публикуем проверенный способ работы с владельцем, версией и областью применения.

Обратная связь

Другие инженеры пробуют сценарий и приносят случаи, в которых он ошибается или не подходит.

AI adoption растёт быстрее доверия

84%

используют или планируют
использовать AI в разработке

46% респондентов не доверяют точности AI-выводов; 33% доверяют.

Источник: Stack Overflow Developer Survey, 2025

<https://survey.stackoverflow.co/2025/ai>

Context Engineering

Нужные знания в нужный момент, а не бесконечные промпты.

О чём расскажу на практике

Пять вопросов из опыта кроссплатформенной команды

Знания

Как знания инженеров перестают теряться между проектами и платформами?

SSOT

Почему первоисточник должен жить рядом с библиотекой, которой он принадлежит?

UI Context

Что Web, iOS и Android могут разделять, если код и стек у них разные?

Вклад команды

Куда разработчики контрибютят skills, плагины, фреймворки и MCP?

Результат

Как отличить демонстрацию AI от реально полезного улучшения процесса?



КОНТЕКСТ

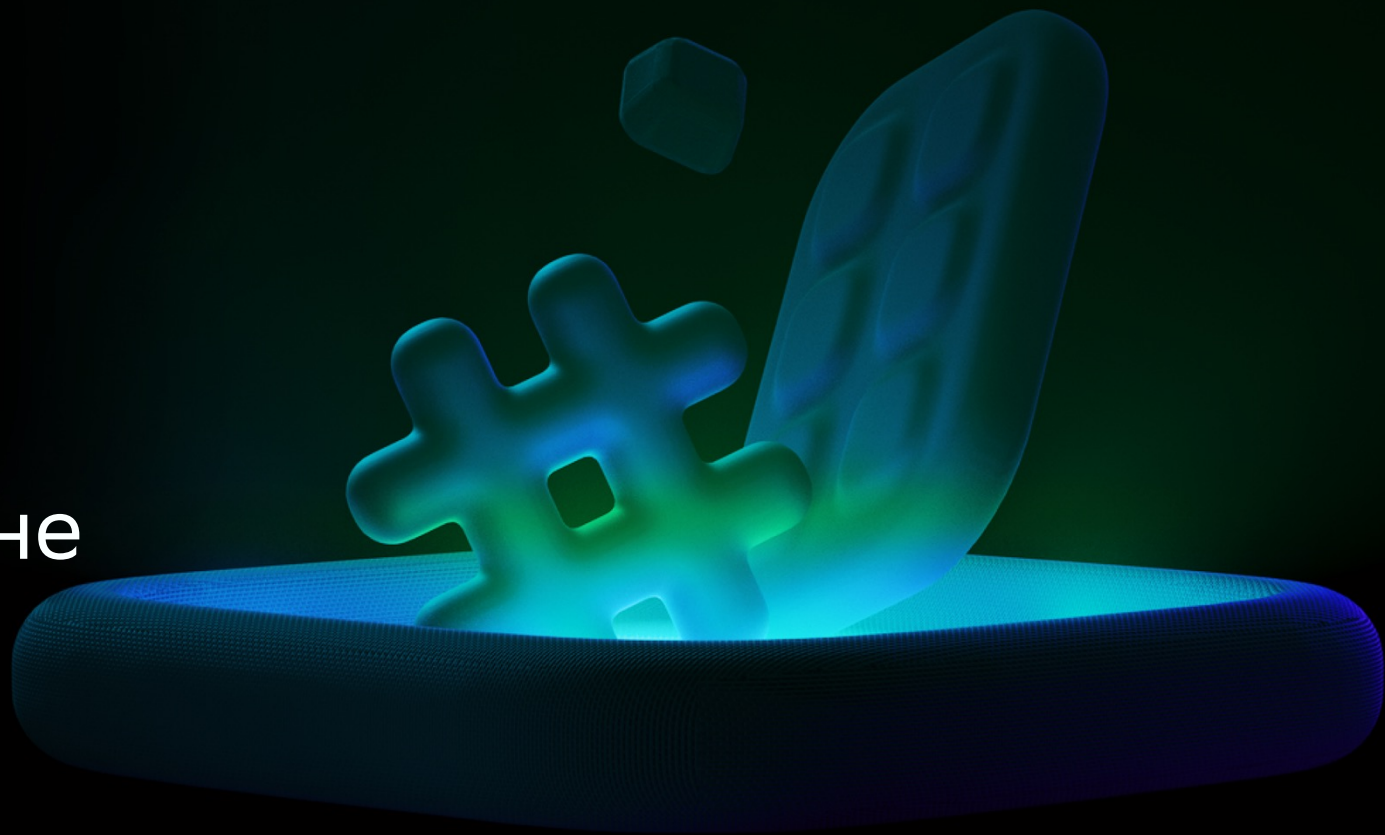


Знание должно быть доступным, актуальным и проверяемым.



Команда
Сбера

Context Engineering — не один файл



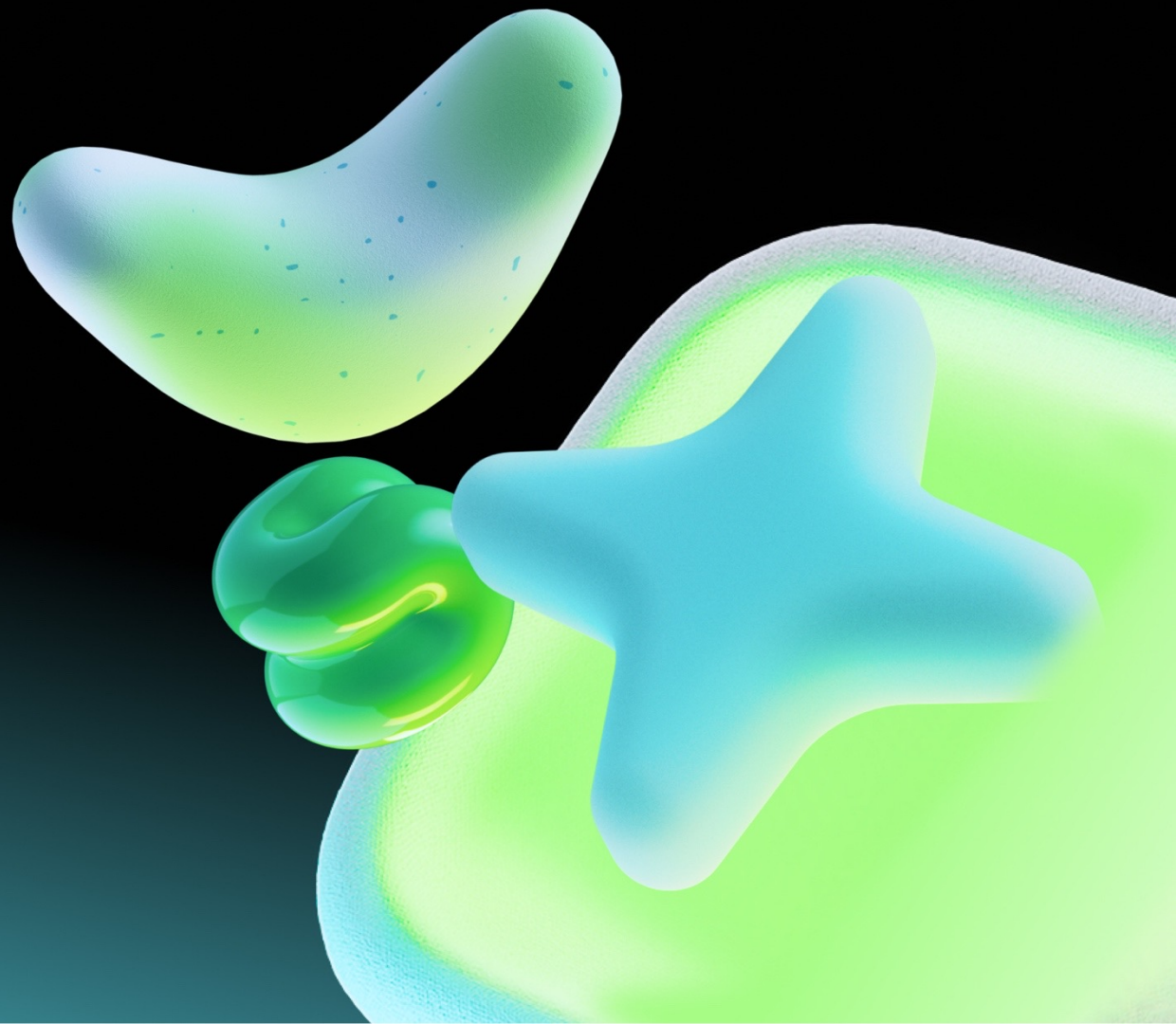
Это дисциплина владения, версионирования и
доставки инженерных знаний.



Команда
Сбера

Project SSOT

Локальные архитектурные решения, навигация,
аналитика и бизнес-ограничения продукта.

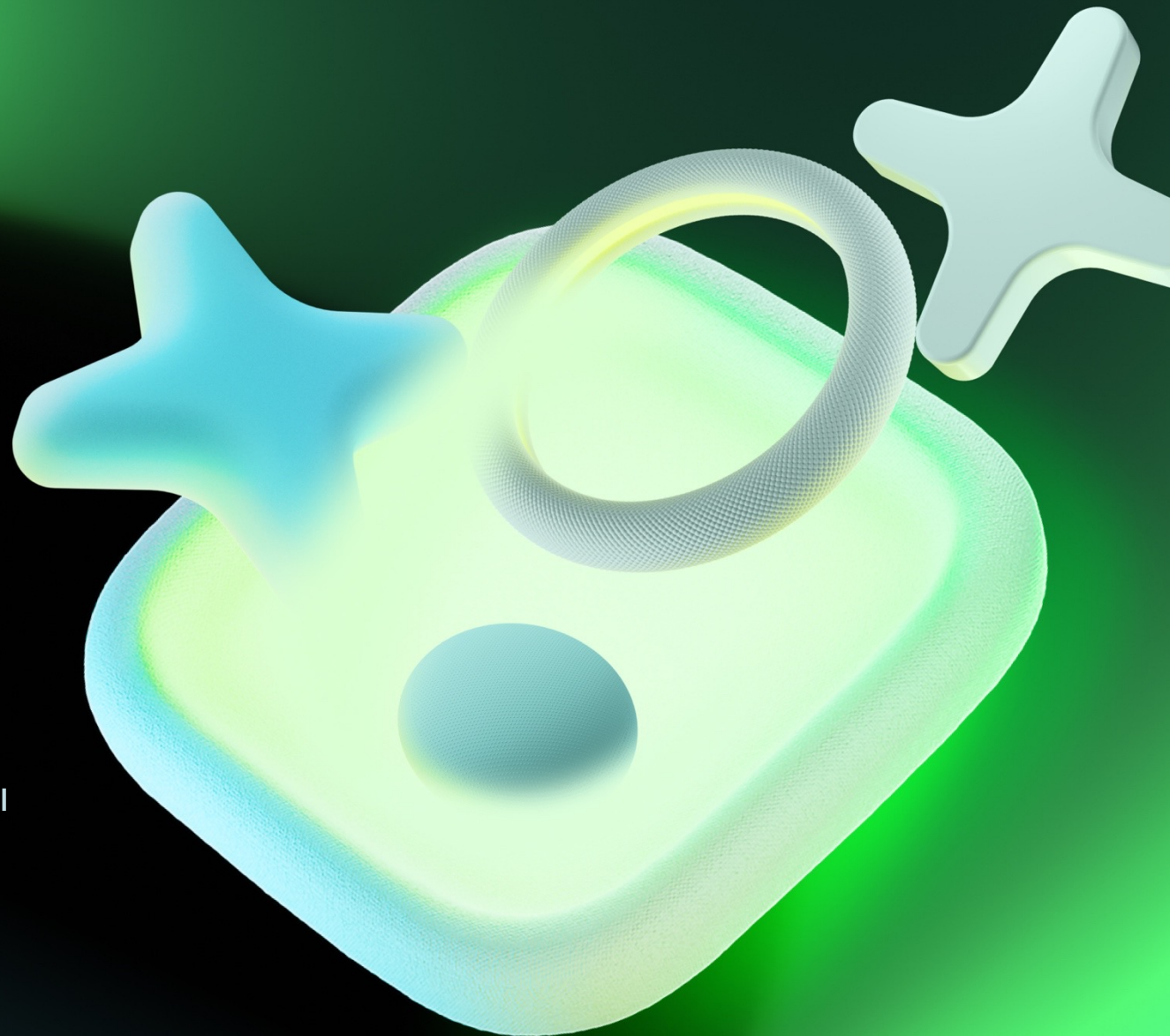




Команда
Сбера

Library SSOT

API, версии, примеры и ограничения должны обновляться вместе с платформенной библиотекой.





Команда
Сбера

UI Context

Общие макеты, состояния и пользовательские сценарии — между Web, iOS и Android.

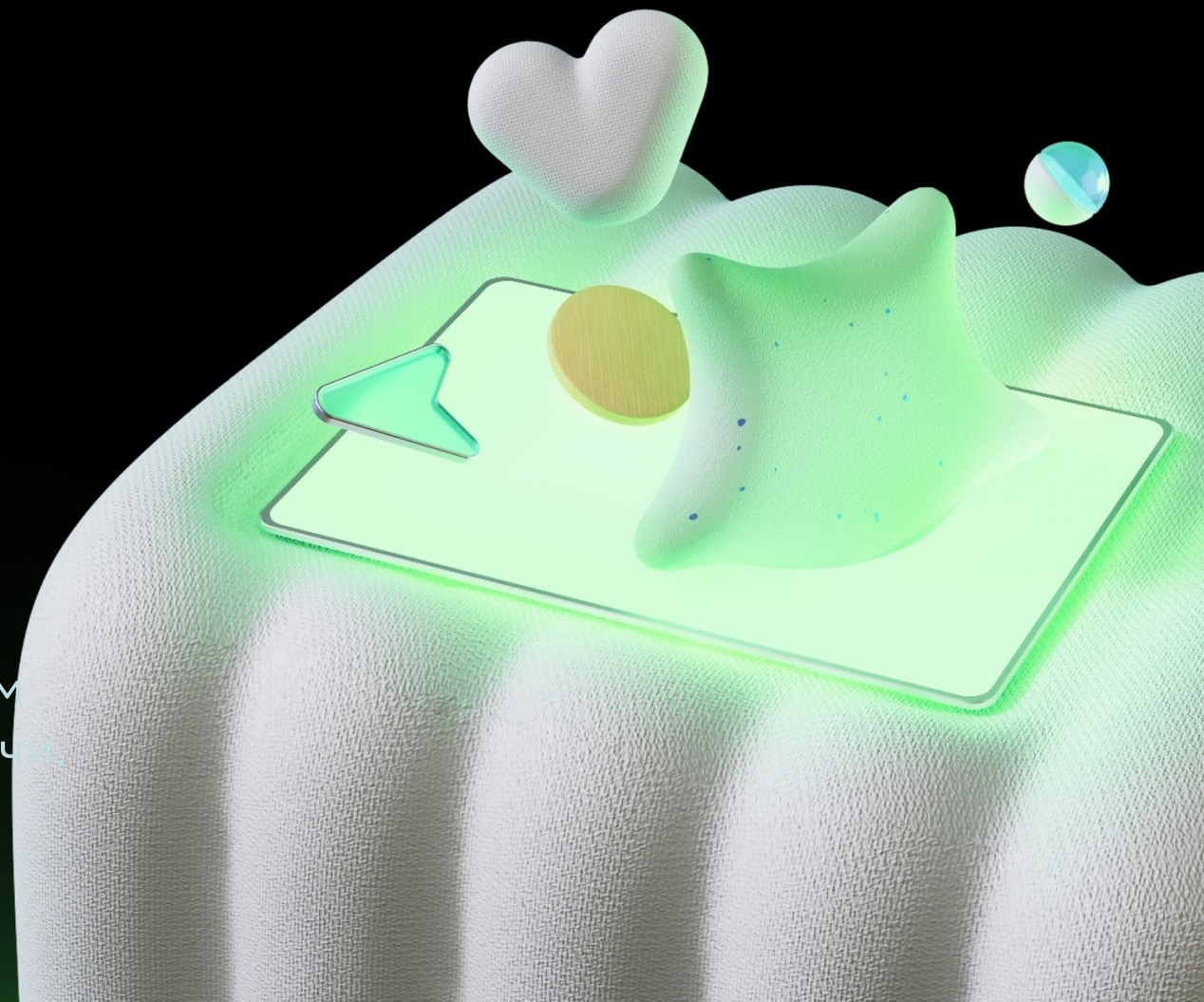




Команда
Сбера

Контекст по запросу

Не загружаем в агента всё подряд. Подключаем
только нужные источники для конкретной задачи



Что у нас действительно общее

Не исходный код: общий пользовательский интерфейс и продуктовый смысл

Design

Экраны, композиция, визуальные требования и макеты.

States

Loading, error, empty, disabled и другие воспроизводимые состояния.

Flows

Переходы, сценарии, обратная связь и ожидаемое поведение.

Validation

Acceptance criteria, визуальные тесты и доступность интерфейса.



Команда
Сбера

У reuse есть цена

Удобная инструкция постепенно становится стандартом. Значит, у неё должны быть владелец и срок пересмотра.

Четыре слоя контекста для feature

Собираем их по владельцу и задаче, а не в один огромный файл

Product intent

Требование, пользовательский сценарий, бизнес-правила и критерии приёмки изменения.

Project SSOT

Модульная архитектура, навигация, аналитика, локализация и проектные ограничения.

Library SSOT

Компонентные API, поддерживаемые версии, паттерны композиции и запреты.

UI Context

Макеты, responsive, loading/error/empty states, flows и ожидаемое поведение.

Почти правильный код тоже стоит времени

66%

сталкиваются с ответами AI, которые
«почти верны»

45% называют трудоёмкую отладку AI-кода проблемой. Это самооценки участников опроса.

Источник: Stack Overflow Developer Survey, 2025

<https://survey.stackoverflow.co/2025/ai>

Зачем инвестировать в reuse

Гипотетический расчёт — не результат нашей команды

40 ч

Создать один проверяемый сценарий миграции

20

Команд могут использовать готовую практику

10 ч

Экономия на одном повторном сценарии

≈ 200 ч

Валовый эффект в условном примере

Иллюстративный расчёт; 40 часов разработки нужно учитывать при оценке чистого эффекта.

Документация и SSOT дополняют друг друга

SSOT — не новая копия всех существующих документов

Документация

Человек читает правила и решает, как их применить. Но документы нередко устаревают или расходятся.

Машиночитаемый контракт

Агент получает версию правил из первоисточника, а проверки подтверждают их применение в задаче.

Стандартизация не означает одинаковый код

Объединять Web, iOS и Android в один технический SSOT опасно

Что можно разделить

Дизайн, состояния, сценарии, критерии готовности, терминологию продукта и UX-ограничения.

Что остаётся локальным

Репозитории, архитектура, SDK, сборка, библиотеки, особенности тестирования и интеграций.

Из чего собирается контекст агента

Шесть частей инженерной среды, а не шесть новых PDF

Entry Point

Точка входа из CLI или IDE. Инженер выбирает задачу и нужную capability.

Marketplace

Поиск общих skills, плагинов и агентов с владельцем, версией и ограничениями.

Runtime

Запуск задач и инструментов с разрешениями, проверками и точками участия человека.

Context

SSOT продукта и библиотек, спецификации, UI-артефакты и правила безопасности.

Evaluation

Наборы типовых задач и регрессионные проверки, позволяющие сравнивать версии.

Governance

Права доступа, аудит и классы риска. Особенно важно для изменений с последствиями.

Инженеру нужна одна точка входа

Самостоятельно собирать каждый раз десятки MCP и файлов контекста неэффективно

Что делает разработчик

Выбирает задачу: проверить экран, подготовить миграцию, создать тесты, исправить a11y.

Что обеспечивает платформа

Находит подходящий skill, выдаёт актуальный SSOT, подключает инструменты и проверяет результат.

МСР: два повседневных примера

Идея — меньше ручного переноса информации между сервисами

Task tracker

Агент читает актуальную задачу, требования, зависимости и критерии приёмки.

Design tools

Агент получает дизайн, токены, состояния и ссылки на UI-артефакты.

Инженер

Контролирует доступ и подтверждает, что источники действительно актуальны.

МСР соединяет агента с реальной работой

Интеграции с task-трекером и макетами — часть контекста

Task-трекер

Получить актуальную постановку, связанный еріс, ограничения и acceptance criteria.

Дизайн

Открыть макет, состояния компонентов, токены и ссылки на актуальные визуальные артефакты.

Проверяемый результат

Проследить, какое требование и какой дизайн повлияли на конкретное решение агента.

Sarability должна быть понятной другим

Внутренний marketplace нужен для поиска и доверия

Карточка

Название, назначение, владелец, поддерживаемая платформа и ограничения применения.

Основания доверия

Версия, changelog, quality checks, реальные примеры запуска и известные ограничения.

Плагины — тоже инженерный вклад

Не всё полезное для AI обязано называться skill

Расширения

Разработчики создают адаптеры для задач платформ, reusable tools и полезные команды CLI.

Интеграции

MCP-серверы для дизайна, задач, документации и внутренних сервисов с проверяемыми правами.

Платформенные библиотеки

SSOT и примеры компонентов становятся частью процесса разработки, а не отдельным архивом.

Context Supply Chain

Как знание попадает из первоисточника в конкретный запуск агента

Источник

Контракт библиотеки, ADR, задачи, дизайн и правила продукта. У каждого есть владелец.

Подготовка

Проверяем версию, применимость, конфликты и свежесть документации.

Доставка

Агент получает подходящую часть контекста в момент работы, а не всё подряд.

Обратная связь

Реальные ошибки показывают, какой SSOT или способ доставки нужно обновить.

Почему агент выбрал старый компонент

Это может быть проблема цепочки знаний, а не качества модели

Симптом

Агент вставил deprecated UI-компонент и предложил код, который выглядит убедительно.

Причина

В проекте оказалась устаревшая копия документации вместо SSOT актуальной библиотеки.

Исправление

Правим первоисточник, удаляем конфликтующий контекст и добавляем regression case.

Один удачный запуск — не evaluation

Для публикации skill нужны воспроизводимые проверочные кейсы

Набор задач

Проверяем обычный экран и сложные случаи: состояния, сетевые ошибки, старые зависимости.

Автоматические проверки

Сборка, тесты, архитектурные правила, доступность и корректность использования UI-кита.

Human review

Инженер проверяет UX и проектные решения. Протоколируем замечания и ручные исправления.

Повторяемость

Сравниваем версии на одних и тех же кейсах, а не только по субъективному впечатлению.

Польза для разработчика ещё не командный эффект

17%

пользователей AI-агентов отмечают
улучшение командного
взаимодействия

При этом 69% пользователей агентов сообщают о росте личной продуктивности. Разные показатели одной выборки.

Источник: Stack Overflow Developer Survey, 2025

<https://survey.stackoverflow.co/2025/ai>

Четыре требования к проверке UI-агента

Красивая генерация — ещё не готовая feature

Build

Изменение компилируется на нужной платформе, зависимости и сборка не сломаны.

Architecture

Сохранены границы модулей, использованы поддерживаемые библиотеки и соглашения.

Behavior

Loading, error, empty, success и основные взаимодействия совпадают со спецификацией.

Accessibility

Не потеряны семантика, клавиатурные сценарии, контраст и платформенные правила a11y.

У каждого skill есть жизненный цикл

Опубликованный сценарий не должен становиться догмой

Experiment

Команда проверяет гипотезу на типовых задачах.

Evaluate

Регрессия на кейсах и фиксация ручной доводки.

Publish

У владельца есть версия, score и набор проверок.

Review

Исправляем ошибки на основе реального feedback.

Retire

Выводим устаревшие практики из обращения.

Кто создаёт AI-инструменты в команде

Разработчики контрибютят не только в продукты

Инженеры Web, iOS и Android

Пишут skills, плагины, MCP-интеграции и обвязки. Собирают SSOT библиотек и демонстрируют рабочие примеры.

Тимлид и платформа

Помогаем выбрать пилот, провести ревью, согласовать правила и опубликовать решение для других продуктов.

Как команда делится практиками

Инженерная практика должна быть видимой, а не частным хаком

Инженерные встречи

Показываем skill в действии, обсуждаем сбои и спорные решения.

Внутренний marketplace

Публикуем версии и владельцев, чтобы найденным решением мог пользоваться соседний продукт.

Обратная связь

Собираем неудачные случаи как backlog команды и платформы.

Performance review

Учитываем вклад в общие инструменты наряду с доставленными features.

Как я фасилитирую внедрение AI

Не требую «использовать AI». Помогаю улучшать конкретный workflow

Начинаем с трения

Вместе ищем повторяемые задачи и зоны задержек между платформами. Делаем маленький пилот.

Разбираем кейсы

На встречах делимся удачными и неудачными запусками. Автор показывает артефакт и ограничения.

Признаём вклад

Skill, фреймворк, MCP и SSOT считаем инженерным вкладом — учитываем в обсуждении результатов.

Что считаем вкладом в платформу

Не только feature, но и способность команды делать её лучше

Создание

Skill, MCP, плагин, framework или качественный SSOT библиотеки с понятным владельцем.

Переиспользование

Другая команда смогла применить практику, внести улучшение и не зависела от автора.

Качество агента $\neq$ эффект на delivery

Популярный инструмент может не улучшать сроки и нагрузку команды

Capability Quality

Сборка, тесты, UX, соблюдение SSOT и завершение workflow. Насколько надёжен инструмент?

Organizational Outcome

Lead time, rework, reuse и стоимость результата. Стало ли проще работать всей команде?

Что проверяем в наших экспериментах

Это будущие измерения, а не уже полученные цифры

Missing context

Как часто агент спрашивает то, что уже показано в UI-артефактах?

Invented behavior

Как часто он придумывает поведение, которого нет в спецификации?

Traceability

Можно ли связать результат с конкретным design, state или требованием?

Manual rework

Сколько приходится исправлять после генерации и ревью?

Метрики формируют поведение команды

Оптимизация completion rate не должна закреплять
плохие инженерные решения.

Закон Гудхарта

«Когда показатель становится целью, он перестаёт быть хорошим показателем»

Когда метрика — цель

У агента растёт completion rate, но он берётся лишь за простые сценарии. Сложные задачи остаются инженерам.

Что измерять вместе

Проверяем lead time, ручной rework, качество результата и охват сложных задач, а не одну красивую цифру.

Доверие — отдельная инженерная задача

3%

очень сильно доверяют точности AI-ответов

Высокое использование не отменяет необходимости evaluation, human review и надёжного контекста.

Источник: Stack Overflow Developer Survey, 2025

<https://survey.stackoverflow.co/2025/ai>

Красивая метрика может нас обмануть

Не надо поощрять агента обходить сложные случаи

На дашборде

Увеличилось число законченных задач, но агент берёт только простой тип изменений.

В разработке

Сложные UI-состояния и архитектурные исключения возвращаются людям как ручной rework.

Marketplace меняет практики

Поиск, статусы и оценки формируют негласный стандарт

Ranking

Первый результат чаще выбирают. Его архитектура начинает распространяться по компании.

Certification

Verified воспринимают как рекомендацию. Значит, статус должен иметь срок актуальности.

Evaluation

То, что измеряем, становится рабочим определением качества.

Metrics

То, что превращаем в KPI, команды начинают оптимизировать — не всегда к лучшему.

Универсальный skill — не всегда выигрыш

Reuse не должен заставлять три платформы игнорировать свои контексты

В отчёте

Один сценарий используют всё чаще. Кажется, практику удалось масштабировать.

В коде

Команды добавляют исключения, переписывают результат и обходят чужую архитектуру.

Что делать

Сужать область применения, создавать специализированные варианты и пересматривать оценки.

ЗАНОВО

Feedback должен менять не только модель, но и нашу инженерную практику.

Неудачный запуск — полезный материал

Сигнал

Собираем входные данные, ошибку, ручную доработку и ожидаемый результат.

Причина

Отделяем сбой модели от нехватки инструмента, контекста и неверного стандарта.

Новая версия

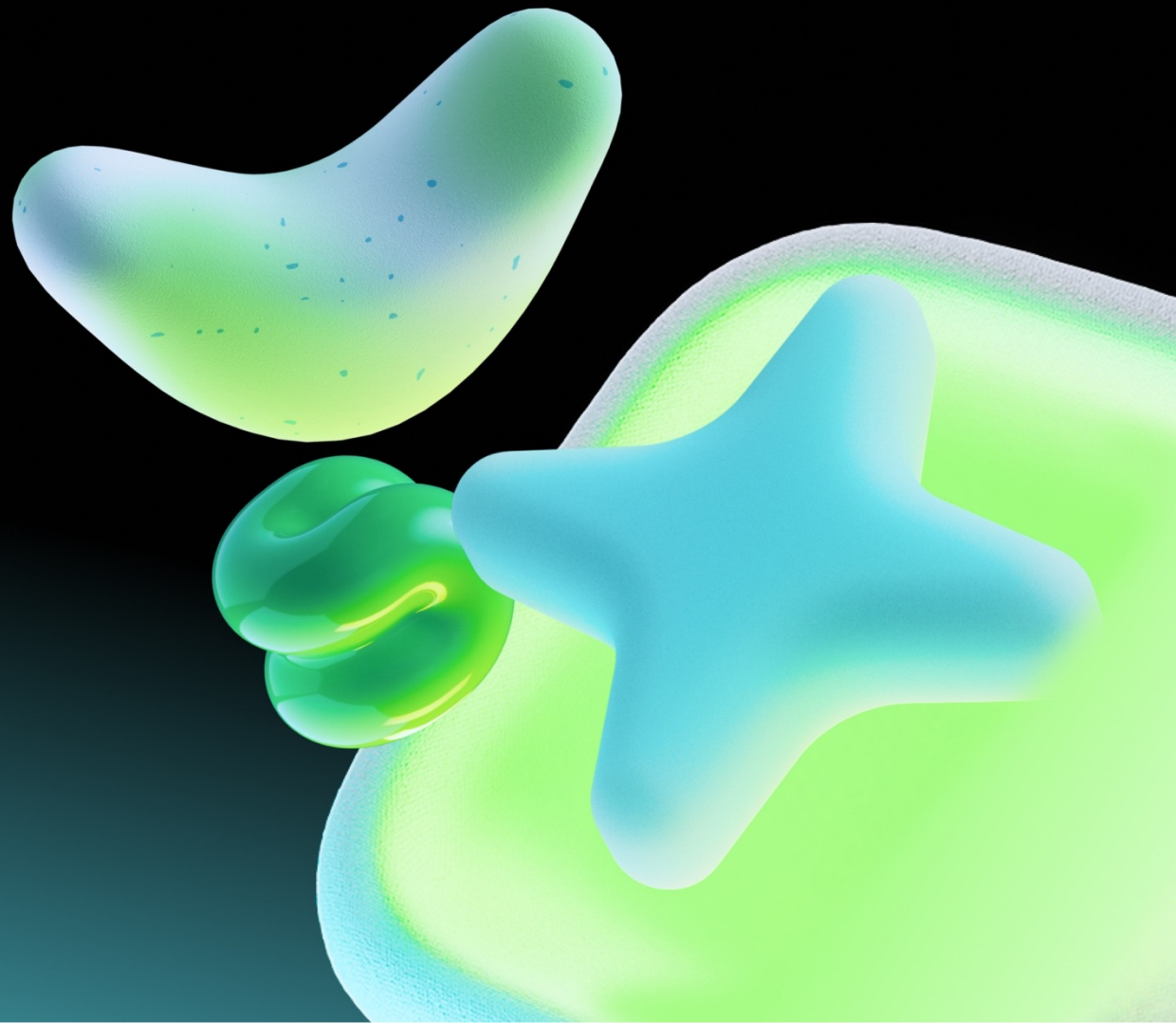
Обновляем нужный слой и прогоняем regression cases до повторной публикации.

Пять причин одной ошибки

Начинаем с симптома — а исправлять можем разные слои

Model

Модель неверно поняла требование или ошиблась в рассуждении. Нужны новые тестовые случаи.



Пять причин одной ошибки

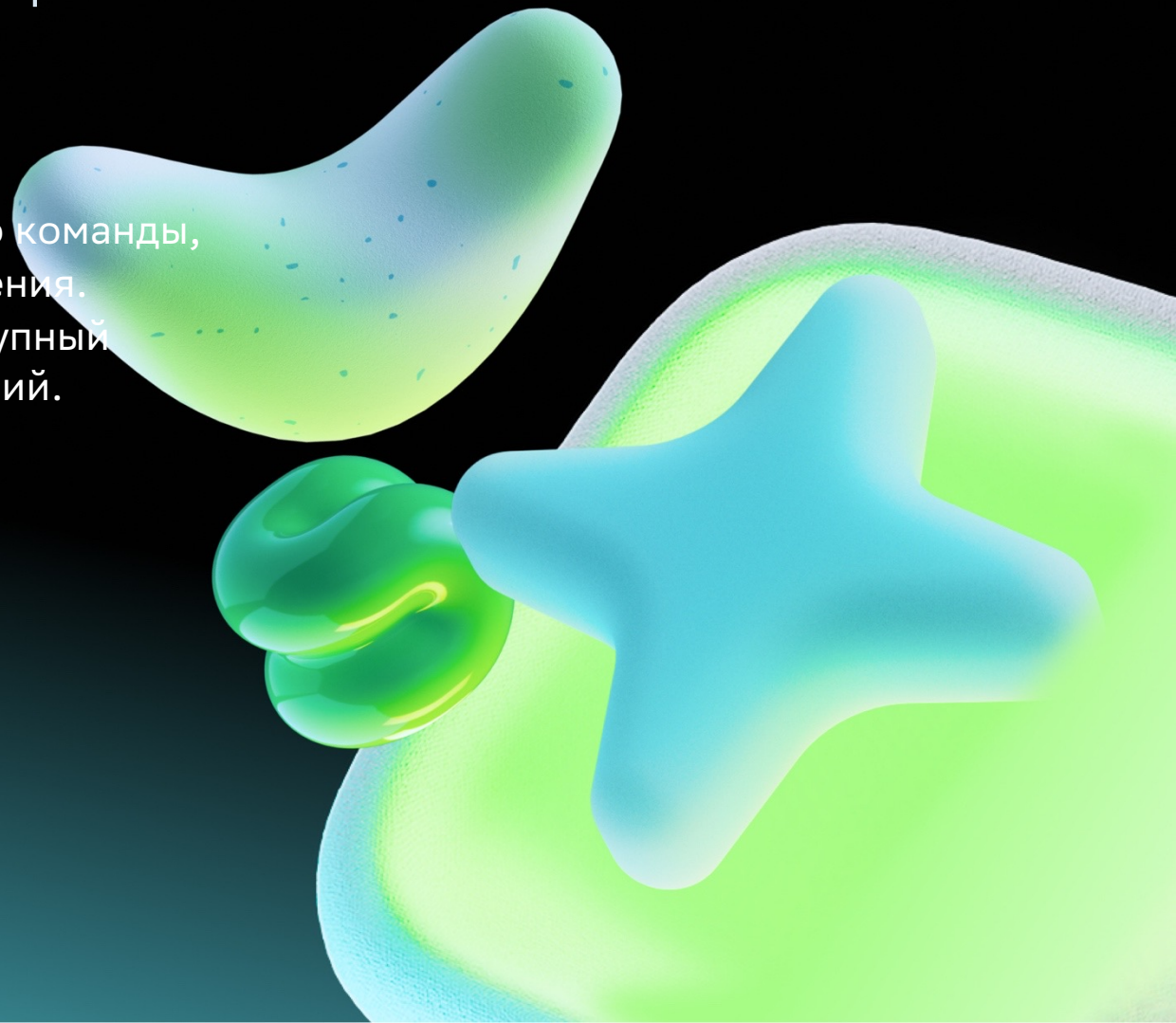
Каждая новая карточка — отдельная гипотеза о причине сбоя

Model

Модель неверно поняла требование или ошиблась в рассуждении. Нужны новые тестовые случаи.

Tool

Агенту не хватило команды, MCP или разрешения. Исправляем доступный интерфейс действий.



Пять причин одной ошибки

Каждая новая карточка — отдельная гипотеза о причине сбоя

Model

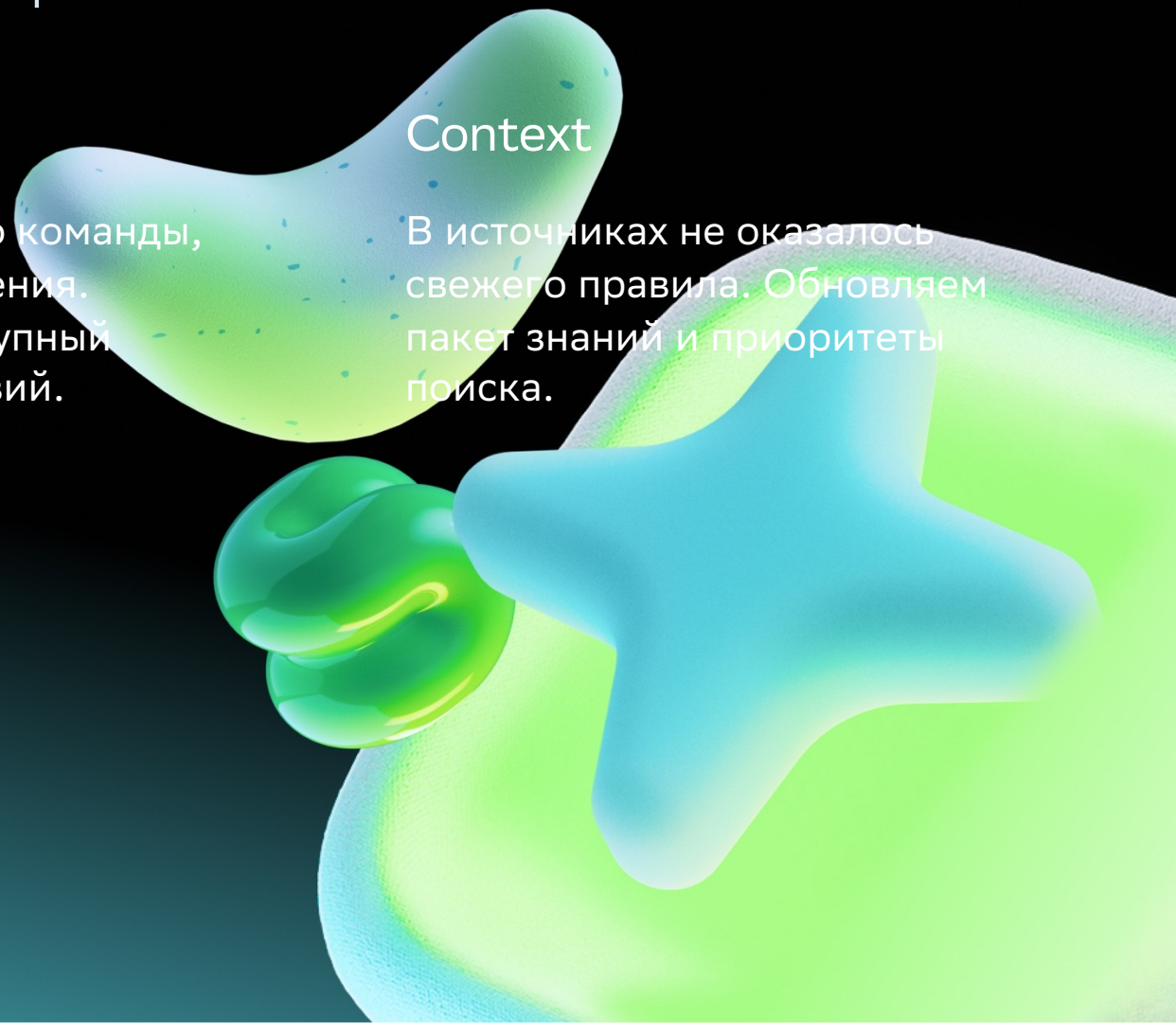
Модель неверно поняла требование или ошиблась в рассуждении. Нужны новые тестовые случаи.

Tool

Агенту не хватило команды, MCP или разрешения. Исправляем доступный интерфейс действий.

Context

В источниках не оказалось свежего правила. Обновляем пакет знаний и приоритеты поиска.



Пять причин одной ошибки

Каждая новая карточка — отдельная гипотеза о причине сбоя

Model

Модель неверно поняла требование или ошиблась в рассуждении. Нужны новые тестовые случаи.

Tool

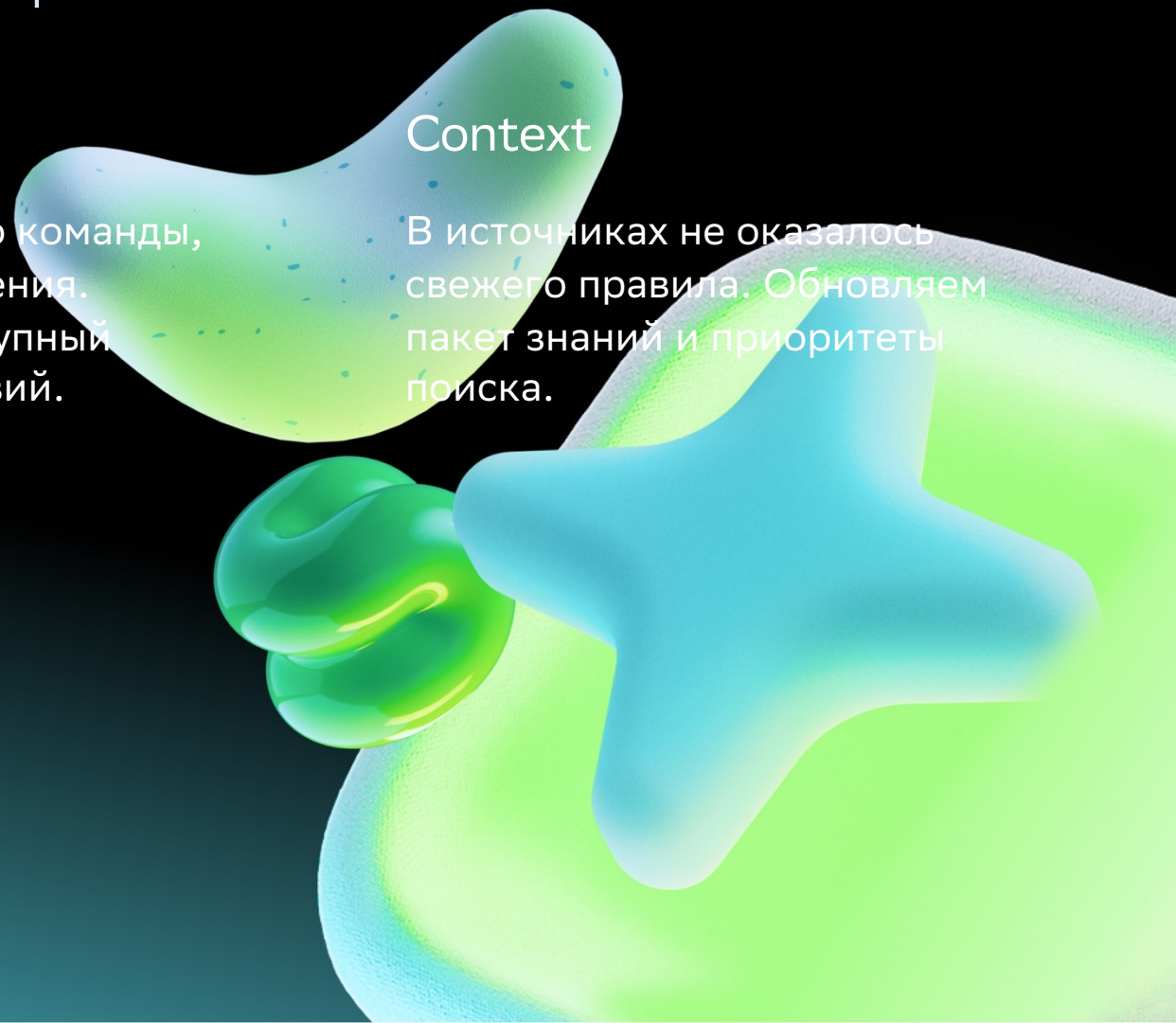
Агенту не хватило команды, MCP или разрешения. Исправляем доступный интерфейс действий.

Context

В источниках не оказалось свежего правила. Обновляем пакет знаний и приоритеты поиска.

Interface

Неверно сформулирован intent или выбран workflow. Улучшаем постановку задачи и UX.



Пять причин одной ошибки

Каждая новая карточка — отдельная гипотеза о причине сбоя

Model

Модель неверно поняла требование или ошиблась в рассуждении. Нужны новые тестовые случаи.

Tool

Агенту не хватило команды, MCP или разрешения. Исправляем доступный интерфейс действий.

Context

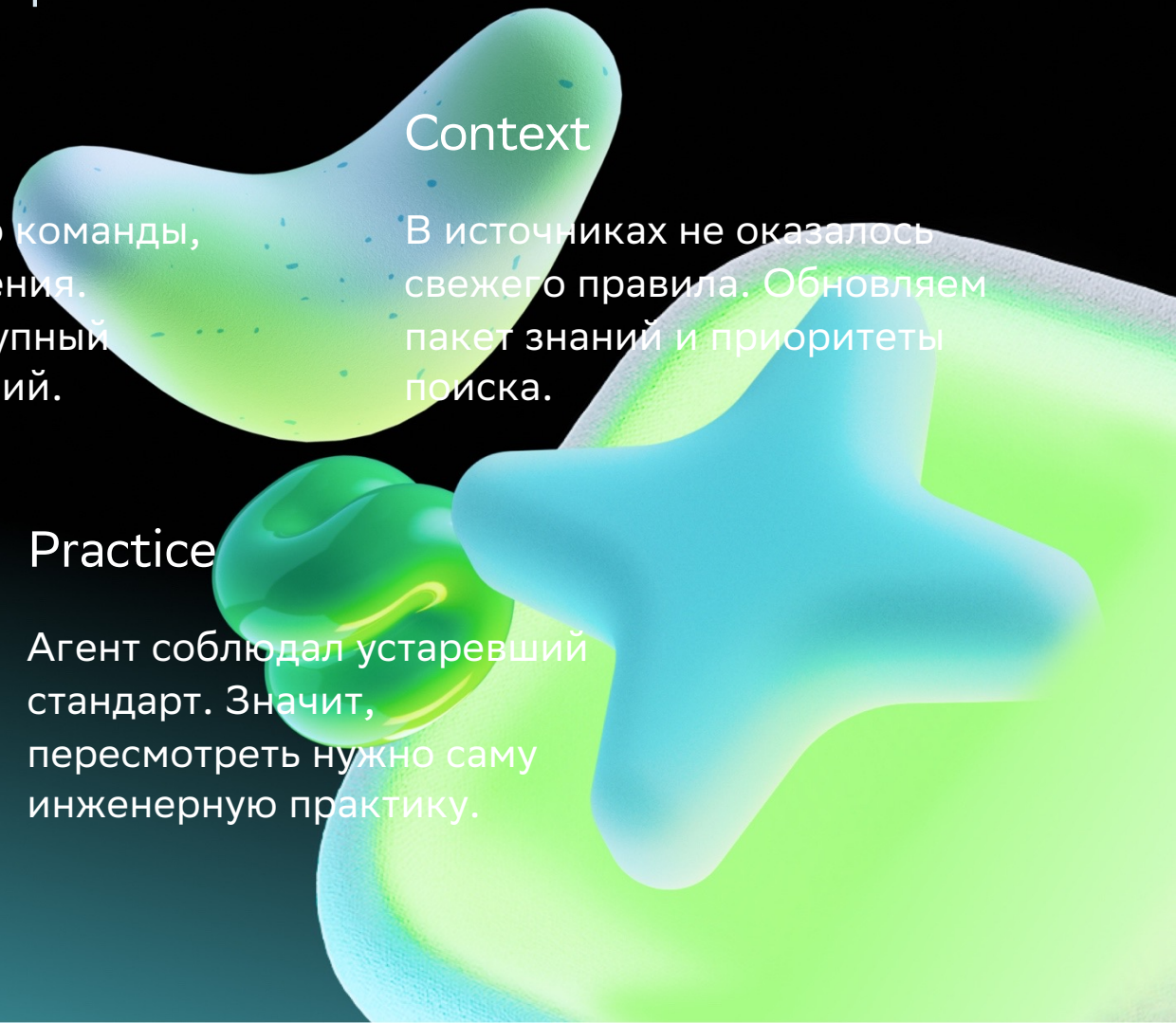
В источниках не оказалось свежего правила. Обновляем пакет знаний и приоритеты поиска.

Interface

Неверно сформулирован intent или выбран workflow. Улучшаем постановку задачи и UX.

Practice

Агент соблюдал устаревший стандарт. Значит, пересмотреть нужно саму инженерную практику.





У контекста должен быть срок
пересмотра

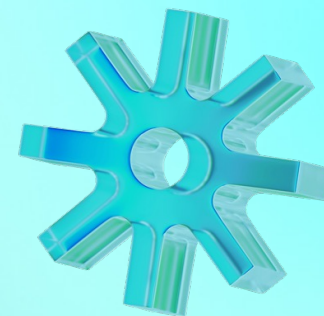
Устаревшие SSOT, проверки и rankings способны закреплять
неправильную практику.



Команда
Сбера

Контекст — актив

Инженер будущего умеет делать знания команды точными, доступными и проверяемыми.



Спасибо!

Вопросы и обсуждение

Николай Константинов

Context Engineering · Web · iOS · Android · AI PDLC

