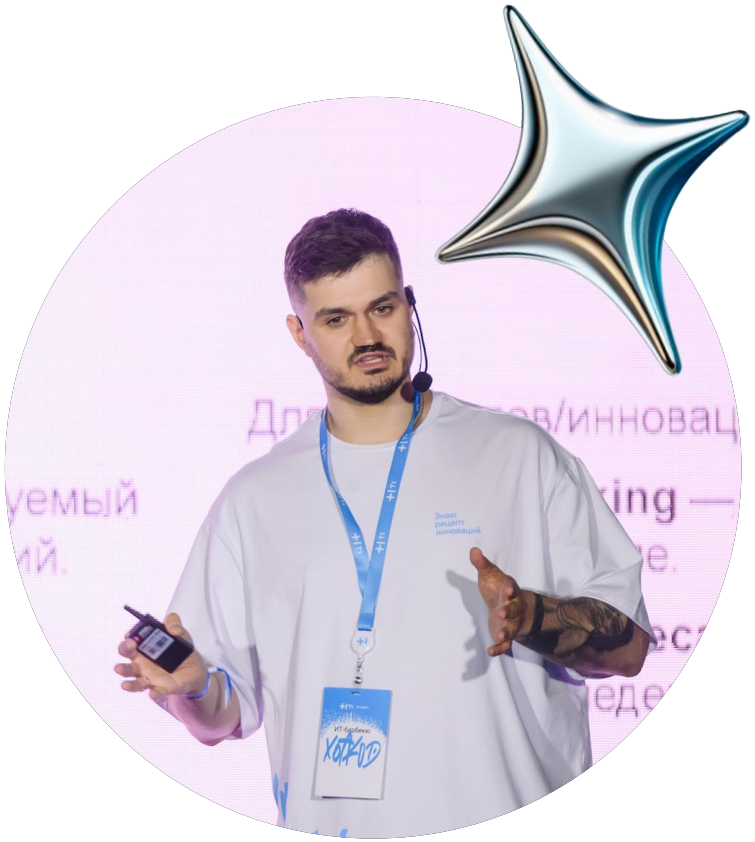
Аналитик, который работает

Практический опыт трансформации роли SA
из документиста в проектировщика
архитектуры и синхронизатора команды

PVS-Studio +1

О спикере



Владимир Бурмистров

Главный системный аналитик
ИТ-Холдинг Т1, проект Dion

+ Порядок в документации
+ Ускорился за счёт ИИ
+ 18 лет в IT, всё ещё не выгорел
+ Умею пользоваться поиском
+ Преподаватель и автор курсов
+ Прошёл путь от автоматизации бухгалтерии и производств до финтеха

PVS-Studio

Филипп Хандельянц

Руководитель разработки статических анализаторов

- 3 года был программистом в отделе разработки статического анализатора C и C++
- Потом 5 лет руководил этим отделом
- Сейчас координирую работу всех отделов разработки
- C++ не отпускает до сих пор, даже курс лекций про него случайно сделал
- В конце 2025 года ВНЕЗАПНО узнал, что я ещё и немного системный аналитик

PVS-Studio +I

Формат встречи

- ~50 минут доклада + секция Q&A
- Вопросы можно задавать в чате

- Будет приз за самый лучший вопрос


Дамы и господа, пристегните ремни

Вы аналитик

Вы аналитик

Дано:

- 8 встреч в день (некоторые по 30 минут)

Вы аналитик

Дано:

- 8 встреч в день (некоторые по 30 минут)
- Документы, которые никто не читает

Вы аналитик

Дано:

- 8 встреч в день (некоторые по 30 минут)
- Документы, которые никто не читает
- Чувство бесполезности

Вы аналитик

Дано:

- 8 встреч в день (некоторые по 30 минут)
- Документы, которые никто не читает
- Чувство бесполезности
- Разработчики не понимают спеки

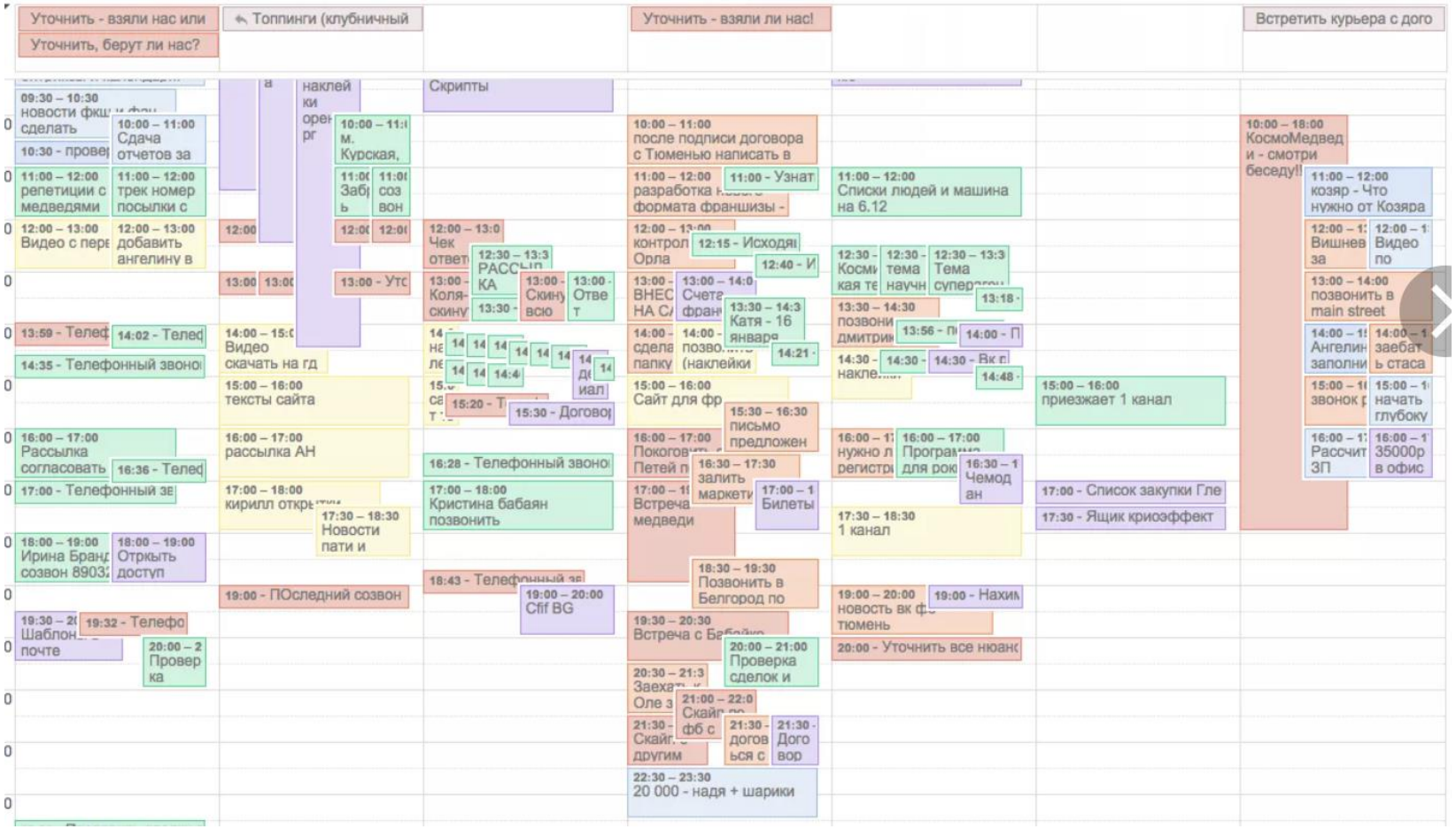
Вы аналитик

Дано:

- 8 встреч в день (некоторые по 30 минут)
- Документы, которые никто не читает
- Чувство бесполезности
- Разработчики не понимают спеки

Знакомо? Значит, вы пришли в правильное место!

Проблема



Аналитик становится дорогим «секретарём» — весь день на встречах, никакого влияния на результат

А что дальше?

- Плохая документация на проекте

А что дальше?

- Плохая документация на проекте
- Недовольная команда

А что дальше?

- Плохая документация на проекте
- Недовольная команда
- Не видно влияния на проект

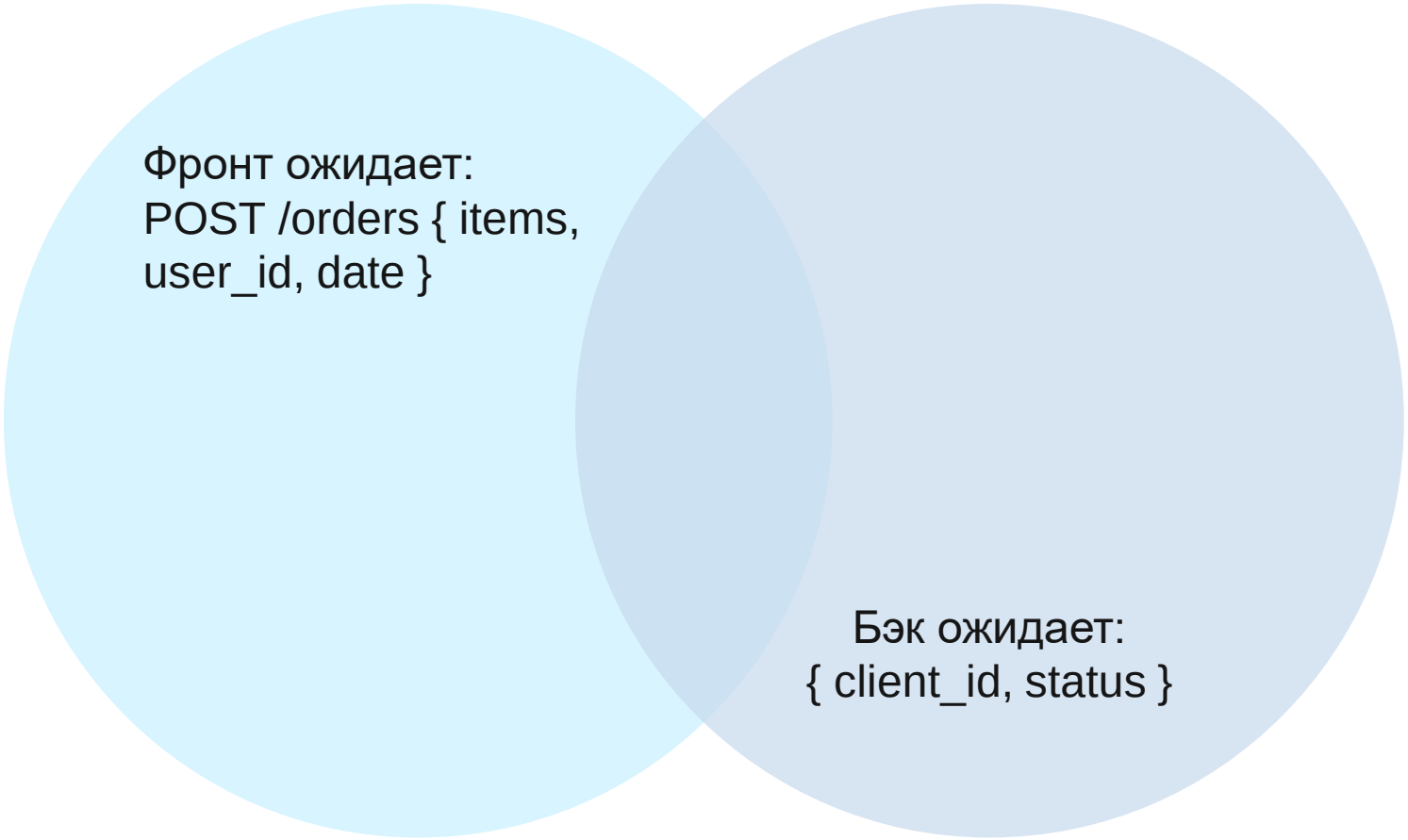
Случай из жизни (Т1)

Расскажу, как делали календарь и сломались на межкомандном взаимодействии.

А я чувствовал себя:

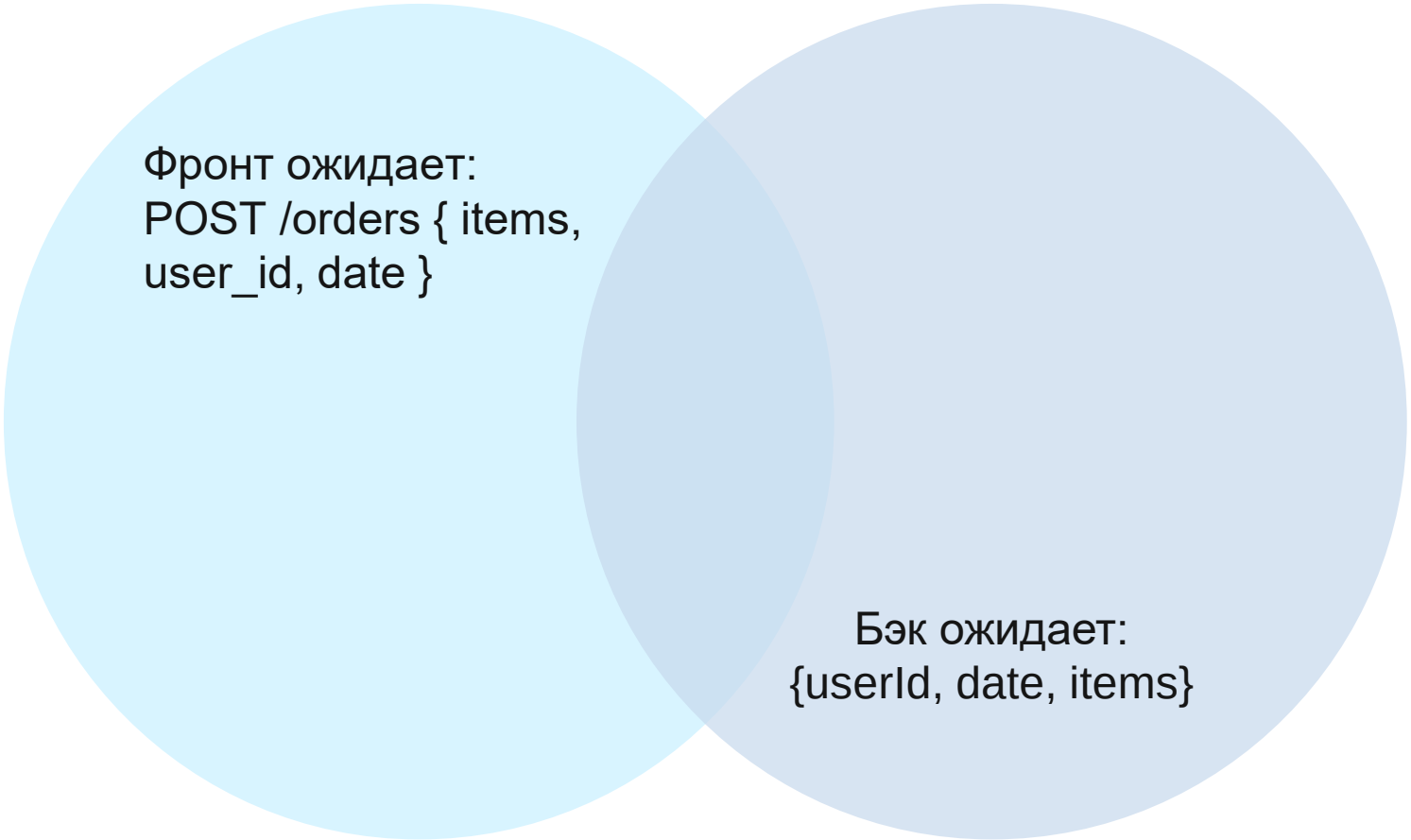
- тупым,
- ненужным,
- меня уволят.

Фронт и бэк сами всё решат



В конце спринта:
фронт и бэк друг друга не понимают → пятница вечером → переделки → выходные

Фронт и бэк сами всё решат



В конце спринта:
фронт и бэк друг друга не понимают → пятница вечером → переделки → выходные

Фронт и бэк сами всё решат

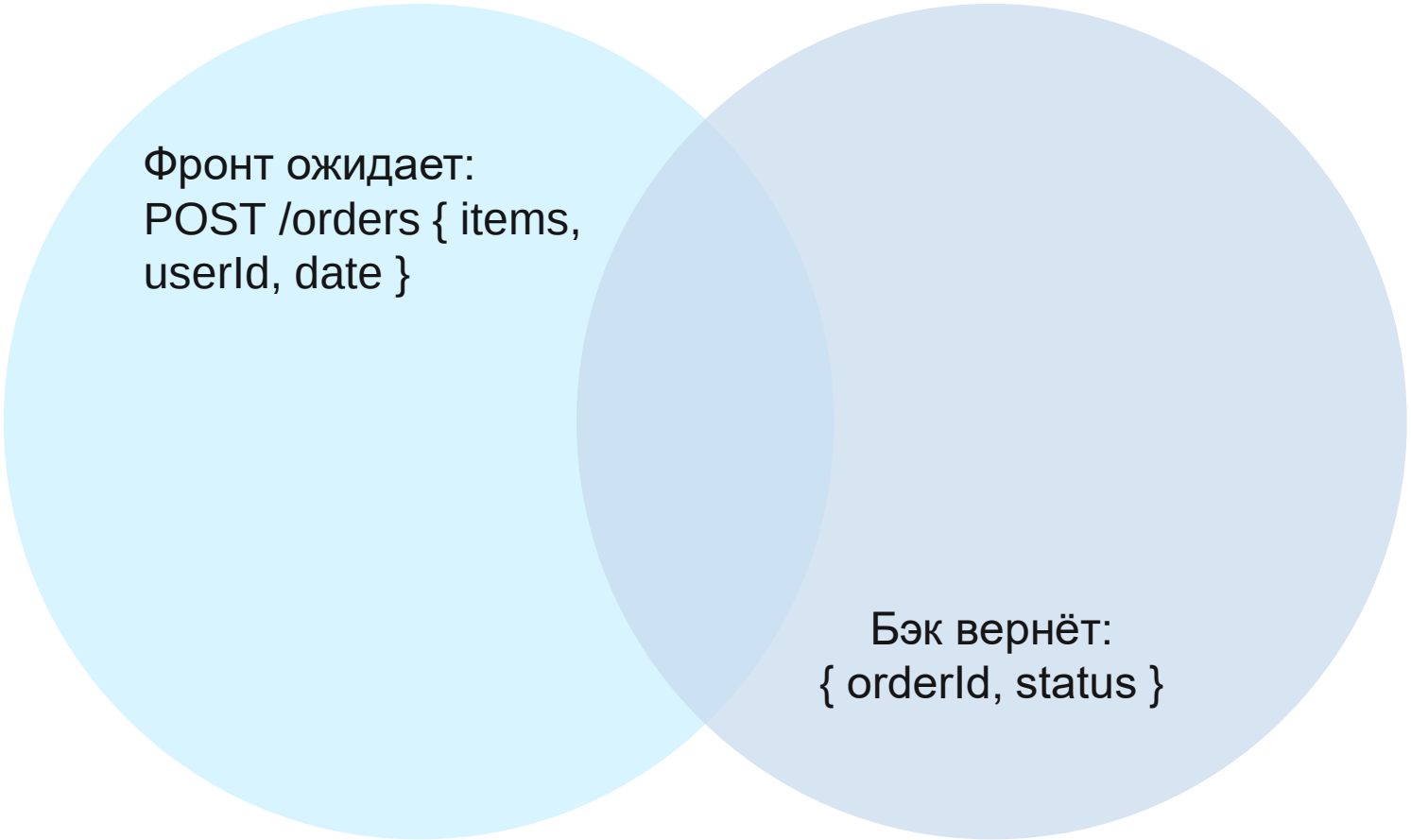
Camel case

```
{
    "firstName": "John",
    "lastName": "Smith",
    "email": "john.smith@example.com",
    "createdAt": "2021-02-20T07:20:01",
    "updatedAt": "2021-02-20T07:20:01",
    "deletedAt": null
}
```

Snake case

```
{
    "first_name": "John",
    "last_name": "Smith",
    "email": "john.smith@example.com",
    "created_at": "2021-02-20T07:20:01",
    "updated_at": "2021-02-20T07:20:01",
    "deleted_at": null
}
```

Фронт и бэк сами всё решат



- **Баги интеграции:**
5-7 критических за спринт
- **Время на переделки:**
30-40% спринта
- **Документация:**
Пишется месяц спустя, никто не читает

Возможно, что это какой-то мой уникальный опыт…

Случай из жизни (PVS-Studio)

Случай из жизни (PVS-Studio)

- Произошёл в момент портирования на Linux

Случай из жизни (PVS-Studio)

- Произошёл в момент портирования на Linux
- В продукте до этого была утилита конвертации отчётов PlogConverter

Случай из жизни (PVS-Studio)

- Произошёл в момент портирования на Linux
- В продукте до этого была утилита конвертации отчётов PlogConverter
- После завершения работ в продукте стало на одну утилиту конвертации больше
- Функционал – почти тот же*

Случай из жизни (PVS-Studio)

- Произошёл в момент портирования на Linux
- В продукте до этого была утилита конвертации отчётов PlogConverter
- После завершения работ в продукте стало на одну утилиту конвертации больше
- Функционал – почти тот же*
- Назвали незамысловато – plog-converter

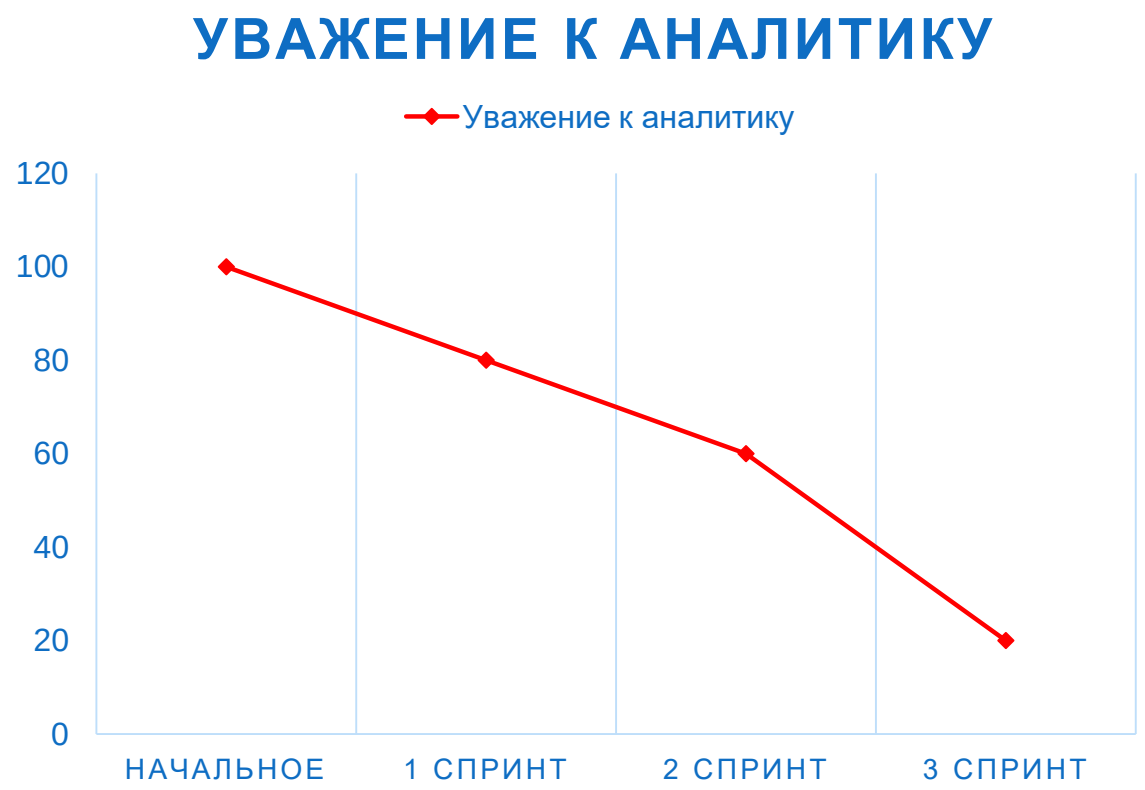
Аналитик либо проектирует систему, либо его не нужно содержать

Аналитик либо проектирует систему, либо его не нужно содержать

Выбирайте первое — результат очевиден

Аналитик либо проектирует систему, либо его не нужно содержать

Выбирайте первое — результат очевиден



Вигерс фигни не скажет

«Ошибки, допущенные на этапе проектирования — это самые болезненные ошибки, потому что для их исправления требуется перезапуск всего производственного процесса»

И Макконнелл тоже не скажет

И Макконнелл тоже не скажет

«Общий принцип прост: исправлять ошибки нужно как **можно раньше**. Чем дольше дефект сохраняется в пищевой цепи разработки ПО, тем больше вреда он приносит на следующих этапах»

А что если перевернуть процесс?

- ~~Боль~~
- ~~Хаос~~

Пример моей команды (Т1)

1. **Аналитик пишет спецификацию до спринта** — подробная техническая архитектура, контракты API, состояния данных
2. **Арх совет** — согласование архитектуры с корп. архитектором
3. **PBR** — показ команде проработанной задачи и разбор корнер-кейсов
4. **Фронт и бэк синхронизируются параллельно** — используют одну спецификацию как договор, разрабатывают в один спринт
5. **Аналитик занимается следующими задачами** — иногда, конечно, отвечает на вопросы разрабов и тестеров
6. **Результат:** в конце спринта всё работает — минимум багов, полная документация готова сразу

Чего достигли?

- Ускорение спринтов на 15-30% (меньше переделок)
- 🦾 Снижение критических багов на 70-90%
- 📋 Документация появляется вместе с кодом, а не месяцы спустя
- 💪 Техлиды и разработчики просят хорошего аналитика (вместо раздражения)
- 🎯 Аналитик видит прямое влияние на метрики и результаты

Пример PVS-Studio (разработка новых анализаторов)

Пример PVS-Studio (разработка новых анализаторов)

- C и C++ анализатор написан основателями аж в 2008 году

Пример PVS-Studio (разработка новых анализаторов)

- C и C++ анализатор написан основателями аж в 2008 году
- C# анализатор – pet-проект сотрудника, перекочевал в продукт в 2015 году
- Java анализатор – pet-проект сотрудника, перекочевал в продукт в 2019 году

Пример PVS-Studio (разработка новых анализаторов)

- C и C++ анализатор написан основателями аж в 2008 году
- C# анализатор – pet-проект сотрудника, перекочевал в продукт в 2015 году
- Java анализатор – pet-проект сотрудника, перекочевал в продукт в 2019 году
- Поставлена задача в 2026 году выпустить 3 новых анализатора: для Golang, JS и TS

Пример PVS-Studio (разработка новых анализаторов)

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с СТО и тимлидами/техлидами

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с СТО и тимлидами/техлидами
3. Проводим груминг с командой(-ами) разработки

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с CTO и тимлидами/техлидами
3. Проводим груминг с командой(-ами) разработки
4. Разрабатываем анализатор и плагины для IDE параллельно

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с CTO и тимлидами/техлидами
3. Проводим груминг с командой(-ами) разработки
4. Разрабатываем анализатор и плагины для IDE параллельно
5. Ревьюеры проверяют, что реализация соответствует спецификации

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с CTO и тимлидами/техлидами
3. Проводим груминг с командой(-ами) разработки
4. Разрабатываем анализатор и плагины для IDE параллельно
5. Ревьюеры проверяют, что реализация соответствует спецификации
6. Аналитик отвечает на появившиеся вопросы разработчиков

Пример PVS-Studio (разработка новых анализаторов)

1. Аналитик пишет спецификацию — архитектура решения, API, входные и выходные данные
2. Согласуем спецификацию с CTO и тимлидами/техлидами
3. Проводим груминг с командой(-ами) разработки
4. Разрабатываем анализатор и плагины для IDE параллельно
5. Ревьюеры проверяют, что реализация соответствует спецификации
6. Аналитик отвечает на появившиеся вопросы разработчиков

Важно: уточняем спецификацию по ОС на всех этапах

Чего достигли? (отзывы коллег)

Чего достигли? (отзывы коллег)

- ✕ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать
- 🔎 Оценка объёма работ стала проще

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать
- 🔎 Оценка объёма работ стала проще
- 🙈 Стало меньше внутренних обсуждений и больше времени на реализацию

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать
- 🔍 Оценка объёма работ стала проще
- 🙊 Стало меньше внутренних обсуждений и больше времени на реализацию
- 📉 Уменьшили время на ревью: решение сверяется согласно спецификации

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать
- 🔍 Оценка объёма работ стала проще
- 🙈 Стало меньше внутренних обсуждений и больше времени на реализацию
- 📉 Уменьшили время на ревью: решение сверяется согласно спецификации
- ☑️ Функционал стандартизирован, не появляются странные решения в продукте

Чего достигли? (отзывы коллег)

- ❌ Отказ от концепции «кто-то накидает простецкий прототип, а отдел потом доведёт до релиза»
- 🎓 Отделы знают о существовании огромного количества сценариев, которые надо поддержать
- 🔍 Оценка объёма работ стала проще
- 🙊 Стало меньше внутренних обсуждений и больше времени на реализацию
- 📉 Уменьшили время на ревью: решение сверяется согласно спецификации
- ☑️ Функционал стандартизирован, не появляются странные решения в продукте
- 😮💨 Интроверты выдохнули

Вигерс фигни не скажет

«Требования должны быть **достаточно хорошими**, чтобы разработка могла продолжаться с **приемлемым уровнем риска**»

Жизненный цикл проекта

Вспоминаем, из чего у нас состоит итерация

Жизненный цикл проекта

Жизненный цикл проекта

Новая инициатива — SC — BRS — SRS — Разработка

Жизненный цикл проекта

Новая инициатива

Большая задача

Мы собрались делать что-то глобальное и не очень

Жизненный цикл проекта

Solution Concept

Большая задача

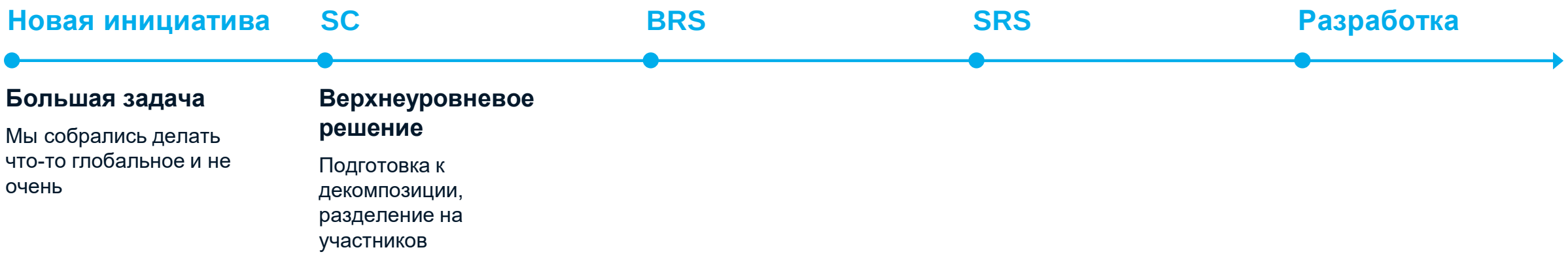
Мы собрались делать что-то глобальное и не очень

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

Жизненный цикл проекта

Solution Concept



Для кого: PO, Архитекторы, Технические лиды, Команды

Жизненный цикл проекта

Solution Concept

Для кого: PO, Архитекторы, Технические лиды, Команды

Когда писать:

- ✅ Всегда при новой крупной инициативе
- ✅ Вовлечены несколько команд
- ✅ Сложная/кросс-функциональная фича
- ⚠ По ситуации для технических задач

Жизненный цикл проекта

Solution Concept

Большая задача

Мы собрались делать что-то глобальное и не очень

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

Чек-лист успеха:

- Цели и бизнес-контекст (зачем это всё?)
- Краткое описание решения
- HLA (High Level Architecture): диаграммы C1 и C2
- Список требований (User Stories/Use Cases) — не детализация!
- Ограничения и допущения решения
- ADR (Architecture Decision Records) — по необходимости

Для кого: PO, Архитекторы, Технические лиды, Команды

Когда писать:

- ✅ Всегда при новой крупной инициативе
- ✅ Вовлечены несколько команд
- ✅ Сложная/кросс-функциональная фича
- ⚠ По ситуации для технических задач

Жизненный цикл проекта

Solution Concept

Новая инициатива — **SC** — **Разработка**

Большая задача

Мы собрались делать что-то глобальное и не очень

Верхнеуровнев… решение

Подготовка к декомпозиции, разделение на участников

…ха:
…нес-контекст (зачем это всё?)
…исание решения
…Level Architecture): диаграммы C1 и

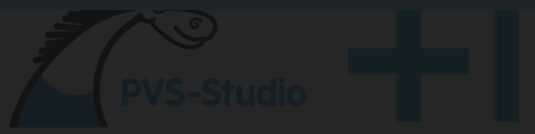
- Список требований (User Stories/Use Cases) — не детализация!
- Ограничения и допущения решения
- …n Records) — по необходимости

Для кого: PO, Архитекторы, Технич…

Когда писать:

- ✅ Всегда при новой крупной инициативе
- ✅ Вовлечены несколько команд
- ✅ Сложная/кросс-функциональная фича
- ⚠ По ситуации для технических задач

TechMeetup #7 System analysis



Жизненный цикл проекта

Business Requirements Specification

Новая инициатива

Большая задача

Мы собрались делать что-то глобальное и не очень

SC

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

BRS

Как это для пользователя

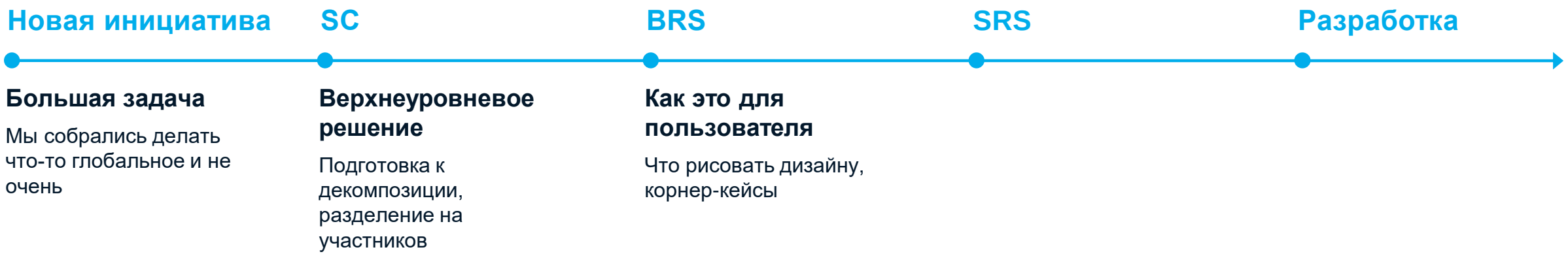
Что рисовать дизайну, корнер-кейсы

SRS

Разработка

Жизненный цикл проекта

Business Requirements Specification



Для кого: PO, Команда разработки, Тестировщики, Дизайнеры

Жизненный цикл проекта

Business Requirements Specification

Большая задача

Мы собрались делать что-то глобальное и не очень

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

Как это для пользователя

Что рисовать дизайну, корнер-кейсы

Для кого: PO, Команда разработки, Тестировщики, Дизайнеры

Когда писать:

☑ Всегда, если это пользовательская фича

✕ Не нужен для чисто технических задач

Жизненный цикл проекта

Business Requirements Specification

Новая инициатива

Большая задача

Мы собрались делать что-то глобальное и не очень

SC

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

BRS

Как это для пользователя

Что рисовать дизайну, корнер-кейсы

Чек-лист успеха:

- Глоссарий (все термины определены)
- User Story в формате: As [role], I want [action], So that [benefit]
- Проблематика: AS IS / TO BE
- Acceptance Criteria в формате Given/When/Then
- Анализ конкурентов (опционально)
- Нефункциональные требования:
  - Требования к UI (ссылка на макеты)
  - Требования к производительности
  - Требования к безопасности

Для кого: PO, Команда разработки, Тестировщики, Дизайнеры

Когда писать:

☑ Всегда, если это пользовательская фича

✕ Не нужен для чисто технических задач

Жизненный цикл проекта

Software Requirements Specification

Большая задача

Мы собрались делать что-то глобальное и не очень

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

Как это для пользователя

Что рисовать дизайну, корнер-кейсы

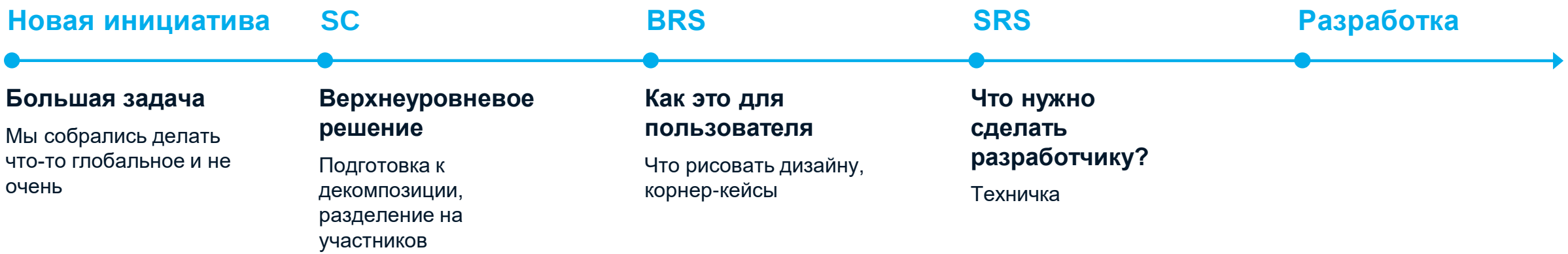
Что нужно сделать разработчику?

Техничка

Для кого: РАЗРАБОТЧИКИ (главные потребители!)

Жизненный цикл проекта

Software Requirements Specification



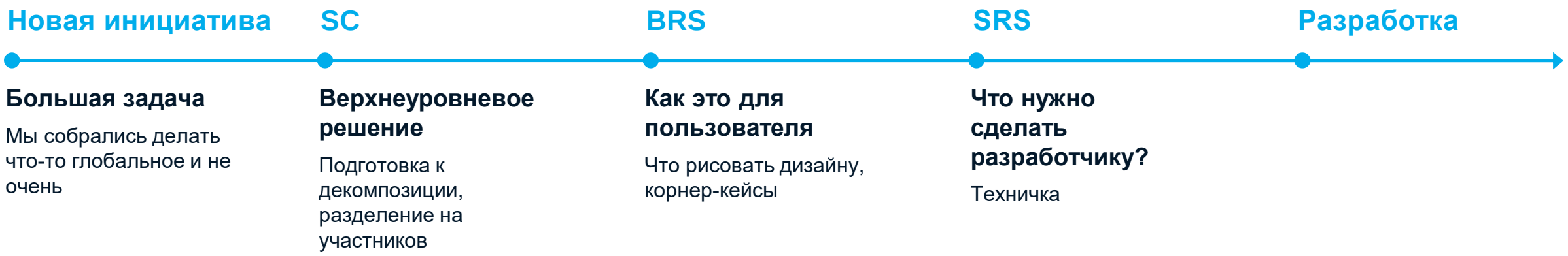
Для кого: РАЗРАБОТЧИКИ (главные потребители!)

Когда писать:

☑ ВСЕГДА

Жизненный цикл проекта

Software Requirements Specification



Для кого: РАЗРАБОТЧИКИ (главные потребители!)

Когда писать:

☑ ВСЕГДА

Золотое правило:

Всегда начинайте SRS с **КРАТКОГО ОПИСАНИЯ РЕШЕНИЯ** (3-5 предложений)!

Без него разработчик смотрит на 50 страниц спеки и думает «ой бл*...»

Техническая спецификация

Настройки продуктов PVS-Studio

Жизненный цикл проекта

Software Requirements Specification

Новая инициатива → **SC** → **BRS**

Большая задача

Мы собрались делать что-то глобальное и не очень

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

Как это для пользователя

Что рисовать дизайну
корнер-кейсы

Для кого: РАЗРАБОТЧИКИ (главные потребители!)

Когда писать:

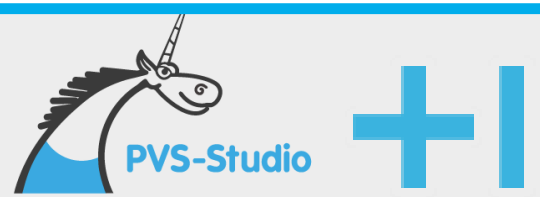
☑ ВСЕГДА

Золотое правило:

Всегда начинайте SRS с **КРАТКОГО ОПИСАНИЯ РЕШЕНИЯ** (3-5 предложений)!
Без него разработчик смотрит на 50 страниц спеки и думает «ой бл*...»

Чек-лист успеха:

- Краткое описание решения (что меняем: 3 API, 2 таблицы, 1 новый сервис)
- Таблица изменений BE/FE
- Диаграммы:
  - C2 (если изменяется архитектура)
  - Sequence diagrams (сценарии взаимодействия)
- Логическая модель данных (ER-диаграммы, если изменяется БД)
- Описание методов: REST/gRPC/WSS/Kafka/GraphQL…
- Маппинг макетов UI и вызовов API
- Нефункциональные требования:
  - Производительность
  - Аудит
  - Информационная безопасность
  - Продуктовые метрики



Жизненный цикл проекта

Software Requirements Specification

Новая инициатива · SC · ...

Большая ... ча
Мы собрались ... что-то глобально ... очень

...
Подготовка к декомпозиции, разделение на участни...

Как это для пользователя
Что рисова...

Для кого: ... (главные потребители!)

Когда писать:
☑ ВСЕГДА

Чек-лист успеха:
- Краткое описание решения (что меняем: 3 API, 2 таблицы, 1 новый ...)
- Таблица ...
- Диаграммы:
  - C2 (если изменяется архитектура)
  - Sequence diagrams (сценарии взаимодействия)
- Логическая модель данных (ER-диаграммы, если изменяется БД)
- Описание методов: REST/gRPC/WSS/Kafka/GraphQL…
- Маппинг макетов UI и вызовов API
- Нефункциональные требования:
  - Производительность
  - Аудит
  - Информационная безопасность
  - Продуктовые метрики

Золотое п...
Всегда начи... SRS с **КРАТКОГО ОПИСАНИЯ РЕШЕНИЯ** (3-5 предложений)!
Без него разработчик смотрит на 50 страниц спеки и думает «ой бл*...»

Заполняйте только применимые разделы

Жизненный цикл проекта

Разработка

Новая инициатива

Большая задача

Мы собрались делать что-то глобальное и не очень

SC

Верхнеуровневое решение

Подготовка к декомпозиции, разделение на участников

BRS

Как это для пользователя

Что рисовать дизайну, корнер-кейсы

SRS

Что нужно сделать разработчику?

Техничка

Разработка

Мы все

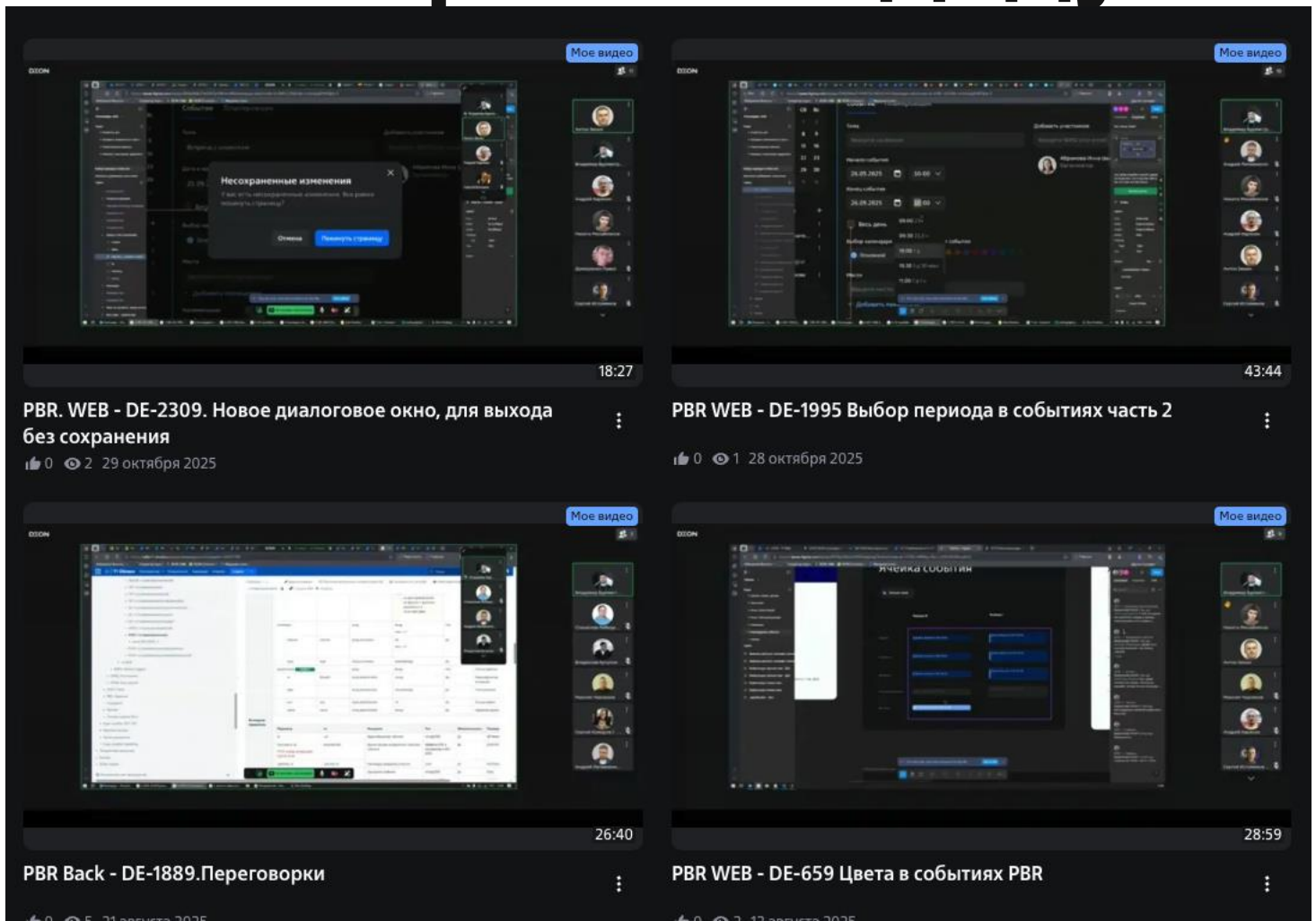
Придут, конечно, разрабы, тестировщики… ну или сделать для них созвон

Много документов

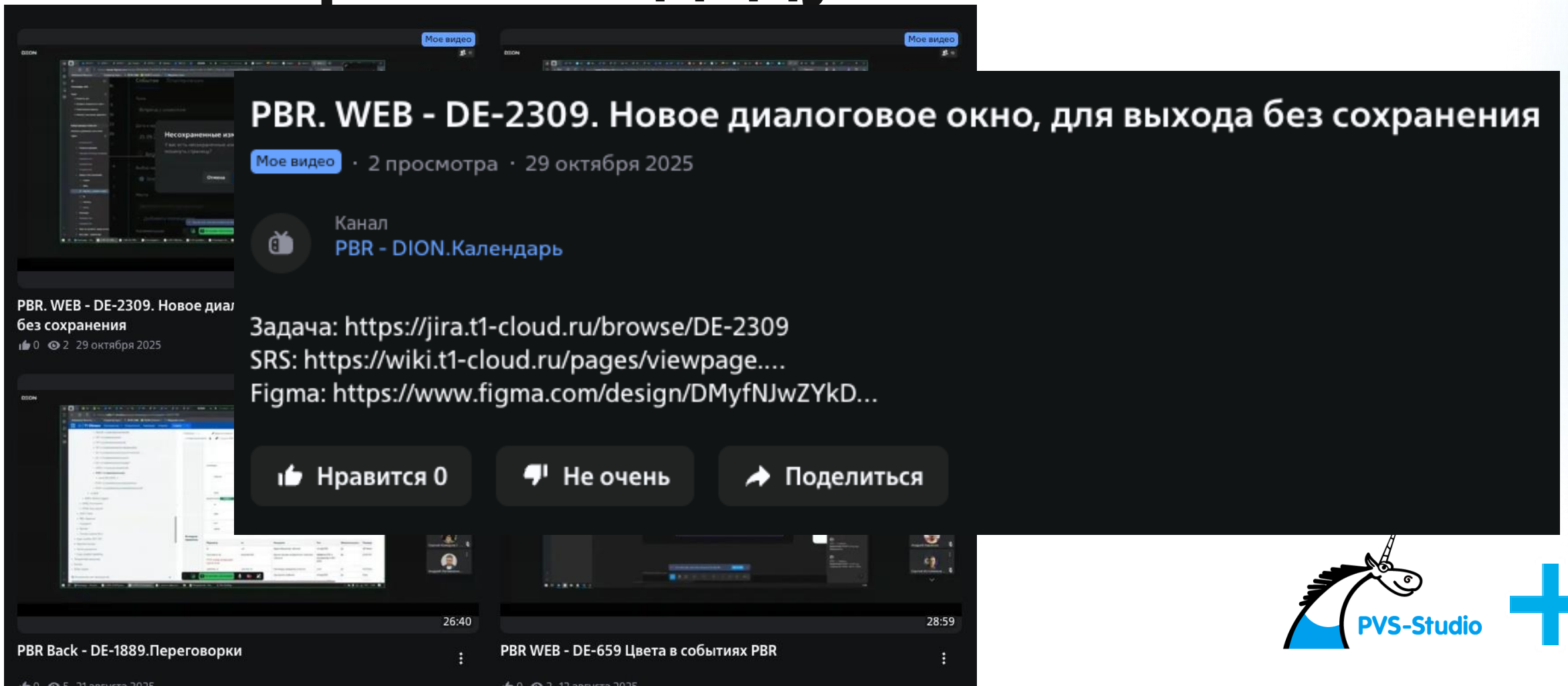
Много документов
Разработка не будет читать

Много документов
Разработка не будет читать
100 вопросов зададут

Много документов
Разработка не будет читать
100 вопросов зададут



Много документов
Разработка не будет читать
100 вопросов зададут



Meeting notes как средство коммуникации в проектной команде

RACI МАТРИЦА ДЛЯ АРТЕФАКТОВ

RACI — это:

- R (Responsible) — Исполнитель
- A (Accountable) — Ответственный за результат
- C (Consulted) — Консультант
- I (Informed) — Информируемый

RACI МАТРИЦА ДЛЯ АРТЕФАКТОВ

RACI — это:

- R (Responsible) — Исполнитель
- A (Accountable) — Ответственный за результат
- C (Consulted) — Консультант
- I (Informed) — Информируемый

Артефакт	Responsible	Accountable	Consulted	Informed
SC	Аналитик, Архитектор	PO, Архитектор	Тех. лиды	Команды
BRS	Аналитик	PO	Дизайнер, Тех. лид	Команда
SRS	Аналитик	Тех. лид	Разработчики	QA, PO
ADR	Архитектор, Тех. лид	Архитектор	Аналитик, Разработчики	PO

Подытог

- Спецификация — это до разработки, а не после
- Каждый артефакт имеет своего потребителя и RACI
- Диаграммы — только если показывают изменения, не просто для красоты

Правило

- Если нет Responsible → не пиши
- Если нет Accountable → не пиши
- Если нет хотя бы одного Consulted → это документ ради документа

Пример документов ради документов

- ❌ SC без решения (просто описание проблемы)
- ❌ ADR без контекста (просто "мы выбрали Redis")
- ❌ BRS без метрик (просто список фич)
- ❌ SRS без примеров (только описание, нет JSON)
- ❌ "Красивая диаграмма текущего состояния" (никто её не обновляет)

Картинка понятнее текста

- c4
- ERD
- Sequence
- State
- BPMN

Картинка понятнее текста

- c4
- ERD
- Sequence
- State
- BPMN

❌ C4 Level 4 (слишком детально, быстро устаревает)
❌ Диаграмма текущего состояния БД (для вики, не для спринта)
❌ UML классы (не нужно SA, это для документирования кода)
❌ "Красивая архитектура" (если не показывает изменения)
❌ Диаграмма, которую никто не обновляет (зачем она нужна?)
❌ Непонятные

Начни сегодня

- Изучи свои документы
- Составь RACI
- Поговори с командой

ЗАКЛЮЧЕНИЕ

Документы ≠ Бюрократия
Документы = Актив (когда они ДЛЯ людей)

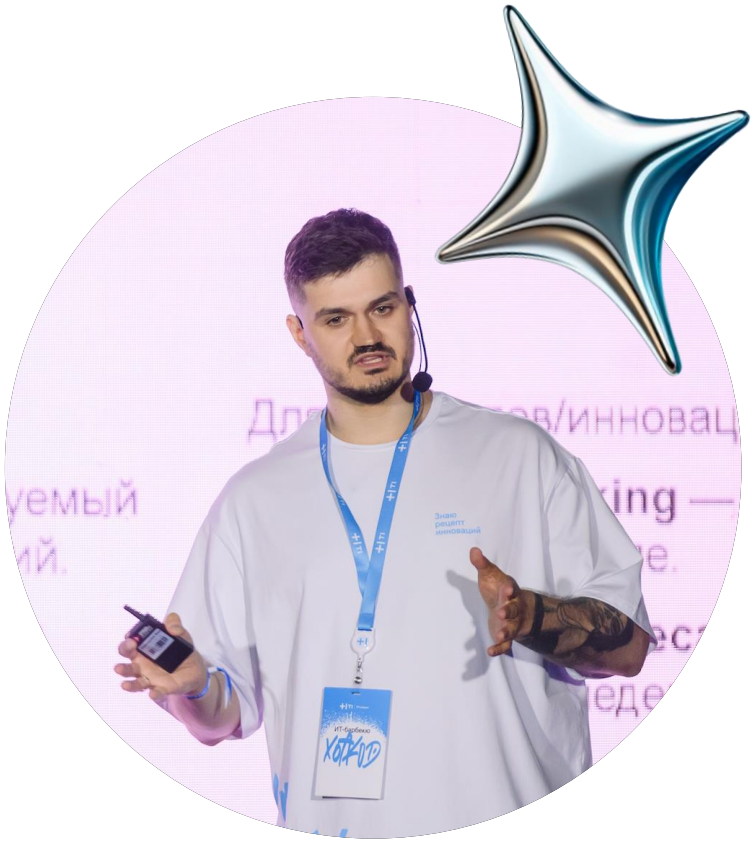
Артефакты = топливо, когда:

- У каждого есть целевая аудитория (RACI)
- Они решают реальные проблемы команды
- Заполняются разумно (здравый смысл > догма)

Делать БОЛЬШЕ, писать МЕНЬШЕ:

- Один хороший документ > три посредственных
- Шаблон = чек-лист, а не повинность
- Документы для людей, а не для процесса

О спикере



Владимир Бурмистров

Главный системный аналитик
ИТ-Холдинг Т1, проект Dion

Тг канал

Личный тг

Карьера в Т1

Диагноз аналитик

Филипп Хандельянц

Руководитель разработки статических анализаторов

Информация о PVS-Studio

Попробовать на 1 месяц

Наша группа в Telegram

PVS-Studio

Спасибо
за внимание

PVS-Studio +1

Информация о PVS-Studio

Попробовать на 1 месяц

Группа PVS в Telegram

Личный тг Фила

Тг канал: Burmistrov - It и около

Личный тг Владимира

Карьера в Т1

Тг канал: Диагноз аналитик