стратоплан
ШКОЛА МЕНЕДЖМЕНТА

SummerCamp 2026: Оргдизайн в эпоху ИИ

Как увидеть поток ценности, найти ограничение и проверить новую конструкцию до реорганизации

Александр Апазиди
24.06.2026

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Представлюсь!

Александр Апазиди

CTO / CIO · C-level опыт · тренер, ментор

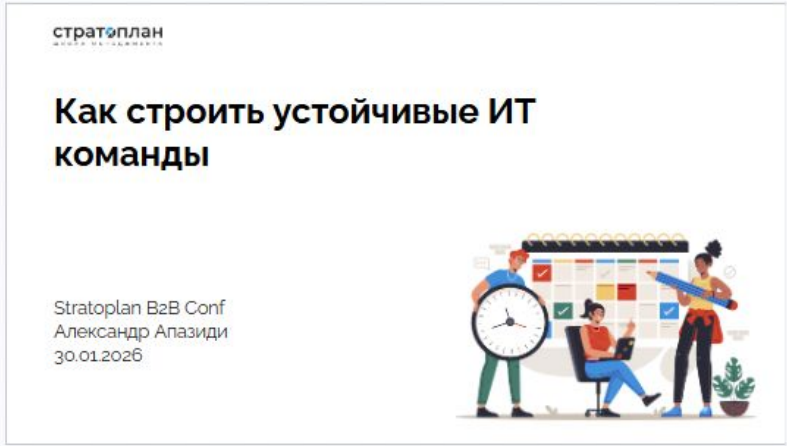
- Более 30 лет в сфере информационных технологий, прошел путь от разработчика и руководителя проектов до ИТ-директора
- Schlumberger, IBM, Nestle, InBev, Volvo, Metro C&C, СИБУР
- Преподаватель в Стратоплан, Яндекс Практикум
- Блог про ИТ менеджмент

https://t.me/apazidi_IT

@apazidi

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Сегодня продолжение прошлых тем



https://t.me/apazidi_IT/83

https://t.me/apazidi_IT/98

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

… и разбор на практике одной организации

Общая идея: берем сырые данные, прогоняем их через ИИ, ищем узкие места, формируем гипотезы улучшения и проверяем их эффект

Сырые следы работы

полный SDLC от запроса до проверки результата

Журнал гипотез

диагноз обновляется после каждого слоя данных

Новый вариант оргдизайна

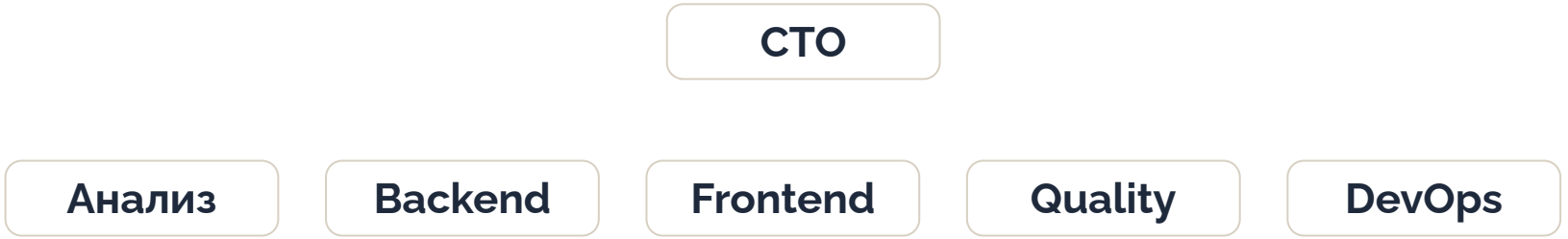
сравним варианты и посчитаем диапазон эффекта

Пакет исходных данных, слайды и инструкции с промптами будут доступны после выступления

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Что производит ваша организация?

Оргчарт обычно не показывает что компания производит

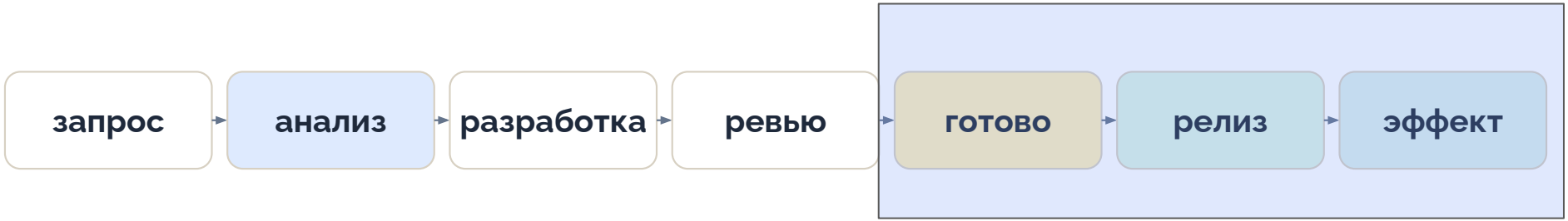


Понятно кто кому подчиняется, но так ли в реальности устроен процесс производства?

Поток ценности зачастую отличается от оргчарта

Поэтому оргдизайн обычно начинается не с прямоугольников с стрелками, но с вопроса: какой результат мы производим и что считается “сделано” ?

Поток ценности зачастую отличается от оргчарта



Поэтому оргдизайн обычно начинается не с прямоугольников с стрелками, но с вопроса: какой результат мы производим и что считается “сделано” ?

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Пробуем ИИ для первого анализа

С помощью ИИ можно сравнительно быстро восстановить реальный поток работы

Для примера я буду использовать пример выгрузки похожей на “сырые” данные из трекера:

- ~100 задач
- Процесс похож на типовой SDLC
  - Получение задачи
  - Аналитика
  - Разработка
  - Ревью
  - QA
  - Релиз в прод
  - Проверка эффекта

С помощью ИИ можно сравнительно быстро восстановить реальный поток работы

CSV не знает оргструктуру, но это следы того, как работа реально прошла через систему

	A	B	C	D	E	F	G	
1	item_id	work_type	risk_class	owner_id	owner_role	requested_at	analysis_started_at	an
2	WI-001	backend	low	DEV-02	Backend	2026-02-14	2026-02-17	
3	WI-002	frontend	low	DEV-02	Backend	2026-01-31	2026-01-31	
4	WI-003	backend	high	DEV-02	Backend	2026-02-08	2026-02-11	
5	WI-004	integration	medium	DEV-04	Frontend	2026-02-16	2026-02-18	
6	WI-005	bugfix	low	DEV-02	Backend	2026-01-19	2026-01-19	
7	WI-006	analytics	low	DEV-02	Backend	2026-03-03	2026-03-05	
8	WI-007	ui	low	DEV-01	Backend	2026-02-24	2026-02-25	
9	WI-008	bugfix	low	DEV-04	Frontend	2026-02-08	2026-02-09	
10	WI-009	bugfix	low	FULL-01	Fullstack	2026-01-28	2026-01-28	
11	WI-010	bugfix	low	OPS-01	DevOps	2026-02-05	2026-02-05	
12	WI-011	ui	low	FULL-01	Fullstack	2026-02-16	2026-02-16	
13	WI-012	frontend	low	DEV-03	Frontend	2026-02-06	2026-02-09	
14	WI-013	infra	medium	OPS-01	DevOps	2026-01-08	2026-01-10	
15	WI-014	infra	low	DEV-03	Frontend	2026-02-05	2026-02-06	

Ссылка на файл

Вопрос в чат: есть что-то подозрительное в этих данных? Как бы человек/аналитик в них разбирался?

С помощью ИИ можно сравнительно быстро восстановить реальный поток работы

Попробуем на практике (ChatGPT или Perplexity (Claude))

```
## Запрос 1. Полный поток
Перед тобой CSV с историей задач. Сначала проверь качество
данных. Затем восстанови фактический поток от запроса до
проверки результата. Покажи, где работа выполняется, где
ждет, какие этапы обходятся. Не предлагай решений.
```

Пока ИИ думает: подумайте и о своих данных

В демо

DEV-01, QA-01, TL-01 — роли и идентификаторы

У себя

замените имена на роли или номера

Не загружайте реальные имена, зарплаты, коммерческие условия и персональные данные.

Пока ИИ думает: три времени потока

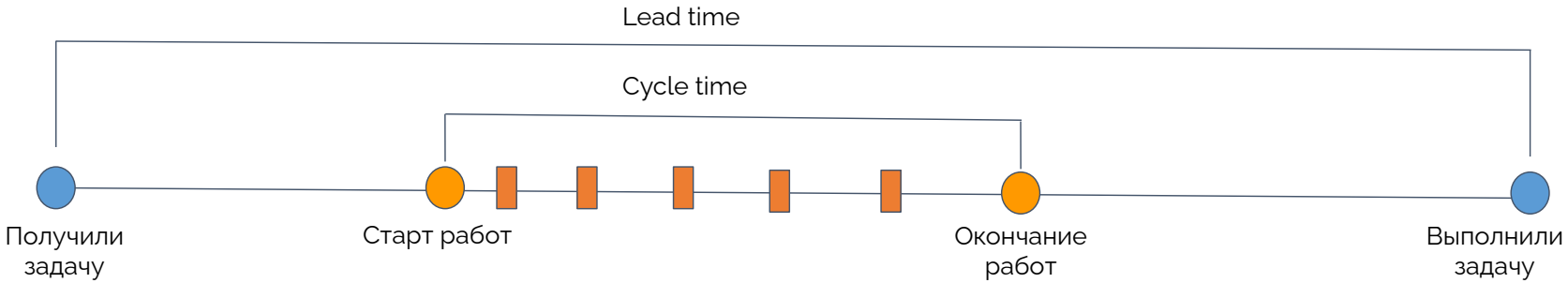
Delivery time = Build time + Wait time + Decision time

10-20% 60-70% 20-30%

При росте масштаба, в хрупких/неустойчивых структурах в первую очередь растет wait + decision time

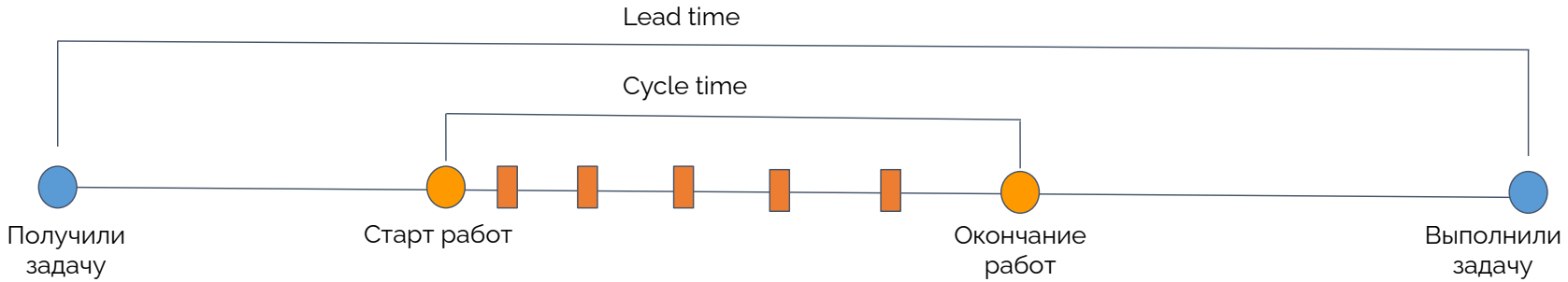
Wait time: представим его визуально

- Время выполнения задачи - lead time (от получили до выполнили)
- Фактические работы - оранжевые
- Все остальное - ожидания
  - Либо внутри команды (чаще)
  - Либо внешние (реже)



ИИ чаще всего ускоряет Build Time.
Организация часто тормозит в Wait Time.

Если ускорить build time (оранжевое), то wait time (промежутки) никуда не денется
Следовательно и ускорение в итоге получится на единицы процентов



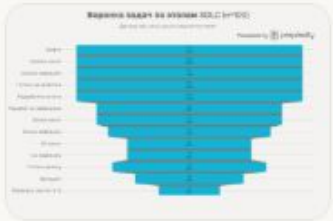
Первые результаты: воронка сделанного

Пропуск этапов

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Обходы этапов

13 задач (13%) явно обходят QA через флаг `bypassed_testing=yes` и выходят напрямую в релиз. Из них: `sdlc_trace_raw.csv`



- Все 12 задач, у которых есть `released_at` при пустом `qa_started_at`, помечены `bypassed_testing=yes` — обход полностью задокументирован флагом `sdlc_trace_raw.csv`
- Среди них: 4 `backend`, 3 `bugfix`, 2 `integration`, 2 `frontend`, 1 `infra` — обходят тестирование в том числе задачи с `risk_class=medium` (WI-064, WI-086) `sdlc_trace_raw.csv`
- **5 задач** возвращались на доработку (`returned_to_dev_count=1`) — все с `low` и `medium` риском, причём WI-013 при этом ещё и `exception_flag=yes` `sdlc_trace_raw.csv`

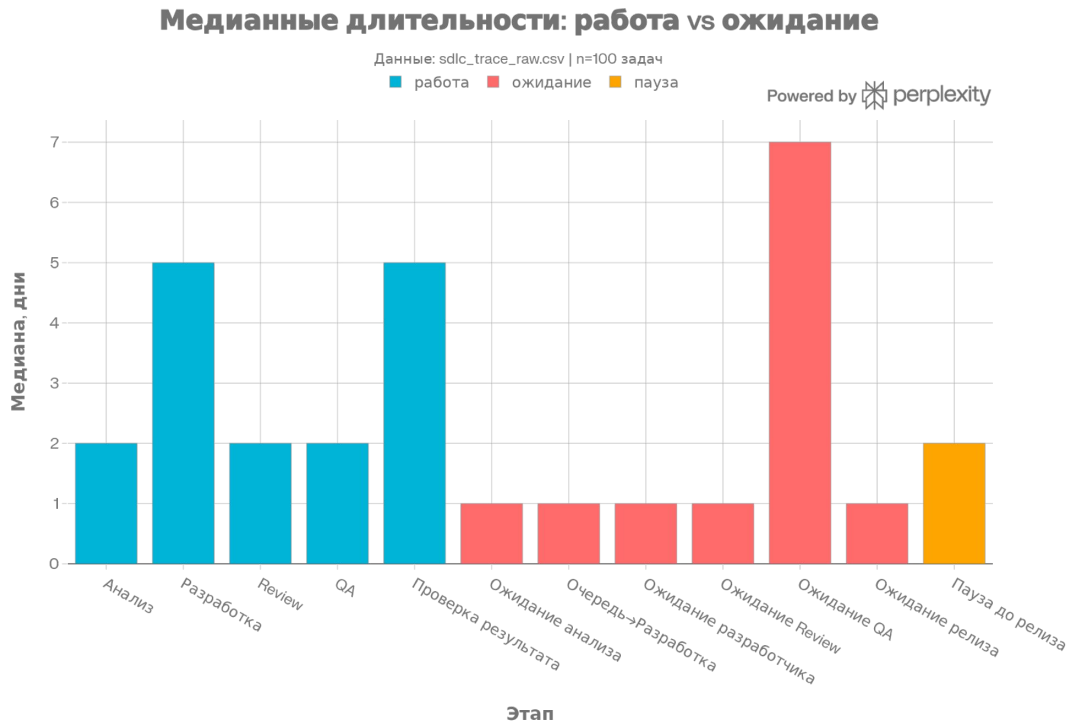
Незавершённые задачи не имеют признака отмены — поле просто пустое. Нельзя отличить задачу в работе от брошенной: 52 задачи без `released_at` и 73 без `outcome_checked_at` могут быть как незаконченными, так и незафиксированными. `sdlc_trace_raw.csv` `d2z0o16i8xm8`

Build time vs Wait time (64%) : гипотеза подтвердилась

Для 27 полностью проверенных задач:

Метрика	Значение
Средний lead time от запроса до проверки результата	27.4 дн
Медианный lead time	27 дн
Средняя активная работа	9.8 дн
Среднее ожидание	17.6 дн
Доля ожидания в полном lead time	~64%

Где работа ждёт дольше всего



Ожидание QA: один симптом, но четыре версии

1. Мало мощности проверки

QA физически не успевает

2. Высокий WIP

много задач одновременно ждёт решения

3. Плохой вход

задачи возвращаются или требуют уточнений

4. Права решений

типовые и сложные решения идут одним маршрутом

Теория ограничений

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

1. Определи ограничение системы
2. Максимизируй его отдачу
3. Подчини все остальное этому решению
4. Расширь ограничение
5. Повторить

Журнал гипотез v1 : после анализа сырых данных из трекера задач

Гипотеза	Уверенность	Что проверить
Недостаточно мощности QA/review	medium	нужно получить данные о доступных компетенциях и фактической нагрузке
Плохое качество входных данных	medium	нужны причины возвратов и критерии готовности
Неравномерная подача задач (mura)	medium	проверить даты подачи задач, релизные окна, итд
Типовые и сложные решения сходятся на одном эксперте	low	карта прав решений: кто может принять качество, исключение и выпуск

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Проверим наши гипотезы

Проверим наши гипотезы

```
## Запрос 2. Версии ограничения
На основе найденных ожиданий сформулируй 4-6 конкурирующие
гипотезы об ограничении. Для каждой: что поддерживает,
альтернативное объяснение, какие данные помогут проверить.
```

Какие данные вы добавили бы следующими?

A

компетенции

B

анализ списка решения и ЛПР

C

причины возвратов

D

календарь встреч

E

KPI функций

Напишите в чат свой выбор (A-E)

Добавим матрицу компетенций

A	B	C	D	E	F	G
employee_id	role	Backend	Frontend	DevOps	QA	Analysis
TL-01	Tech Lead	2	1	1	0	1
DEV-01	Backend Dev	2	0	0	0	0
DEV-02	Backend Dev	2	0	0	0	0
DEV-03	Frontend Dev	0	2	0	0	1
DEV-04	Frontend Dev	0	2	0	0	0
OPS-01	DevOps	1	0	2	0	0
QA-01	QA Lead	0	0	0	2	1
QA-02	QA Engineer	0	0	0	1	0
BA-01	Analyst	0	0	0	0	2
FULL-01	Fullstack	1	1	0	0	0

Ссылка на файл

Чат: где single point of failure?

Добавим список решений и кто их принимает

	A	B	C	D	E	F
1	decision	formal_owner	actual_path	decision_mode	typical_wait_days	risk
2	Признать задачу ни	QA-01	QA-01 confirms man	manual gate	2	bottleneck
3	Разрешить обход ру	QA-01 + TL-01	ad hoc message, the	exception	1.5	quality debt
4	Архитектурный рис	TL-01	review comments + v	expert decision	3	TL bottleneck
5	Изменить приорите	BA-01 + TL-01	weekly backlog meet	batch decision	4	batching
6	Готовность к выпус	QA-01	QA queue, exceptior	approval	5.5	queue
7	Проверка результат	BA-01	manual follow-up, nc	manual follow-up	7	invisible work

Ссылка на файл

Теперь мы видим не только кто умеет, но и кто имеет право остановить или пропустить работу.

Добавим контекста: матрицу компетенций и карту решений

```
## Запрос 3. Компетенции и права решений
Обнови журнал гипотез с учетом матрицы компетенций и карты
решений. Раздели симптом, ограничение, механизм и
структурную причину.
```

Пока ИИ думает: как получить карту компетенций?

- Привязана к человеку (Знает/Умеет/Устойчивый навык)
- Показывает уровни компетенций
- Помогает вычислить bus-factor

Сотрудник/ Компетенция	РКО	Инвести-ционные продукты	Фронтэнд	Десктоп	Сглажива-ние конфлик-тов	Навыки продажи	Жизненный цикл операции	Распреде-ление ответствен-ности в компании
Иванов А.В.	–	–	☆	✓	✓	–	✓	–
Ясина М.Н.	–	☆	–	–	–	☆	☆	✓
Смирнов В.О.	✓	–	☆	–	☆	✓	✓	✓
Петров А.Г.	☆	✓	–	–	–	✓	–	✓
Самойлова А.А.	✓	–	–	☆	–	✓	✓	–
Белкина Е.В.	–	–	☆	✓	✓	–	☆	✓
Кошкин И.П.	☆	✓	✓	✓	–	–	–	–

Пока ИИ думает: как получить список решений и ЛПР

- RACI / DACI матрицы
- Стандарты и процедуры
- Переписка и чаты

Пока ИИ думает: decision time точно так же съедает мощность

Delivery time = Build time + Wait time + Decision time

10% 70% 20%

Даже если нарастить компетенцию, но не дать право решения, итоговая мощность системы не вырастет:

- Формально у нас 2 QA, но решения может принимать только QA-01

Пока ИИ думает: Decision time обычно не видно не трекере

Видно в трекере

- кто вел карточку
- когда менялся статус

Часто скрыто

- кто принимал решение
- согласовывал риск
- держал очередь

Поэтому одного process mining недостаточно для оргдизайна.

Журнал гипотез: симптом → ограничение → механизм → причина

Источники: sdlc_trace_raw.csv + competency_matrix.csv + decision_rights_map.csv

Powered by perplexity

Слой	H1: QA-01 singleton	H2: Пакетная приоритизация	H3: Bypass без критерия	H4: Outcome невидим
Симптом	Ожидание перед QA 7 дн.; approval wait = 5.5 дн. (карта решений)	exception_flag задачи ждут QA дольше: 8 дн. vs 7 дн.	13 bypass; 20% low-risk обходят QA; все через ad hoc	Outcome зафиксирован у 27/100 задач; BA-01 вне dev-борды
Ограничение	Только QA-01 имеет can_decide_low_risk_qa=yes; QA-02 — уровень 1, без права	Смена приоритета — batch decision BA-01+TL-01, только на weekly встрече	Обход требует QA-01+TL-01 вместе; no формального критерия нет	BA-01 единственный ответственный; нет трекинга в dev-системе
Механизм	manual gate: QA-01 подтверждает вручную; QA-02 не может разблокировать	Задачи накапливаются без смены приоритета; exception-first в очереди не помогает	Bypass идёт через QA-01 'exceptions first': очередь создаёт стимул обходить её	После released_at задача уходит из борды; follow-up ручной, нерегулярный
Структурная причина	Право на решение не делегировано QA-02 (RiskReview=1, но can_decide=no)	Нет on-demand-канала приоритизации: вся власть у BA-01+TL-01 на батчевой встрече	Нет формализованного criteria low-risk: каждый bypass — уникальное согласование	Outcome — не часть SDLC-контракта: не трекируется, не назначается
Что проверить	Загрузка QA-01 по времени; параллельность задач в QA; мог ли QA-02 закрыть хоть одну задачу самостоятельно	Дата weekly-встречи vs delta(ready_for_dev → dev_started); кластеризация старта задач по дням недели	Есть ли шаблон/чеклист low-risk в регламенте; распределение bypass по типу задачи и owner	Есть ли outcome у задач где BA-01 = owner; корреляция outcome_lag с work_type

Журнал гипотез v2 : с учетом матрицы компетенций и карты решений

Сводная версия журнала

ID	Симптом	Возможное ограничение	Механизм	Структурная причина
H1	Длинная очередь `review_done → qa_started`	`QA-01` как QA/risk/release gate	Ручное подтверждение risk/QA/release readiness	Право решения сконцентрировано у одного QA Lead
H2	Задачи разных типов одинаково ждут перед QA	Готовность задачи к проверке	Перед QA всплывают риск, данные, стенды, архитектура, критерии	Ключевые компетенции разнесены по BA/TL/QA/OPS
H3	Released ≠ outcome checked	Post-release validation	Проверка результата идет manual follow-up вне dev board	Outcome ownership отделен от delivery flow
H4	Bypass QA + ad hoc exceptions + зависания после review	Decision batching / exception path	Weekly/ad hoc decisions управляют движением задач	Поток зависит от shared decision roles и неявных gates

Проверка выдачи ИИ: факт, интерпретация или гипотеза?

Что тут реально факт, а что гипотеза и интерпретация?

- ожидание перед проверкой — 7,6 дня
- QA-01 является узким местом
- если дать QA-02 право решения, Lead Time упадет

ИИ часто пишет гипотезы тоном фактов. Здесь нельзя расслабляться.

Что модель увидела, а что додумала

Тип	Пример
Факт	даты статусов, обходы проверки, карта решений
Интерпретация	типовые и сложные решения идут одним маршрутом
Гипотеза	разделение маршрутов сократит Lead Time
Выводы, которые нельзя принимать на веру	кто плохой или кого нужно заменить

Галлюцинации ИИ - часто это лишь недостаток входных данных в промпте и недостаточно ограниченный формат выводов

Промежуточный вывод: Узким местом часто является не человек, а маршрут через человека

Мнение ChatGPT

Обновленный вывод

После добавления матрицы компетенций и карты решений я бы переформулировал главный вывод так:

Ограничение, скорее всего, находится не в "ручном тестировании" как операции, а в контуре принятия решений вокруг качества, риска и готовности к выпуску.

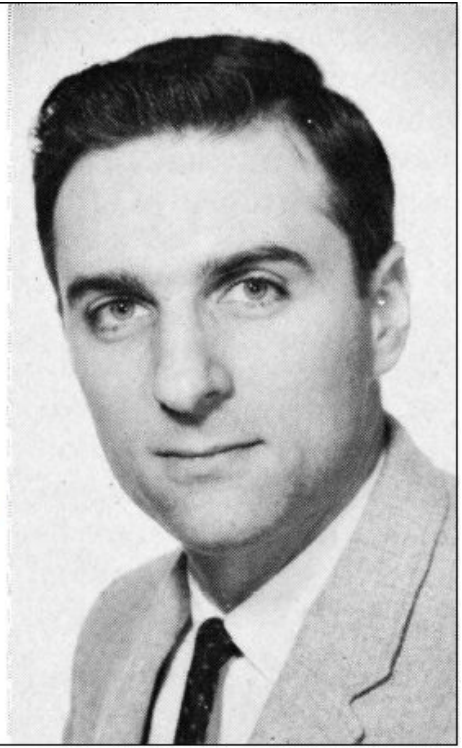
Самый сильный кандидат — **QA-01 как централизованный gate.**
Но конкурирующая гипотеза остается важной: возможно, QA-01 не причина, а место, где проявляется более ранняя проблема — **задачи приходят к QA не полностью готовыми к проверке.**

Отдельно ниже по потоку есть еще один разрыв: **release не равен проверенному результату**, потому что outcome validation живет вне основного dev board.

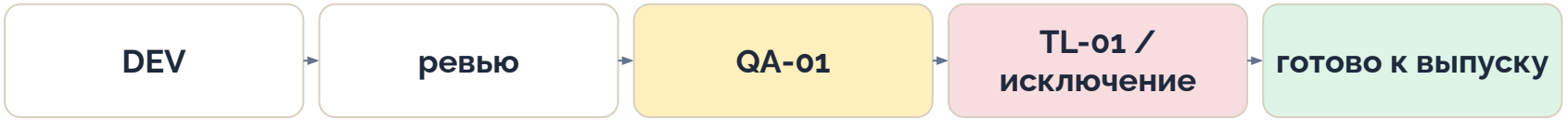
Закон Конвея

«Организации, проектирующие системы, ограничены дизайном, который копирует структуру коммуникации в этой организации»

Мелвин Конвей



Закон Конвея в действии



Каждый нестандартный случай повторяет один и тот же коммуникационный маршрут

То есть нестандартный случай стал стандартным поведением системы

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Проверим гипотезу об узком месте в QA

Что будет если ускорить какой-то этап процесса?

Например ИИ разработка стала быстрее на 40%

Посчитаем сначала мощность (throughput) наших команд Dev и QA

Попробуем на практике (ChatGPT или Perplexity (Claude))

```
посчитай мощность (throughput) команды разработки и
команды QA
```

Сводка мощности: DEV vs QA (янв–март 2026)

n=100 задач | период 74 дня

Метрика	DEV	QA	Разрыв
Завершено задач за период	82	55	−27 задач
Throughput (задач/неделю, медиана)	8.5	7.0	−1.5
Throughput (задач/неделю, среднее)	8.2	6.1	−2.1
Throughput (задач/день)	1.11	0.74	×1.49
Cycle time (медиана, дни)	5.0	2.0	—

Сценарий: разработчики с помощью AI ускорились на 40%

Попробуем на практике (ChatGPT или Perplexity (Claude))

```
с помощью ИИ разработка ускорилась на 40%, но QA остался
тем же - так как работают те же люди QA-01, QA-02

как будет эффект на общую систему?
```

5. Главный вывод

AI-native ускорение разработки на 40% при неизменном QA дает **локальное улучшение**, но не системное.

Система была примерно такой:

```
Dev: ~9 задач/неделю
QA:  ~7 задач/неделю
```

После AI-native станет:

```
Dev: ~12.6 задач/неделю
QA:  ~7 задач/неделю
```

Итог:

```
Общий throughput ≈ тот же
Очередь перед QA ≈ растет в 2.8 раза быстрее
Lead time ≈ ухудшается
```

То есть AI не ускорит delivery flow. Он **ускорит подачу работы в уже существующее ограничение**.

Закон Литтла никуда не делся, теперь его видно явно

$$WIP = Throughput \times Lead\ Time$$

Закон Литтла никуда не делся, теперь его видно явно

При прежней мощности QA, очередь растет быстрее, следовательно при ускорении разработки на 40% и увеличении количества входящих задач и Lead Time всей системы вырастет!

Журнал гипотез v3 : после стресс теста

Гипотеза	Уверенность	Что изменило оценку
Текущая мощность QA ограничивает мощность всей системы	very high	рост входа на 40% превышает текущую мощность маршрута
Типовые и сложные задачи идут через QA-01	high	QA-02 есть, но без самостоятельных прав не разгружает решения по типовым задачам
Неравномерная подача задач (mura)	medium	weekly decisions влияют на приоритеты и выпуск
Плохой вход усиливает возвраты	medium	дефекты и возвраты могут съесть часть эффекта AI-ускорения

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Попробуем исправить ситуацию?

Журнал гипотез v3 : после стресс теста

Гипотеза	Уверенность	Что изменило оценку
Текущая мощность QA ограничивает мощность всей системы	very high	рост входа на 40% превышает текущую мощность маршрута
Типовые и сложные задачи идут через QA-01	high	QA-02 есть, но без самостоятельных прав не разгружает решения по типовым задачам
Неравномерная подача задач (mura)	medium	weekly decisions влияют на приоритеты и выпуск
Плохой вход усиливает возвраты	medium	дефекты и возвраты могут съесть часть эффекта AI-ускорения

Как думаете, с каких шагов стоит начать?

A

ограничить WIP для QA

B

разделить типы задач QA

C

дать полномочия QA-02

D

автоматизация QA

Напишите в чат свой выбор (A-D)

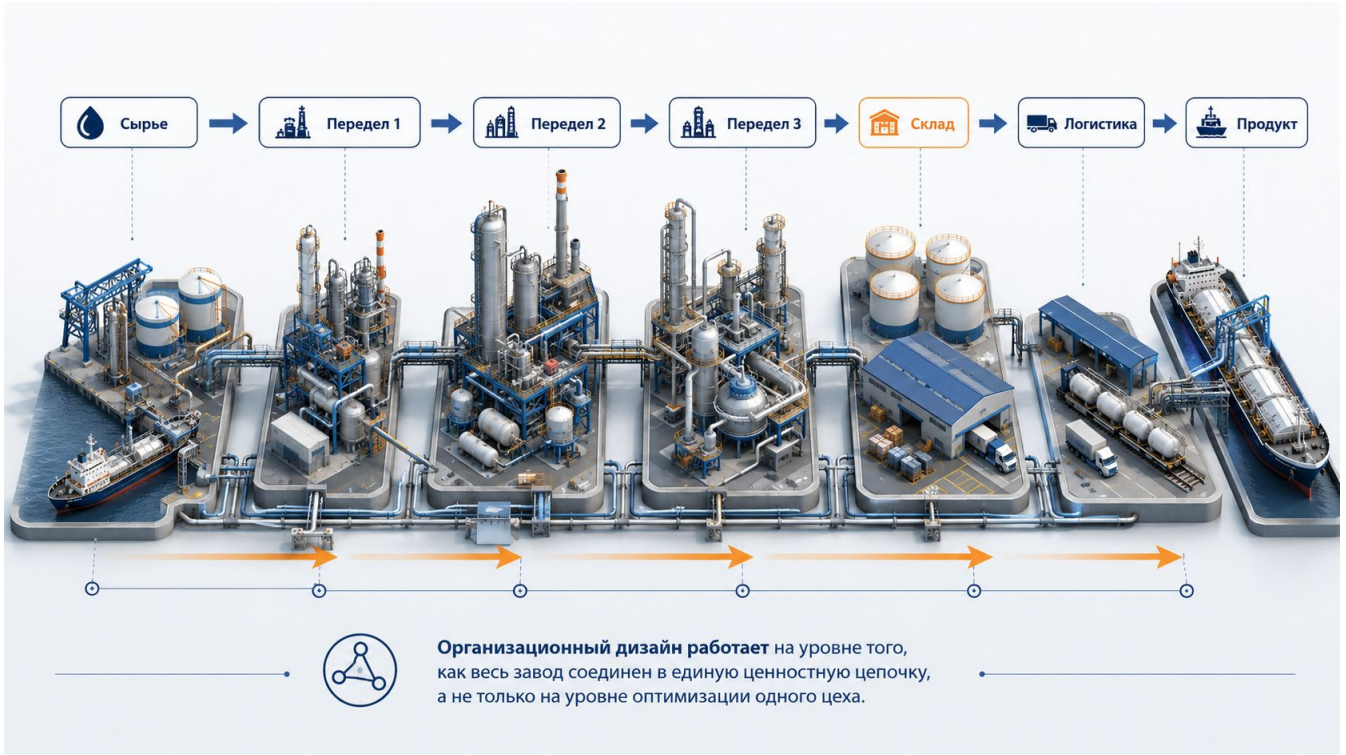
Оргдизайн начинается с цели

Что хотим производить?
Сколько?
Что нельзя ухудшить?

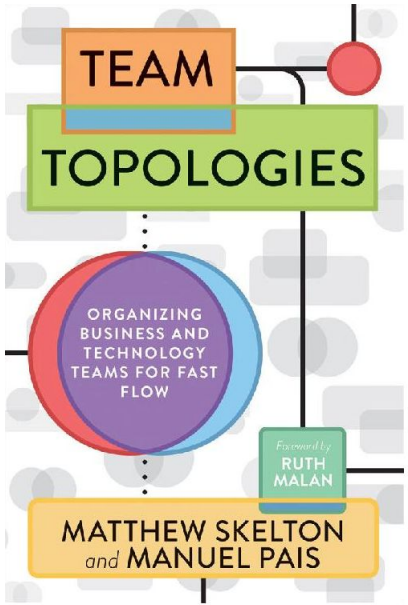
Основные вопросы оргдизайна

1) Кто отвечает за результат всего потока?
2) Где проходят границы команд?
3) Кто и какие решения принимает?
4) Как команды передают работу?
5) Какой набор компетенций находится в командах?

Процесс оптимизирует цех. Оргдизайн проектирует завод.



Team Topologies:
Оргдизайн как инженерное мышление



Идея проста: думать о командах как о частях системы, а не строках в орг чарте

- Команды, как независимые сервисы
- Границы важны: подчеркивают ценность и ответственность
- Измеряем насколько быстро проходит изменение/запрос через команду, а не “сколько ресурсов выделено”
- Формализуем взаимодействие (Team API)

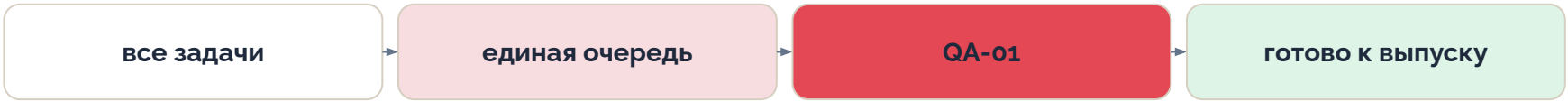
Попробуем сформулировать гипотезы новой организации

Попробуем на практике (ChatGPT или Perplexity (Claude))

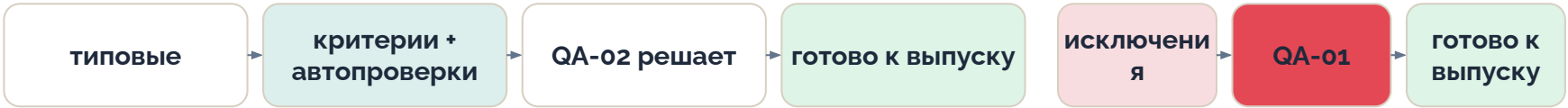
```
Сформулируй три гипотезы, как можно перепроектировать нашу
организацию.

Не принимай решение за руководителя.
Не рисуй “идеальную” оргструктуру.
Учитывай текущих людей и компетенции.

Для каждой гипотезы покажи преимущество, цену, новый риск и
эксперимент.
```

Обе модели рекомендуют разделить задачи на типовые и сложные

Сейчас



Гипотеза



Меняется маршрут работы, а не только “кто кому подчиняется”.

Проведем сценарную симуляцию

Попробуем на практике (ChatGPT или Perplexity (Claude))

```
Переведи выбранную оргдизайн-гипотезу в параметры модели. Проведи осторожный, базовый и оптимистичный сценарий. Покажи диапазон результатов и чувствительность к ключевым допущениям.
```

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Пока ИИ думает: ваш прогноз до расчета?

A

почти не изменится

B

рост x1,3 - x 1,5

C

рост x1,7 - 1,9

D

x2 и выше

Напишите в чат свой выбор (A-D)

5. Диапазон результата

По этой модели диапазон эффекта такой:

Показатель	Сейчас после AI без оргизменения	Диапазон после оргизменения
System throughput	7.3 задач/нед	7.5–11.2 задач/нед
Прирост мощности	—	+2.5% до +52%
Очередь перед QA/gate	+5.3 задач/нед	+1.4 до +5.0 задач/нед
Выпуск за 4 недели	~29 задач	~30–45 задач

Главный вывод:

```
Оргдизайн-гипотеза может дать значимый эффект,
но только если QA-02 реально получает decision rights,
а не просто помогает QA-01 выполнять старый процесс.
```

стратоплан
ШКОЛА МЕНЕДЖМЕНТА

Что попробовать завтра?

Что можно сделать <1 день

- Выгрузить данные из системы (30-50 последних или типовых кейсов, сделок, задач)
- Восстановить фактический поток задач и ожидания
- Собрать несколько версий узкого места
- Добавить другие данные: матрицу компетенций, права решений, информацию о командах, итоги perf review, итд
- Разработать 1-2 сценария улучшения

Какие “следы” подойдут

Трекер задач (Jira, итд)

Маршруты, ожидания, возвраты

Цели компании

- декомпозиция потока ценности по отделам, OKR
- KPI, кто за что будет бороться, локальные оптимизации

RACI, DACI, переписка (Outlook, итд)

карта решений, кто где в процессе

Календарь встреч

ритм работы

Начните не с новой оргсхемы.
Возьмите один поток ценности (value stream)

И посмотрите, что он реально производит,
где ждёт и каким маршрутом проходит через
вашу организацию.

Q&A

Александр Апазиди

CTO//CIO, тренер, ментор

- Более 30 лет в сфере информационных технологий, прошел путь от разработчика и руководителя проектов до ИТ-директора
- Schlumberger, IBM, Nestle, InBev, Volvo, Metro C&C, СИБУР

@apazidi