CUSTIS

Архитектура софта и бизнеса в сложном ИТ-ландшафте

Максим Цепков
Главный архитектор решений CUSTIS
Навигатор по миру Agile, бирюзовых организаций и спиральной динамики

http://mtsepkov.org

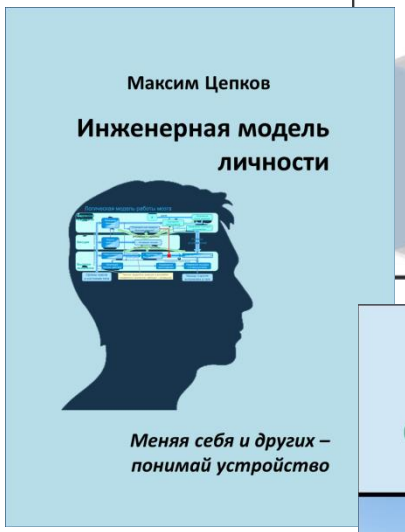
14–15 ноября 2025
Москва

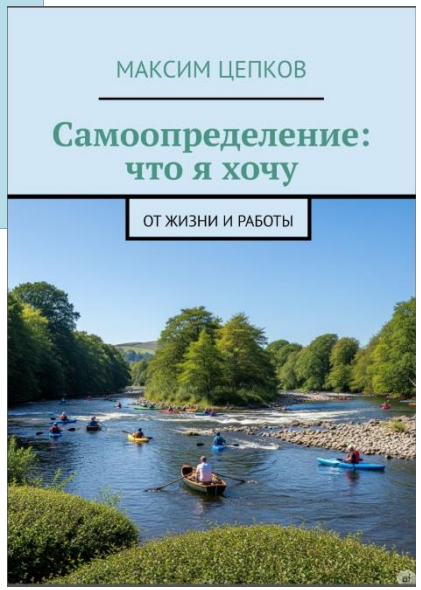
21 analyst days

Немного обо мне

Создание и внедрение больших корпоративных систем (30+ лет)

- Знание практик операционного управления и ведения проектов в коммерческих и государственных организациях и банках.
- Опыт управления проектами в ИТ — от инженерного подхода и PMBOK к современным Agile-методам (с 2007 года).
- Опыт перестройки организаций при внедрении систем.





Навигация в менеджменте цифрового мира

- Agile и самоуправление: бирюзовые организации, холакратия и социократия (**книга**, статьи и выступления).
- Модель спиральной динамики (с 2013 года) и другие модели soft skills, **модели личности** (**книга**) и **самоопределения** (**книга**).
- СМД-методология и развитие СРТ при переходе в цифровой мир.

На сайте mtsepkov.org мои выступления и много других материалов.

О чём этот доклад?

- Задача софта — поддержка бизнеса, при проектировании надо удерживать связь между этими уровнями, а не строить независимые модели.
- Современные архитектуры — гибриды из двух предельных подходов: **монолита и микросервисов,** мы разберем их и коснемся гибридов.
- Транзакции в монолите обеспечивали устойчивость, а масштабирование давало железо, в сервисной архитектуре это часть проектирования.

Мои выступления по смежным темам

- Постановка от модели бизнеса до детального дизайна требований: как делать и кому (13.03.2025)
- Требования или модели — как писать постановки (AnalystDays-2023)
- Визуальное проектирование масштабируемых приложений (TechLead-2021)
- DDD: модели вместо требований 9 лет спустя (ЛАФ-2023)
- Обеспечиваем устойчивость интеграции (SQAdays-2025)

От бизнеса к архитектуре софта

Проект чёрного ящика: user story + экраны

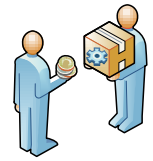

Сделать заказ


Обработать


Собрать


Отгрузить

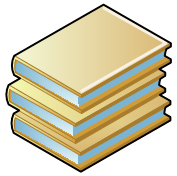

Отвезти


Отдать

Деятельность — обработка заказов интернет-магазина

- Описываем деятельность как user story.
- Рисуем экраны системы.

А всё остальное пусть делают разработчики.



Для покупателя: как создать и оплатить заказ

Для оператора: обработка заказа в офисе

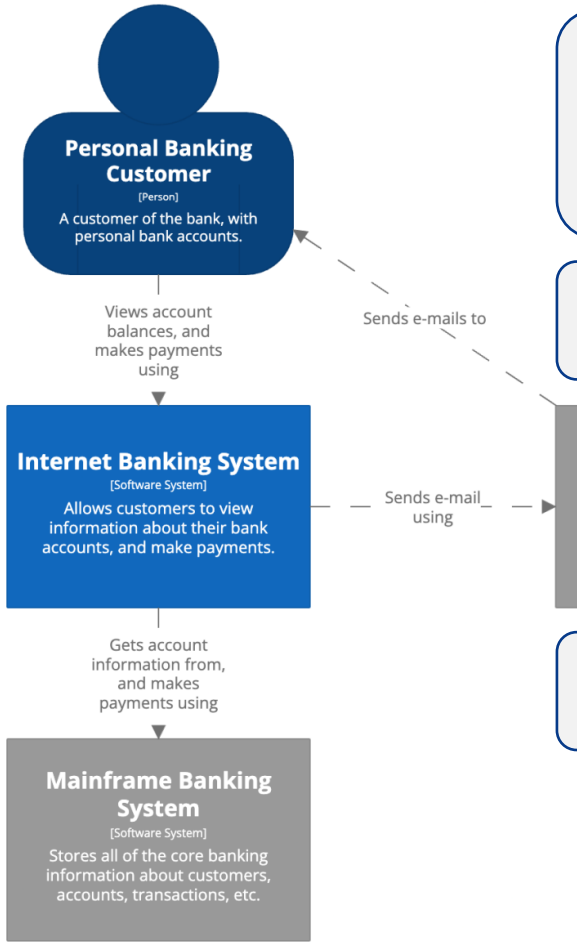
Для кладовщиков: сборка заказов и выдача курьерам

Для курьеров: передача покупателю

Руководства по системе

** Это только обработка заказов, закупку товаров и распределение по складам не рассматриваем*

Контекстная диаграмма C4 Model — система в окружении

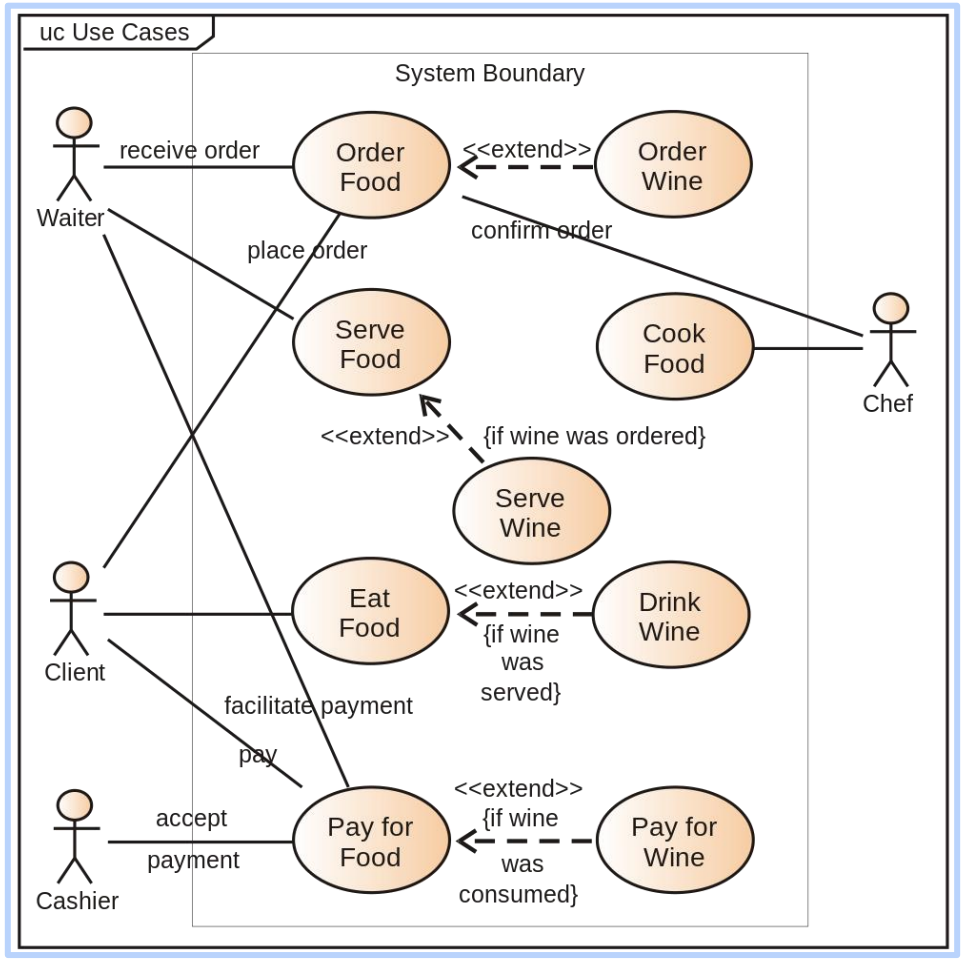


В центре — наша система, вокруг — пользователи и взаимодействующие системы

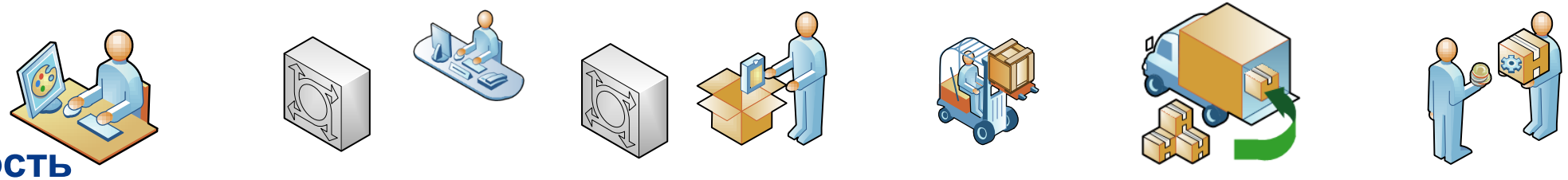
Стрелки — акты взаимодействия

Множество внешних систем — смена парадигмы на сервисы

По сути, это свёрнутая use case диаграмма: вместо овалов — стрелки с подписями



Уровни представления (интернет-заказ)



Деятельность

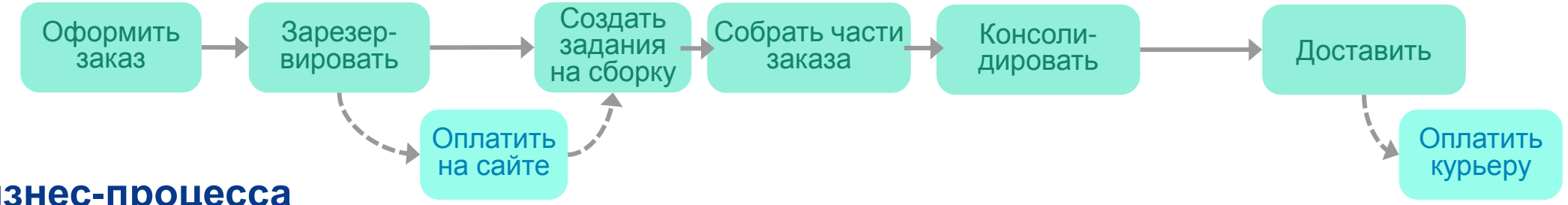
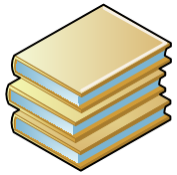


Схема бизнес-процесса

Объекты и их состояния



Для покупателя: как создать и оплатить заказ

Для оператора: обработка заказа в офисе

Для кладовщиков: сборка заказов и выдача курьерам

Для курьеров: передача покупателю

Руководства по системе

Формальная бизнес-модель и реальность

Оптовые продажи магазинам и торговым сетям

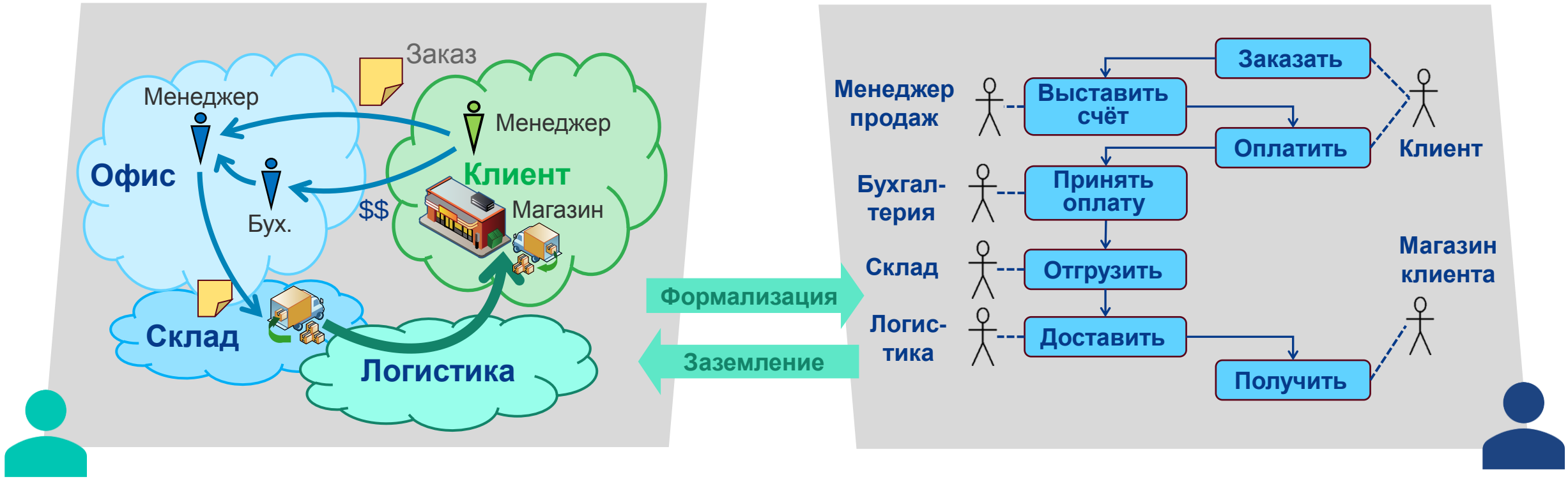
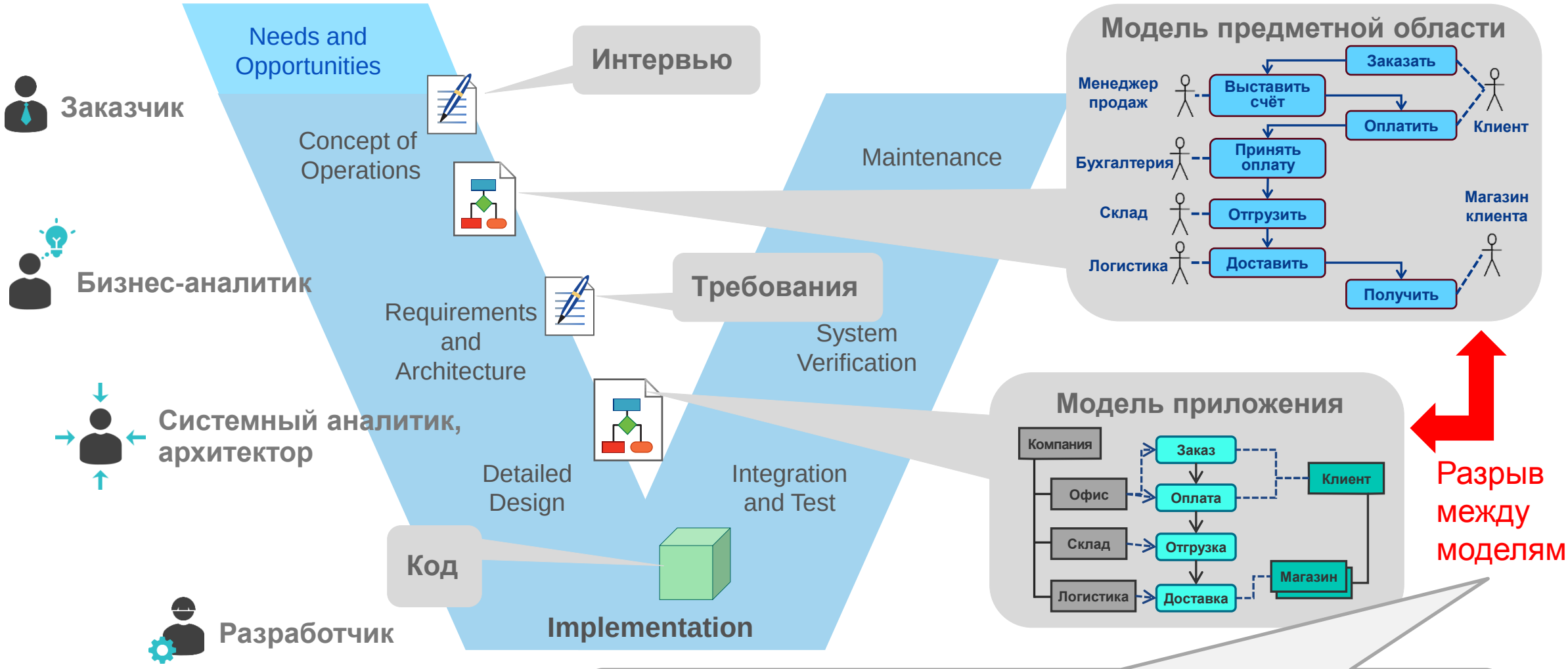


Схема бизнес-процессов — это модель происходящего в реальности.
Видим повседневную деятельность за формальной бизнес-моделью!

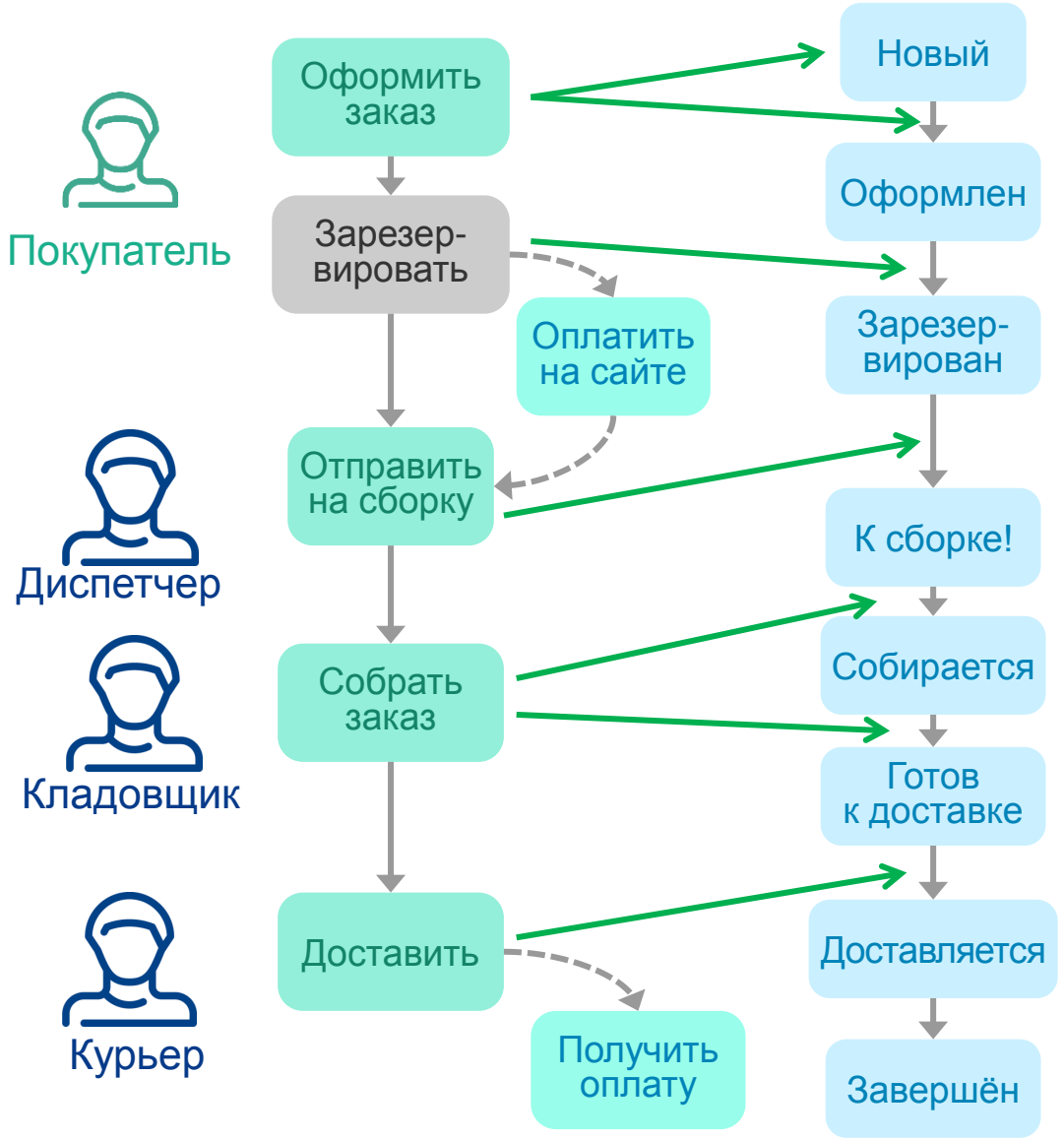
Поддержка артефактами (классика)

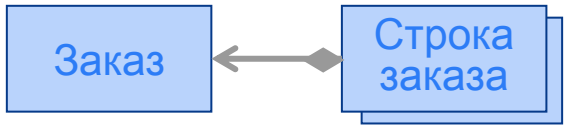


Проблема: каждый уровень отдельно сложно изменять и поддерживать соответствие.

Монолит: бизнес-процесс через workflow документов

Монолит — процесс через workflow

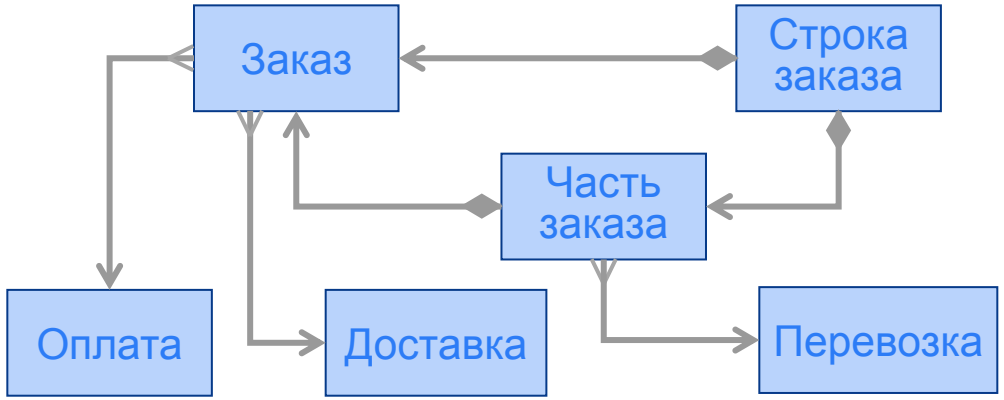




- Состояние заказа — единственный атрибут.
- Отражение оплаты — две суммы в заказе.

Усложнения:

- Собирают на складе несколько кладовщиков.
- Собирают на нескольких складах, объединяют для выдачи покупателю или курьеру.
- Заказ делят — оплата относится к нескольким.



CUSTIS

Формализация снабжения магазинов

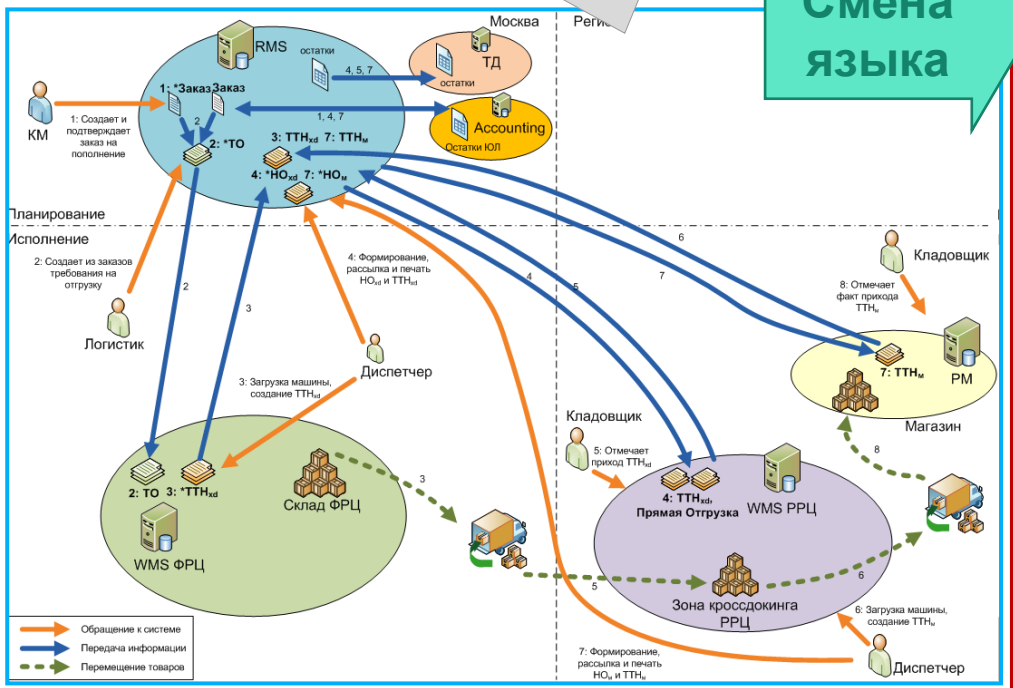
Объекты — Class Diagram

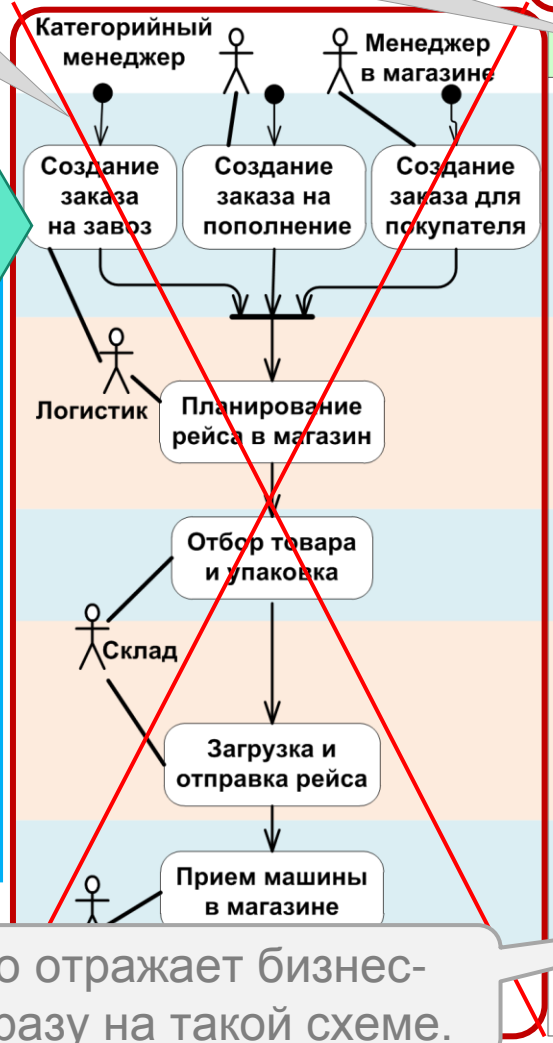
Бизнес-процесс — Activity Diagram

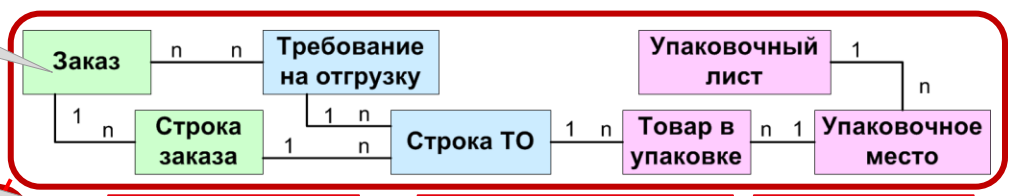
Состояния документов — State Diagram

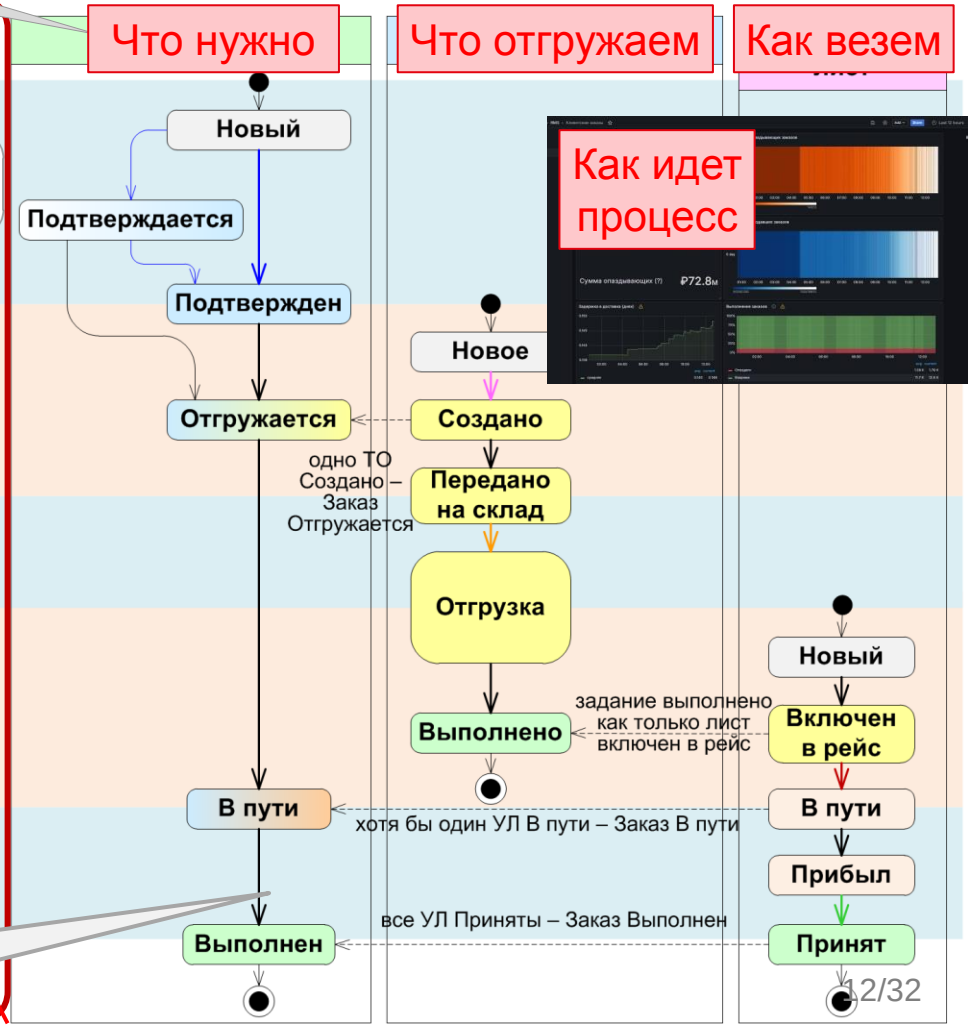
Неформальная схема деятельности и её отражение в существующих системах

Смена языка





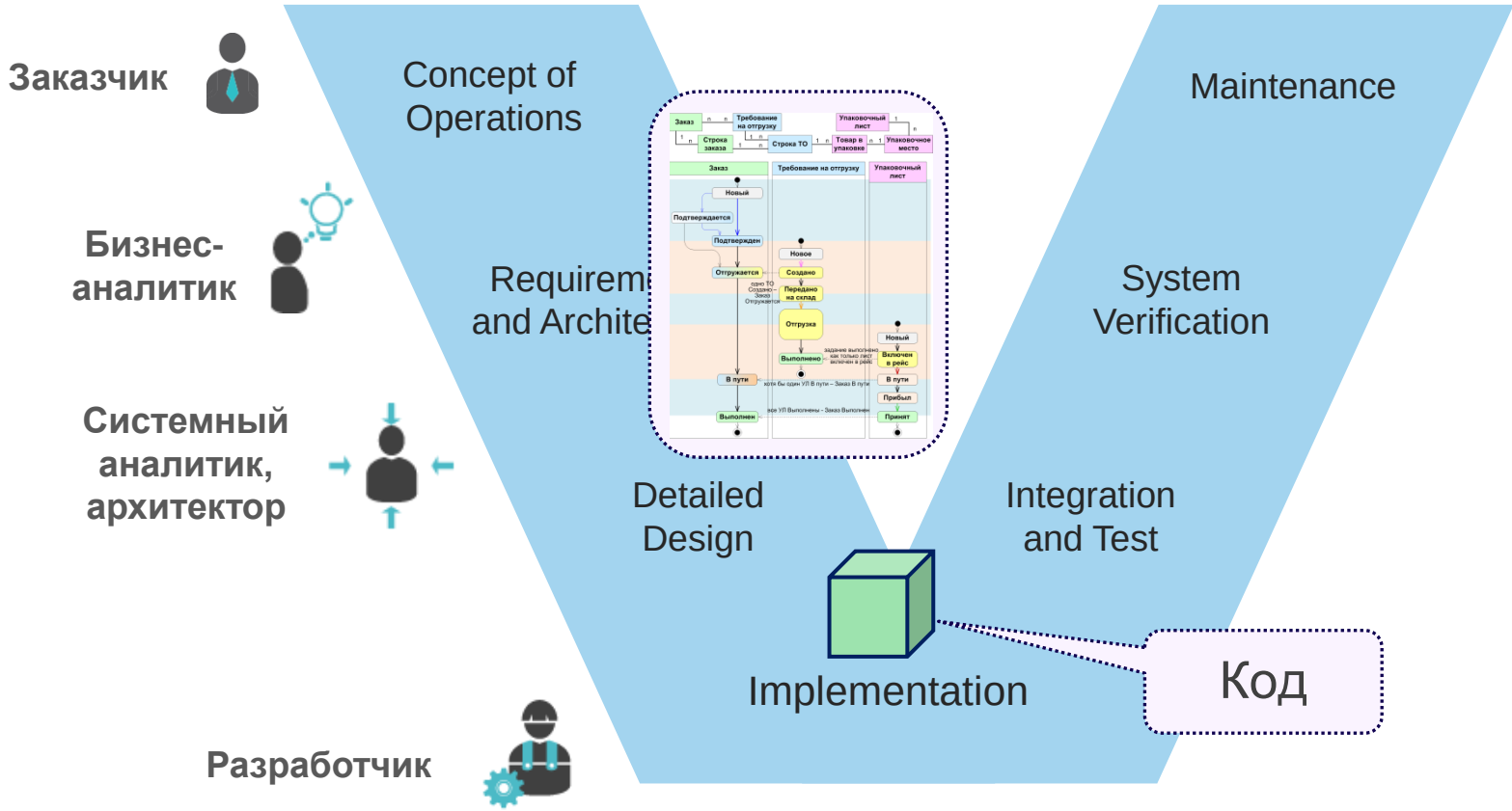




Если workflow документов прозрачно отражает бизнес-процессы, то можно обсуждать их сразу на такой схеме.

Domain Driven Design

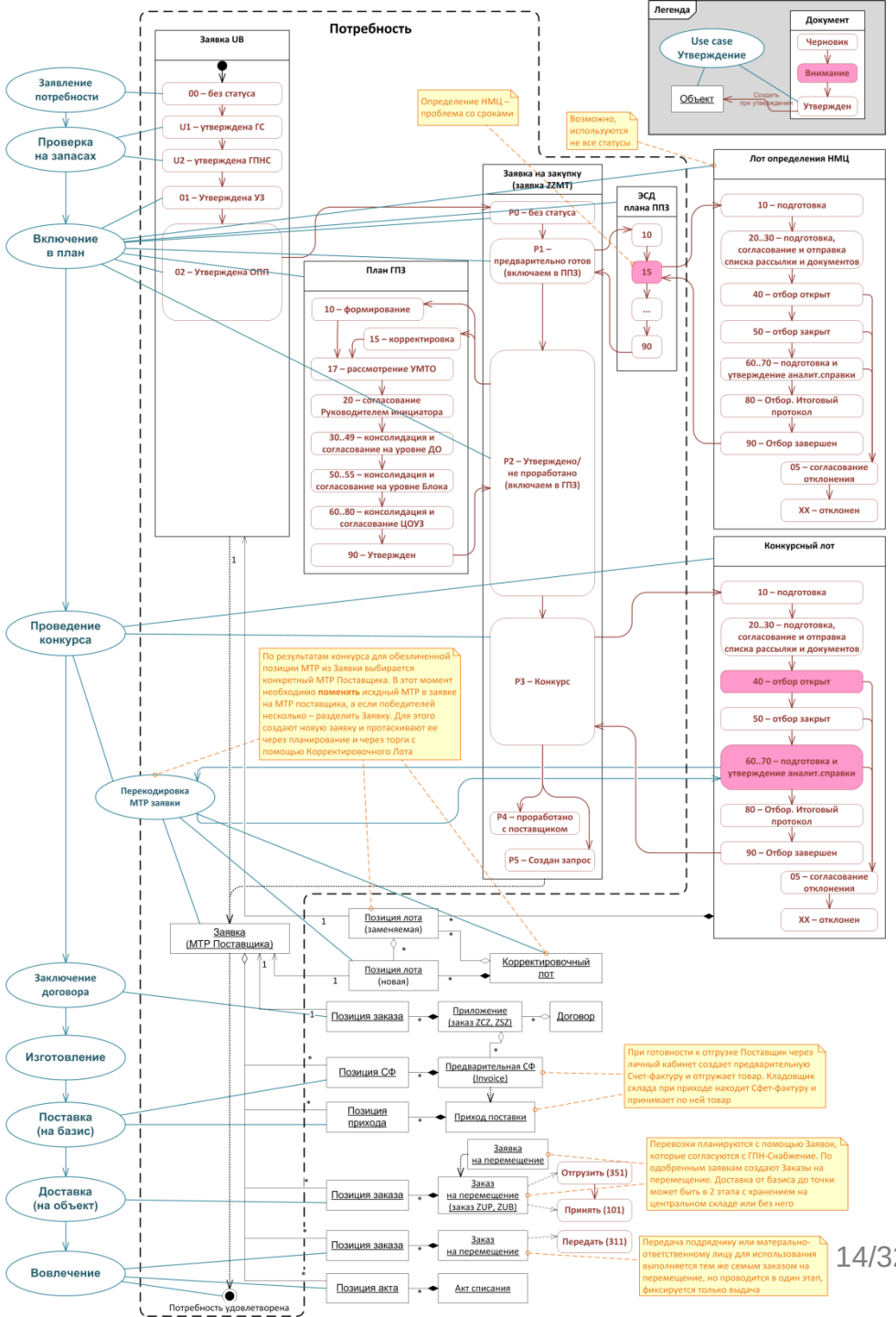
- Единый язык:
  - на основе терминов предметной области;
  - понятен всем участникам проекта.
- Единая модель, приложения и его встройки в бизнес.
- Прозрачное отражение модели в код.



У меня есть много докладов о DDD, последний — «DDD: модели вместо требований 9 лет спустя (ЛАФ-2023)».

Исторические наслоения реального workflow

- Бизнес-процесс усложнялся, для поддержки в системе приспосабливали разные документы.
- Точки внимания бизнеса оказывались глубоко «закопаны» в workflow — их не видно на интерфейсе.
- Вторая часть workflow реализована множеством не связанных документов — это порождает проблему трассировки.



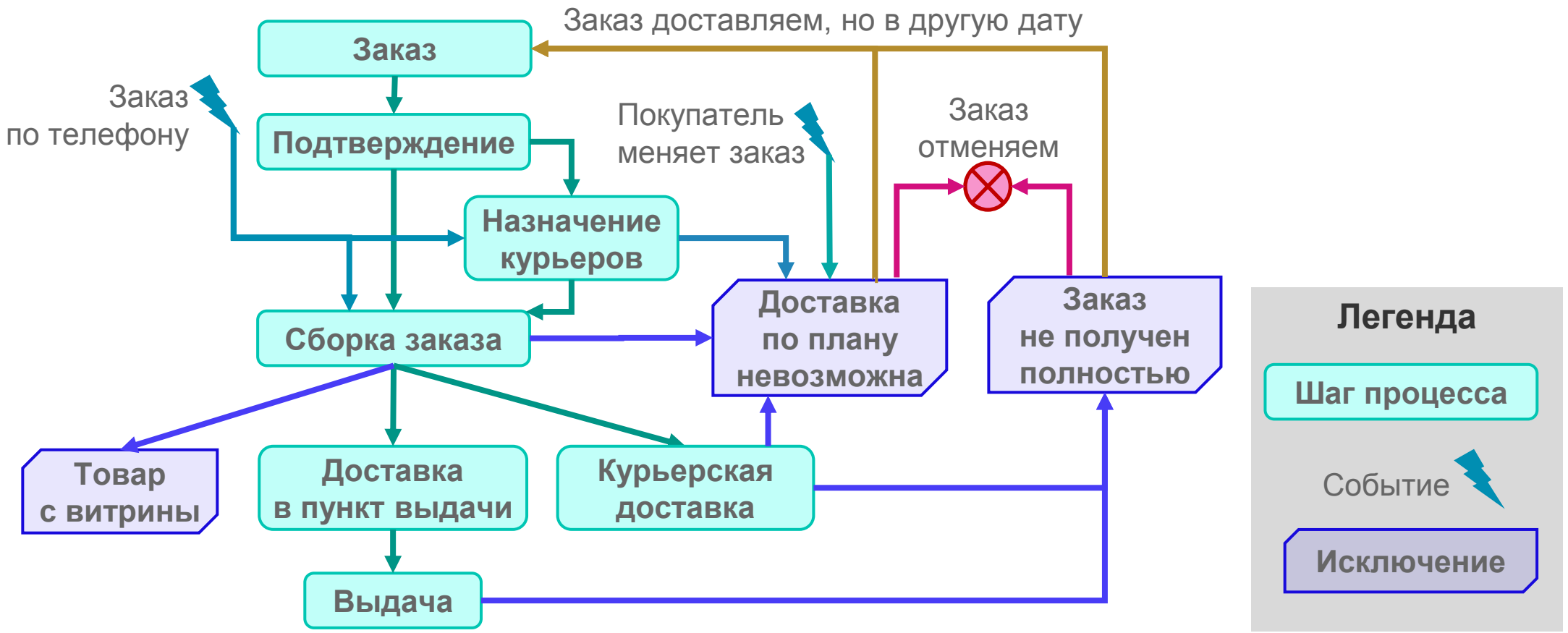
Что не описывается через workflow?

- **Справочники**
  - Структура справочных данных: товары, цены, акции, покупатели, точки выдачи.
  - API и алгоритмы вычисления: цены со скидками, доступные даты доставки.
  - Способы настройки параметров алгоритмов на основе справочных данных.
- **Планирование** деятельности с учётом ограничений требует не только алгоритмов, но и эргономичных форм для выполнения сложных действий
  - Назначение курьеров для развозки заказов.
  - Планирование закупок, пополнения складов и доставки заказов в точки выдачи.
- Как обрабатываем **исключения**: не нашли товар, не можем доставить
- **Dashboard** мониторинга процессов, можно использовать возможности Grafana или аналогов, если сделать журналирование в бизнес-терминах.

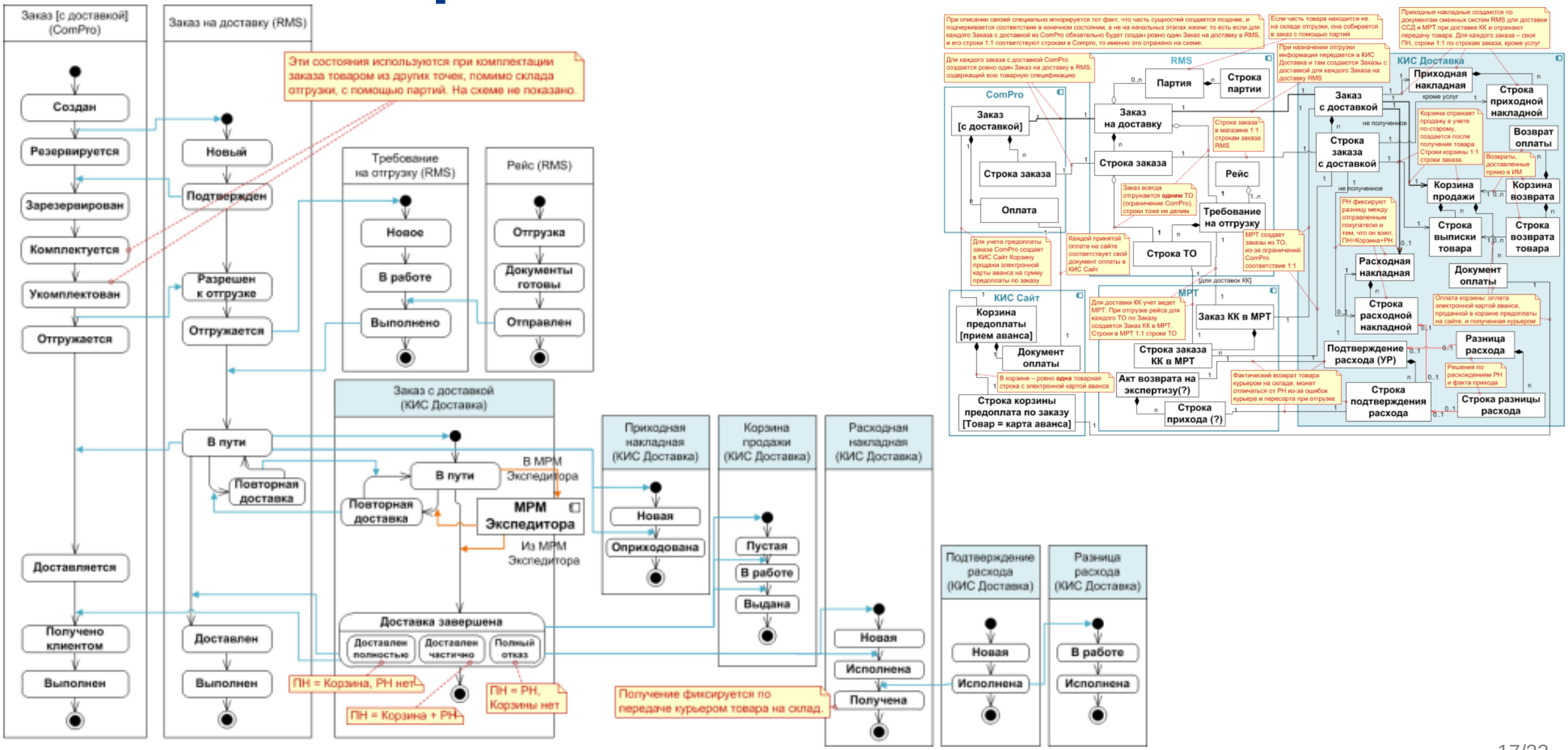
Workflow документов обычно является **основой**, остальное — окружение.

Исключения: Case Management

Процесс обработки заказа интернет-магазина с исключениями из моего выступления «Process и Case Management в информационной системе: от автоматизации As Is к поддержке развития бизнеса», это разобрано от уровня бизнеса до способов реализации.



Workflow через множество систем



(Микро)сервисная архитектура

Зачем появились микросервисы?

- Мечта с помощью ООП получить единую архитектуру предприятия, настраиваемую аналитиком без разработки – провалилась. Реальность – сложный мозаичный ИТ-ландшафт.
- Бизнес требует изменений, а в больших системах приложений это сложно.
- Решение: интеграция через шину и мастер-данные (SOA и аналоги).
- Public web: развитие интернет и смартфонов создало новую реальность.
- Требуется масштабирование – решением стало развертывание в кластере.
- Маленький сервис требует меньше ресурсов – их начали уменьшать
- Маленькие сервисы легче развивать и можно переписать, если необходимо

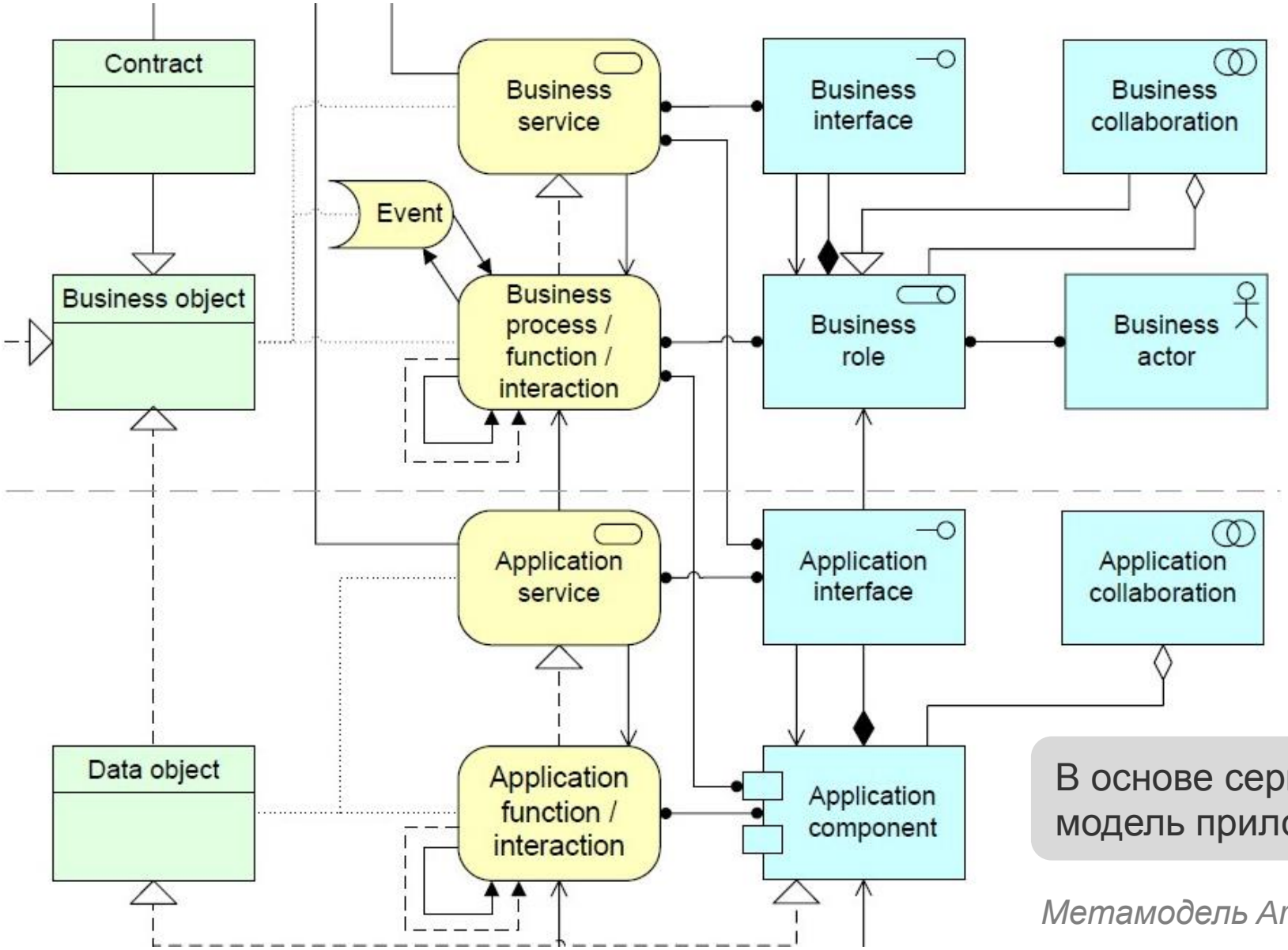
Так появились микросервисы.

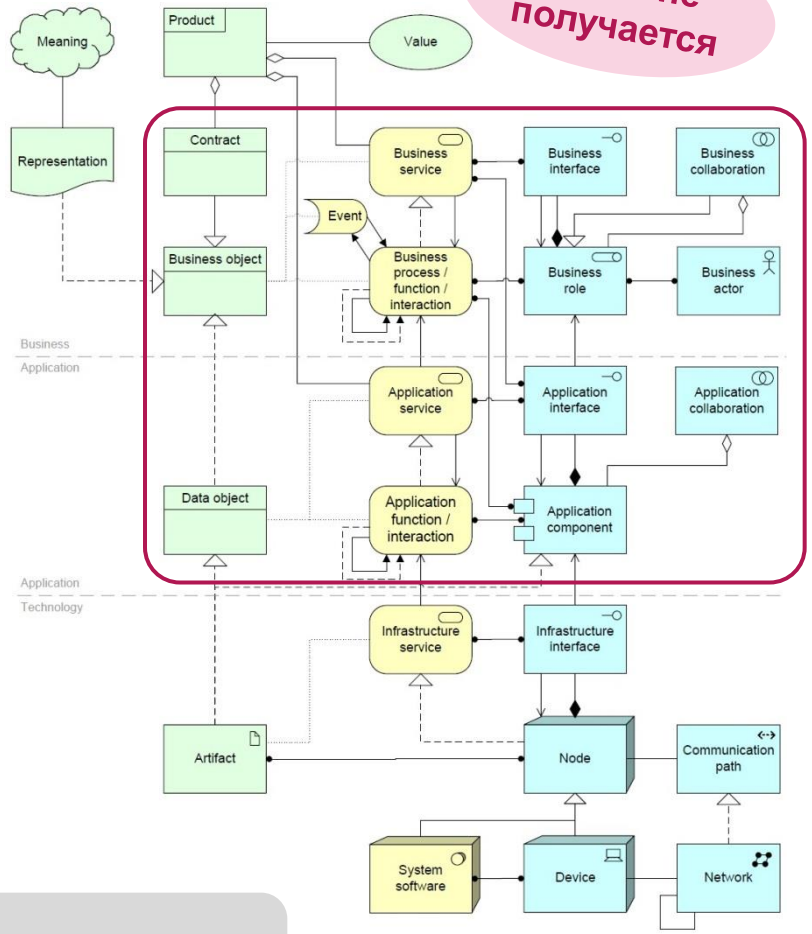
Что изменилось в сервисной архитектуре

- Каждый бизнес-запрос обрабатывают много сервисов.
- Транзакционность и консистентность обеспечивается в приложении.
- Поднимают много экземпляров сервиса, каждый может упасть по ошибкам или блокировкам, а система должна работать устойчиво.
- Асинхронные сообщения и очереди для выравнивания производительности разных сервисов.
- In-memory хранение в базах данных и очередях, сброс в хранилища.
- Восстановление при сбоях узлов кластера и дата-центров — техника и базовый софт не обеспечивают межсистемную консистентность.

Каждый пункт ведёт к своим проблемам работы приложения, и все их надо проверять.

Archimate — гибкая связь бизнеса и софта



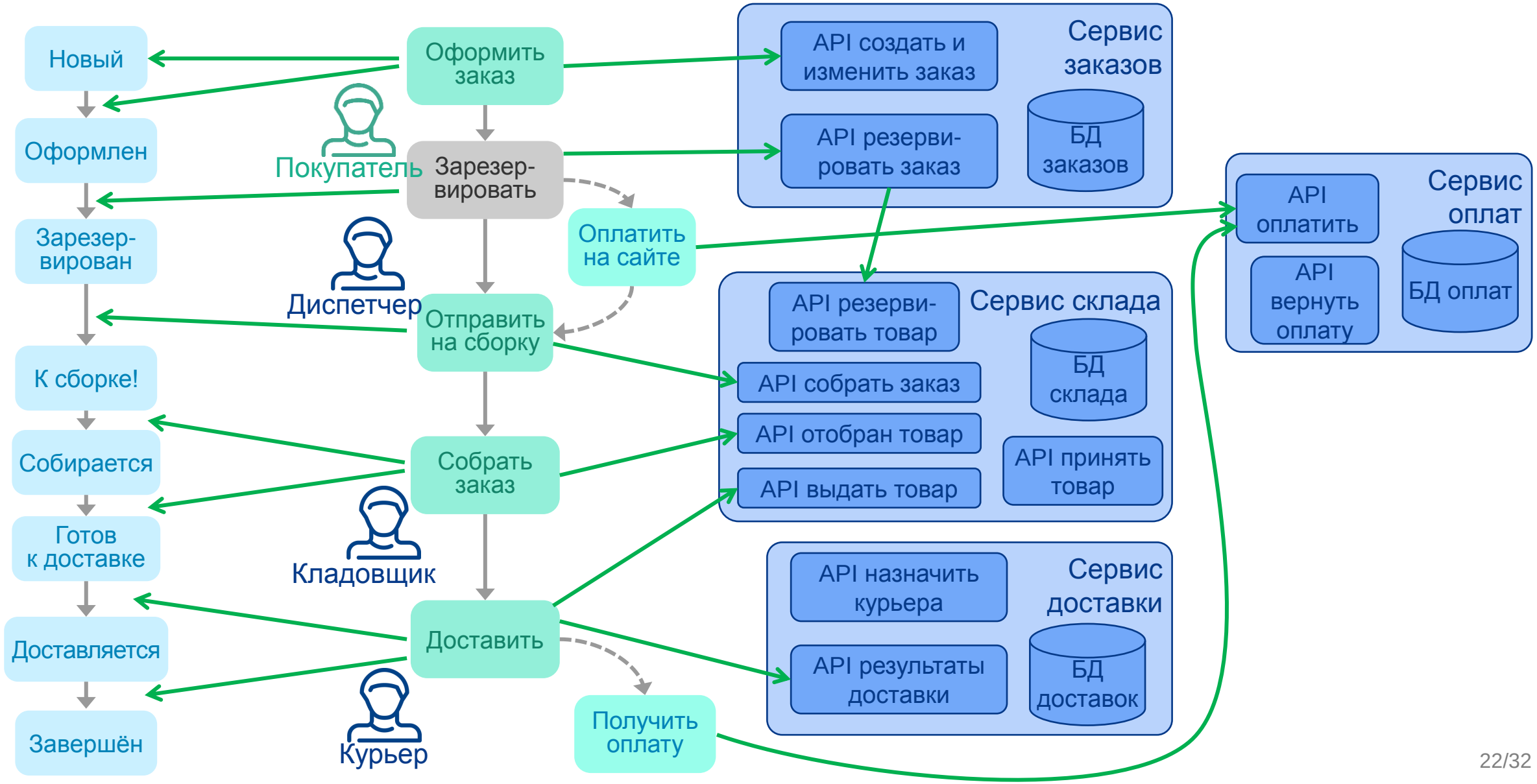


Гибко не получается

В основе сервисная модель приложения

Метамодель ArchiMate 1.0

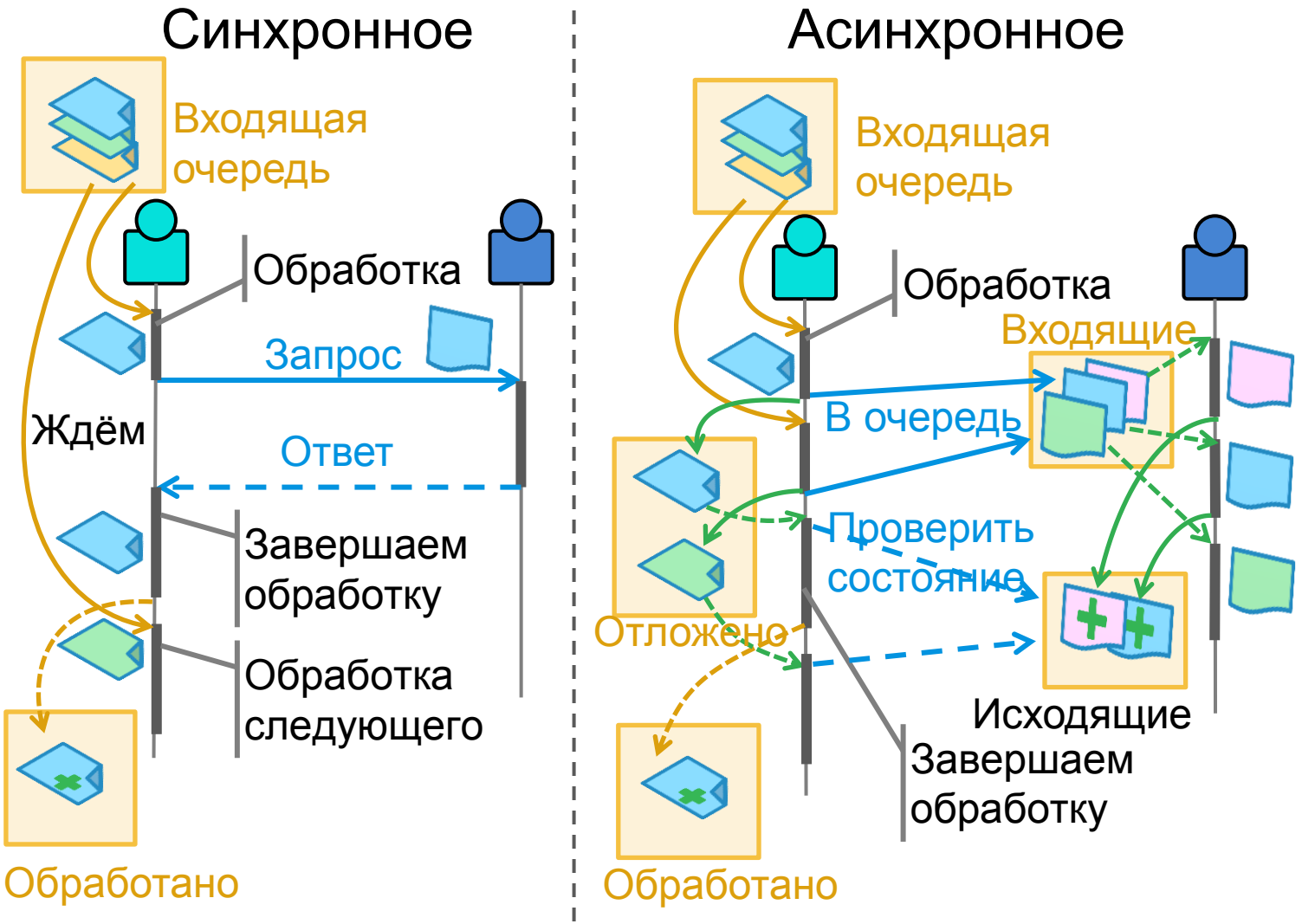
Монолит vs сервисы: workflow vs API



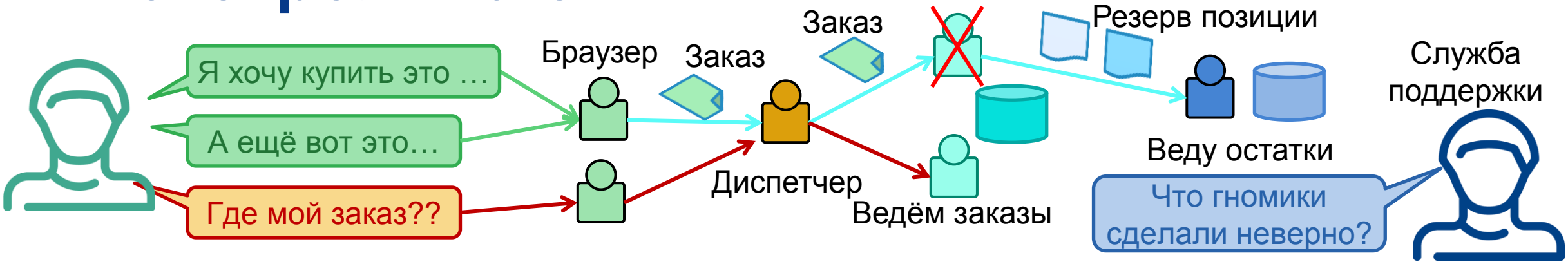
Проектируем API вместо workflow

- Заказ передаётся между сервисами Заказов, Оплат, Склада и Доставки (хореография), или сервис Заказов оркестрирует исполнение, или гибрид: основная цепочка Заказ — Склад— Доставка, а Оплаты — вызываются?
- Транспортный объект: полная структура заказа или то, что нужно сервису? Например, цены и полный адрес со схемой проезда не нужны складу.
- Транспорт должен быть устойчив к расширению возможностей систем!
- Какое API предоставляют справочники?
  - Денормализация: названия товаров хранятся в заказе или каждый раз запрашиваются?
  - Какие у нас сервисы ведения цен, скидок и рекламных акций? Кто вычисляет цену позиции заказа с учётом скидок и акций, учитывая что возможные комплекты и сложные условия (бесплатная доставка, если заказ больше 1000 рублей и для крупных товаров)?
- Планирование доставки и её исполнение — один сервис, или два с общей базой, или два с разными базами, плюс сервис база адресов и геоданных?

Варианты межсервисного взаимодействия



Устойчивость: инстансы умирают, их отстреливают

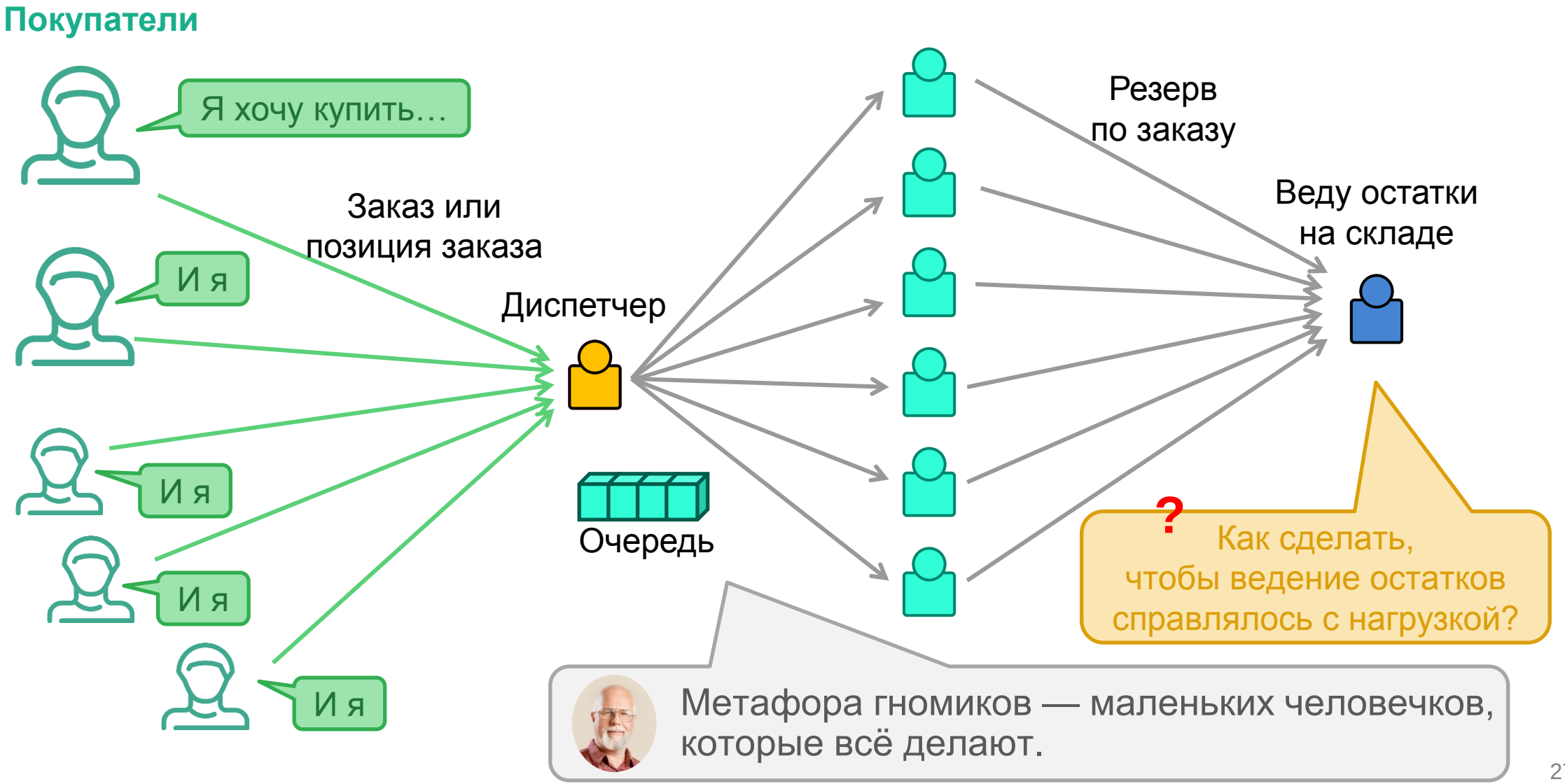


- Ситуация: идёт обработка и резервирование заказа, и в этот момент:
  - Инстанс, ведущий резервирование, падает или его убивают…
  - Покупатель долго не видит ответа в браузере и открывает новый…
- Сценарии для проектирования:
  - Инстанс заказов упал, когда была зарезервирована часть заказа.
  - Инстанс ведения остатков не прислал ответ, но резерв при этом мог быть установлен.
  - Покупатель начал работу из другого браузера, пока запрос первого обрабатывается.

Устойчивость взаимодействия

- Протоколов передачи, обеспечивающих exactly once, не существует!
- Идемпотентные операции обеспечивают устойчивость обработки:
  - Не Insert + Update (CRUD и REST), а **Upsert**, уникальный ключ создаёт клиент.
  - Распределённые транзакции убивают скорость, проектируем без них.
- Учитываем: при падении инстанса портятся связанные с ним очереди.
- Обеспечиваем устойчивость на большом потоке сообщений.
- Инциденты неизбежны — нужна хорошая админка для быстрого разбора.
- Устойчивость приложения обеспечивает проектирование.

Масштабирование сервисов



Варианты ведения остатка на складе

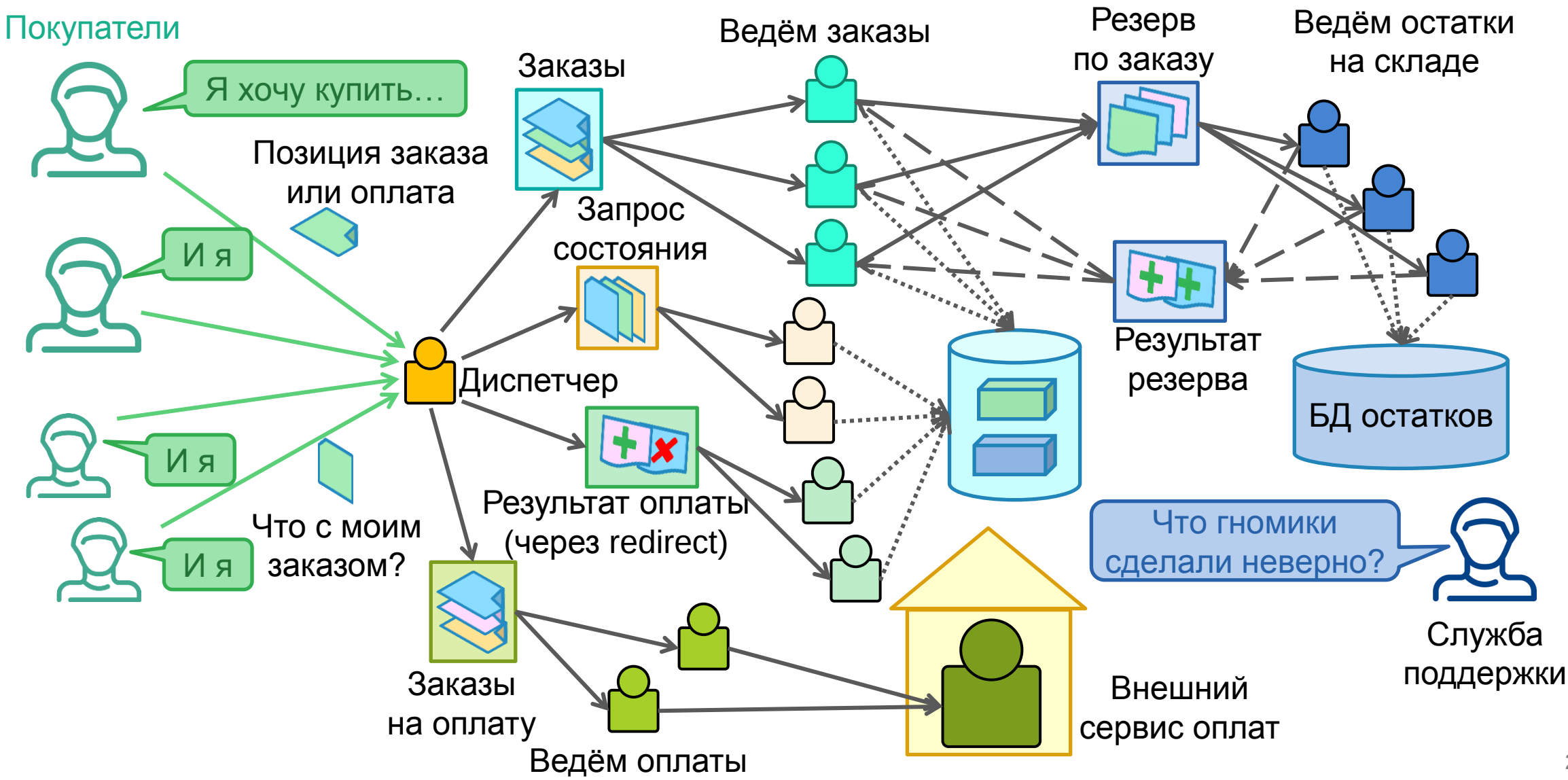
Как обеспечить корректный остаток каждого товара, если сервисов много?

- Очень быстрый гномик: высокопроизводительная БД и железо под узкоспециализированную логику ведения остатков.
- Шардирование по товарам с равномерным рассеиванием по заказам.
- Много гномиков логики остатков и быстрая специализированная БД.
- Очередь на резервирование для равномерной нагрузки.

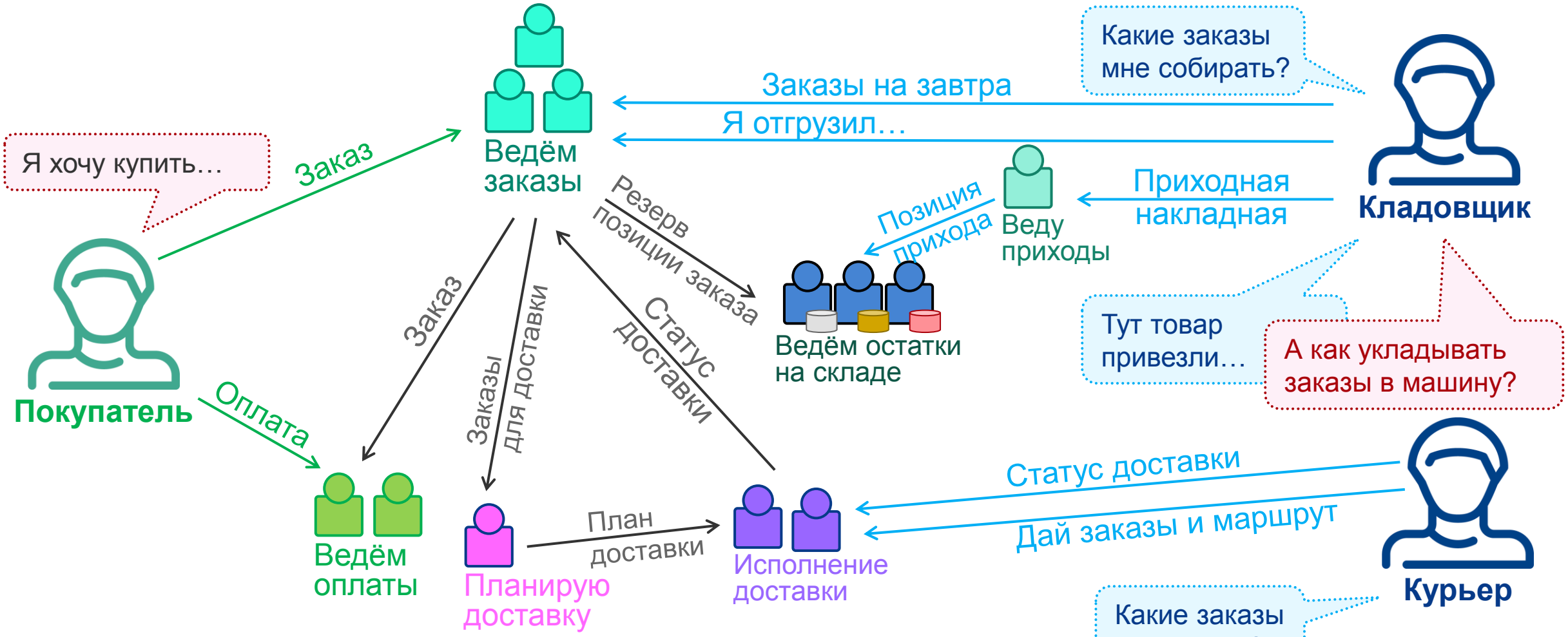
Транзакций нет, а работа асинхронная, и это порождает вопросы:

- Что делать, если зарезервировали не весь заказ?
- Что делать, если резервирование идёт долго?
- Вернее так: переводить ли заказ на оплату, если резерва долго нет?
- А что если сервис заказа упал, а резервы остались?

Ведение остатка на складе — общая БД

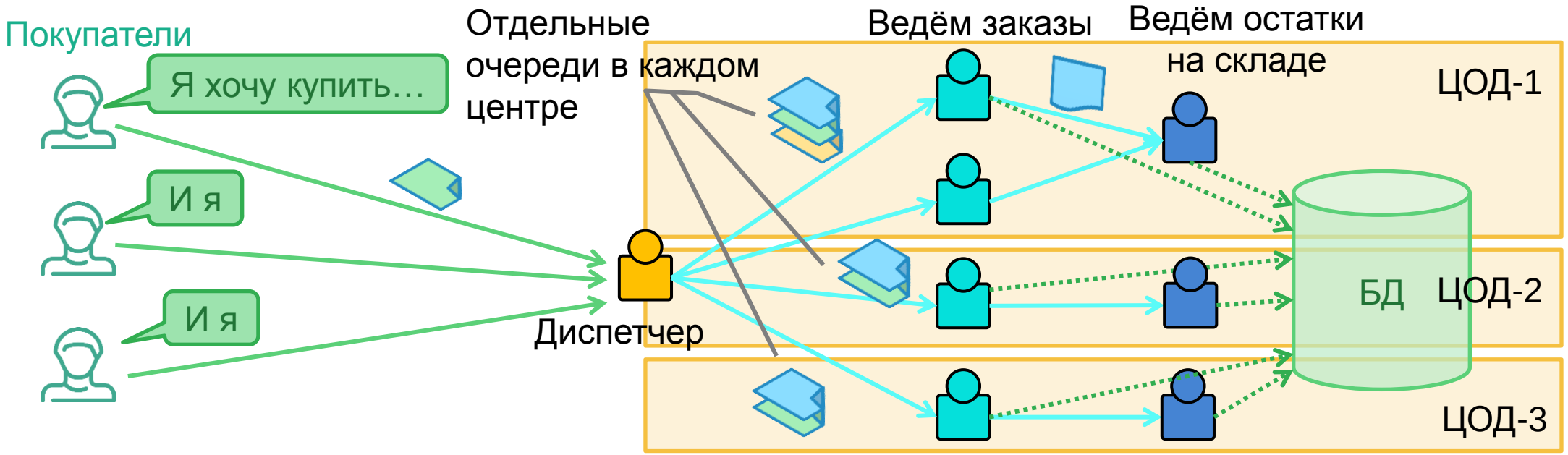


Модель для сервисной архитектуры



Из доклада «Визуальное проектирование масштабируемых приложений»

Надёжность: ноды в разных дата-центрах



- Метафора: ЦОД — дома для гномиков, а ноды — комнаты.
- Обращение в соседнее помещение дольше или невозможно.
- Надо 3 ноды или ЦОДа, чтобы отличить пропажу связи от падения, в метафоре: соседний дом сгорел или телефон не работает.

Итоги

- **Проявляйте связь бизнеса и софта,** ищите схемы, которые хорошо это опишут и покажут важное.
- Монолит и микросервисы — два полюса, реальные архитектуры — гибриды из разномасштабного софта.
- Разбирайтесь в технике взаимодействия сервисов, чтобы проектировать их устойчивое взаимодействие.
- Масштабирование закладывается при проектировании.

21 analyst days

Вопросы по докладу и обратная связь

Максим Цепков
http://mtsepkov.org
@MaximTsepkov

На сайте много материалов по анализу и архитектуре, Agile и менеджменту самоуправления, моделям soft skill, мои доклады, статьи и конспекты книг

CUSTIS

Вакансии

Пишите на hr@custis.ru, подходите с вопросами!