CUSTIS

Что такое архитектура и как она влияет на тестирование

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

http://mtsepkov.org

24–25 октября 2025
Москва

37 sqa days

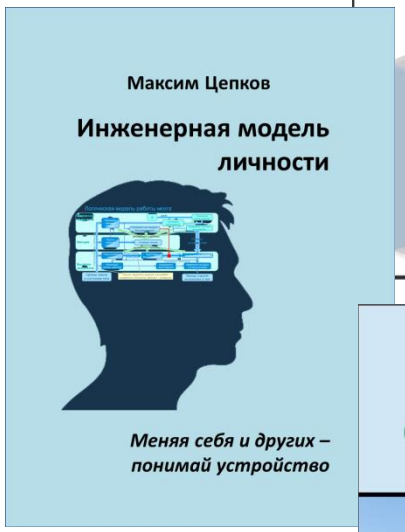
Немного обо мне

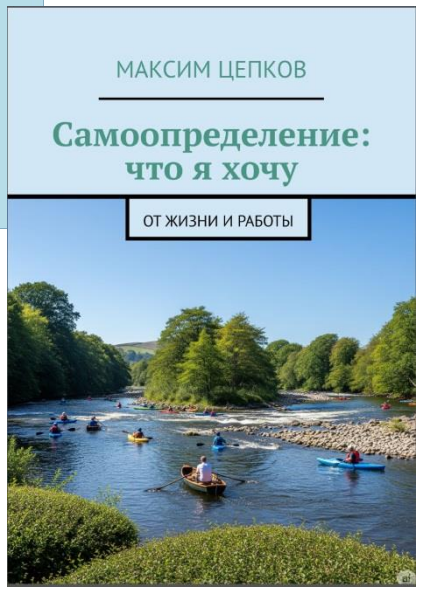
Создание и внедрение больших корпоративных систем (30+ лет)

- Знание практик операционного управления и ведения проектов в коммерческих и государственных организациях и банках.
- Опыт управления проектами в IT — 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 самоуправления: бирюзовые организации, холакратия и социократия (**книга**, статьи и выступления).
- Модель спиральной динамики (с 2013) 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?

- Есть два подхода к архитектуре:
  1. Монолит
  2. Микросервисы

  И их комбинация.
- Архитектура принципиально меняет тестирование приложений, особенно по обеспечению устойчивости и масштабированию под нагрузкой.
- Мы на простом примере разберём различия.
- И немного поговорим о более сложных вариантах.

Введение: от бизнеса к архитектуре софта

Уровни представления (интернет-заказ)

Деятельность

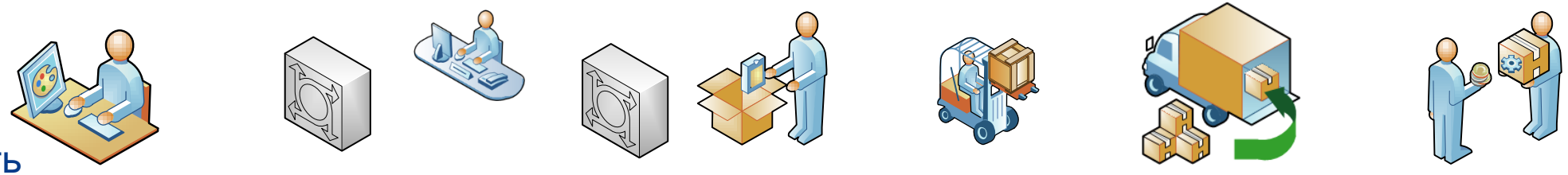
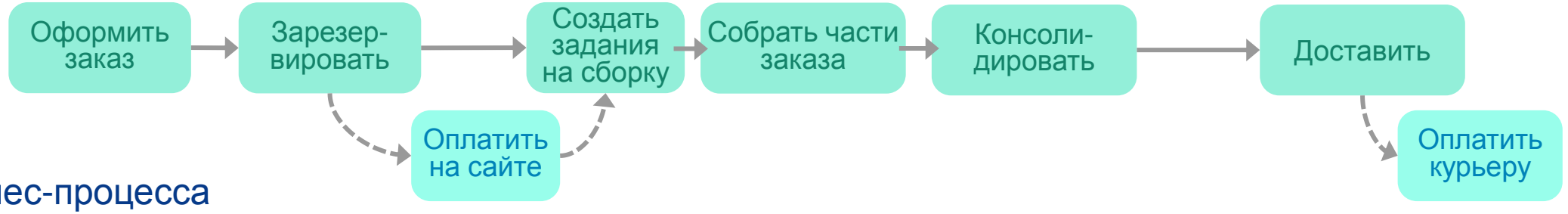


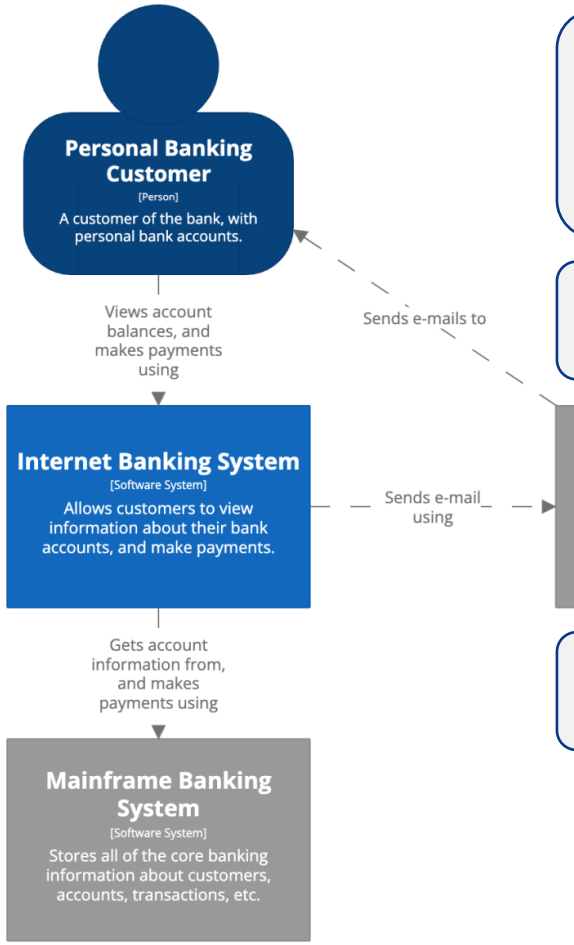
Схема бизнес-процесса



Объекты и их состояния

Руководства по системе

Контекстная диаграмма C4 Model — система в окружении

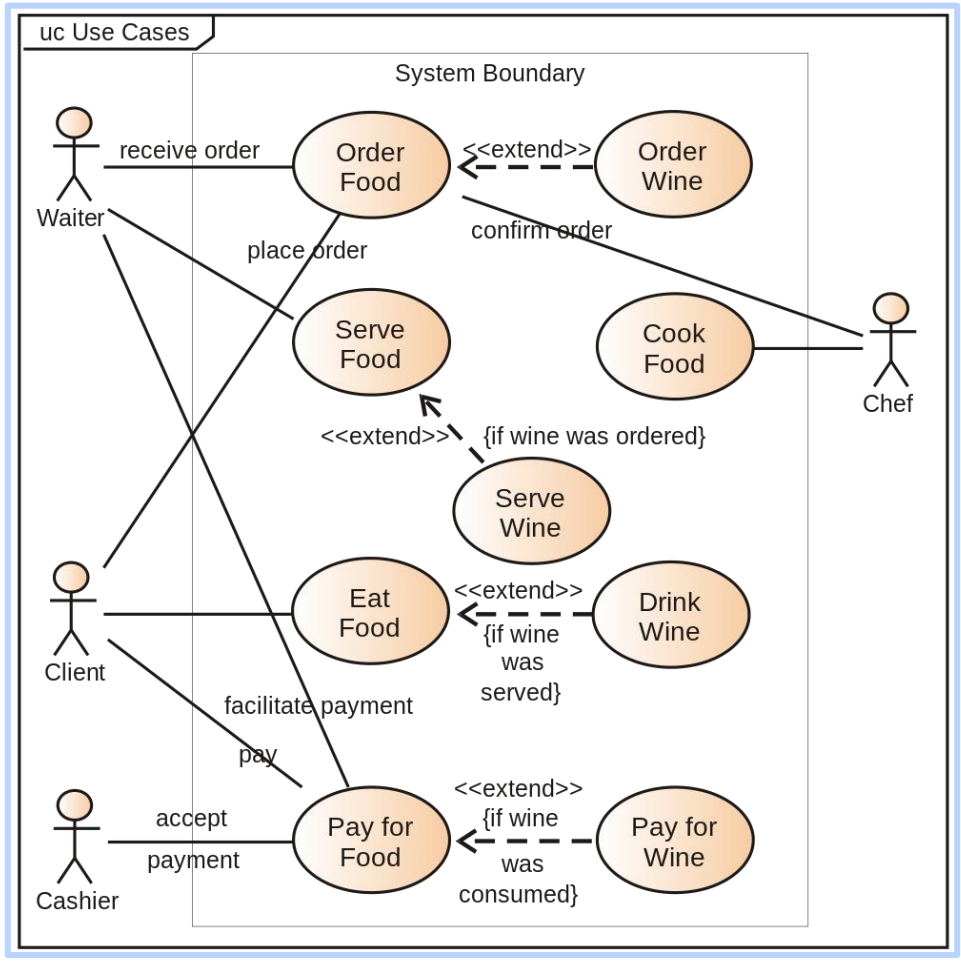


В центре — наша система, вокруг — пользователи и взаимодействующие системы

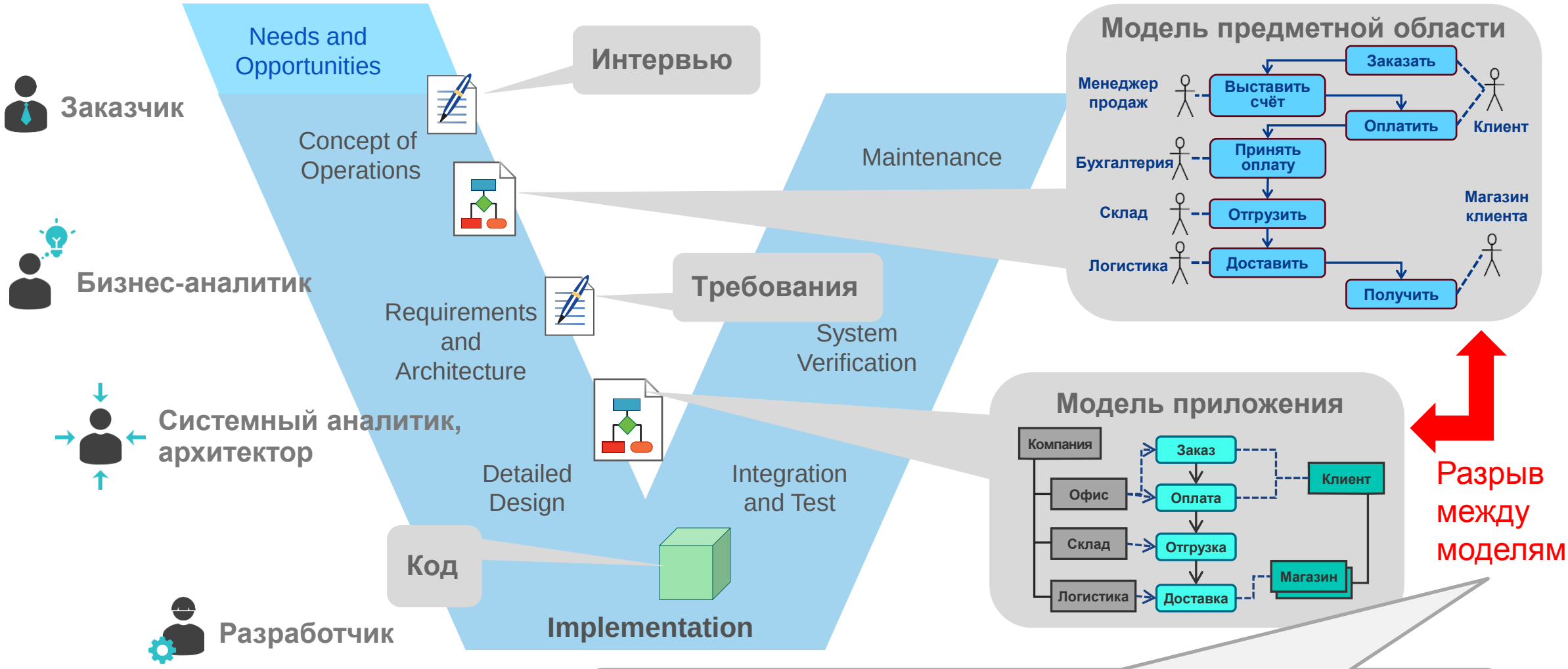
Стрелки — акты взаимодействия

Множество внешних систем — смена парадигмы на сервисы

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



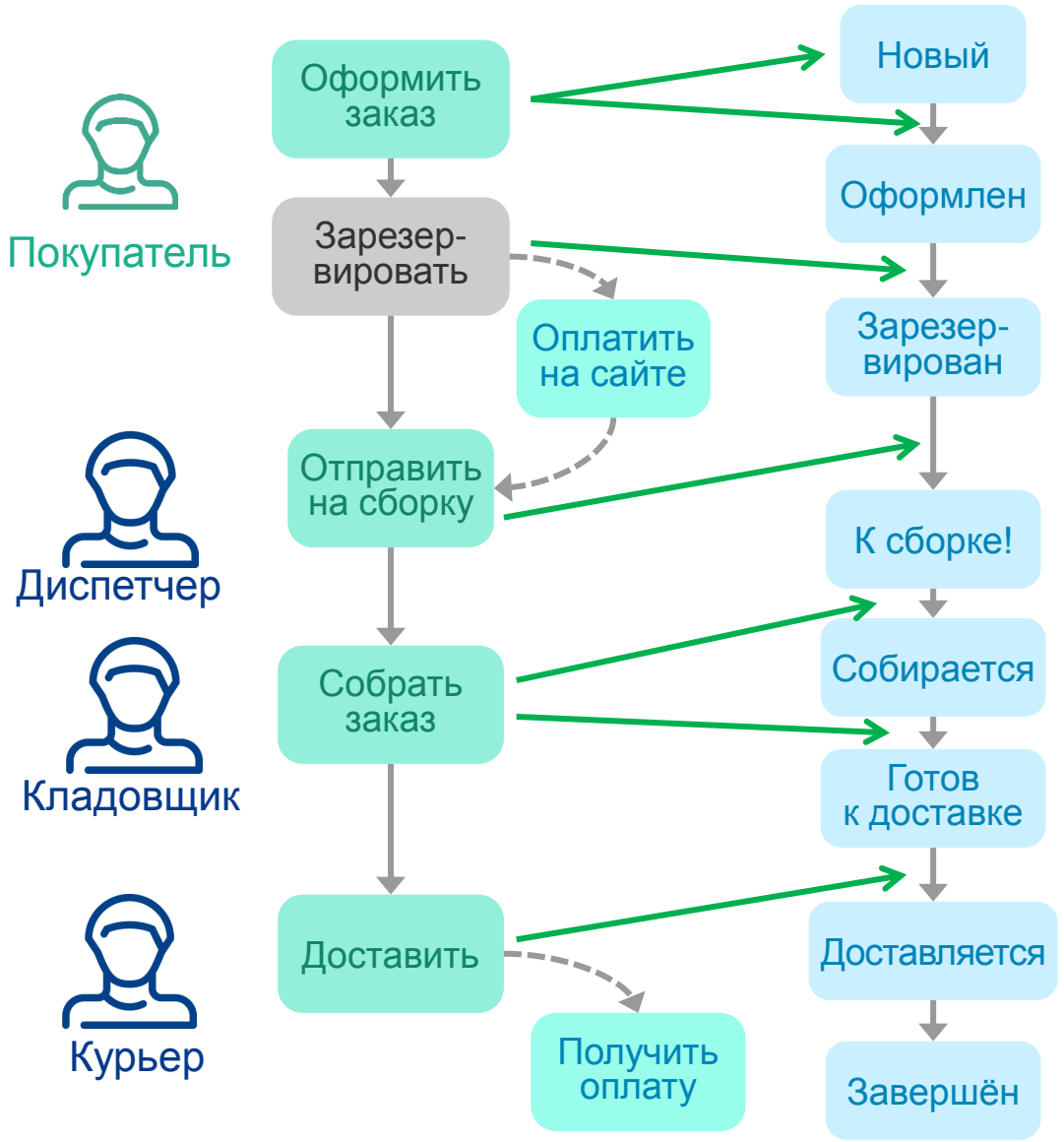
Поддержка артефактами (классика)

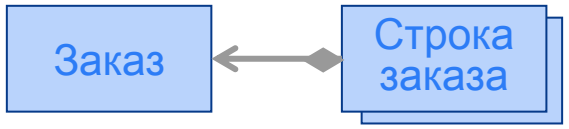


Проблема: каждый уровень отдельно сложно изменять и поддерживать соответствие.

Монолит vs сервисы

Монолит — процесс через workflow





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

Сложные кейсы тестирования:

- Диспетчер все заказы отправил на сборку.
- Сборка заказа кусочками.
- Что если при сборке чего-то нет?
- Что если покупатель отказывается, начата сборка или при получении?
- Работа с платёжным гейтом.

Усложнения:

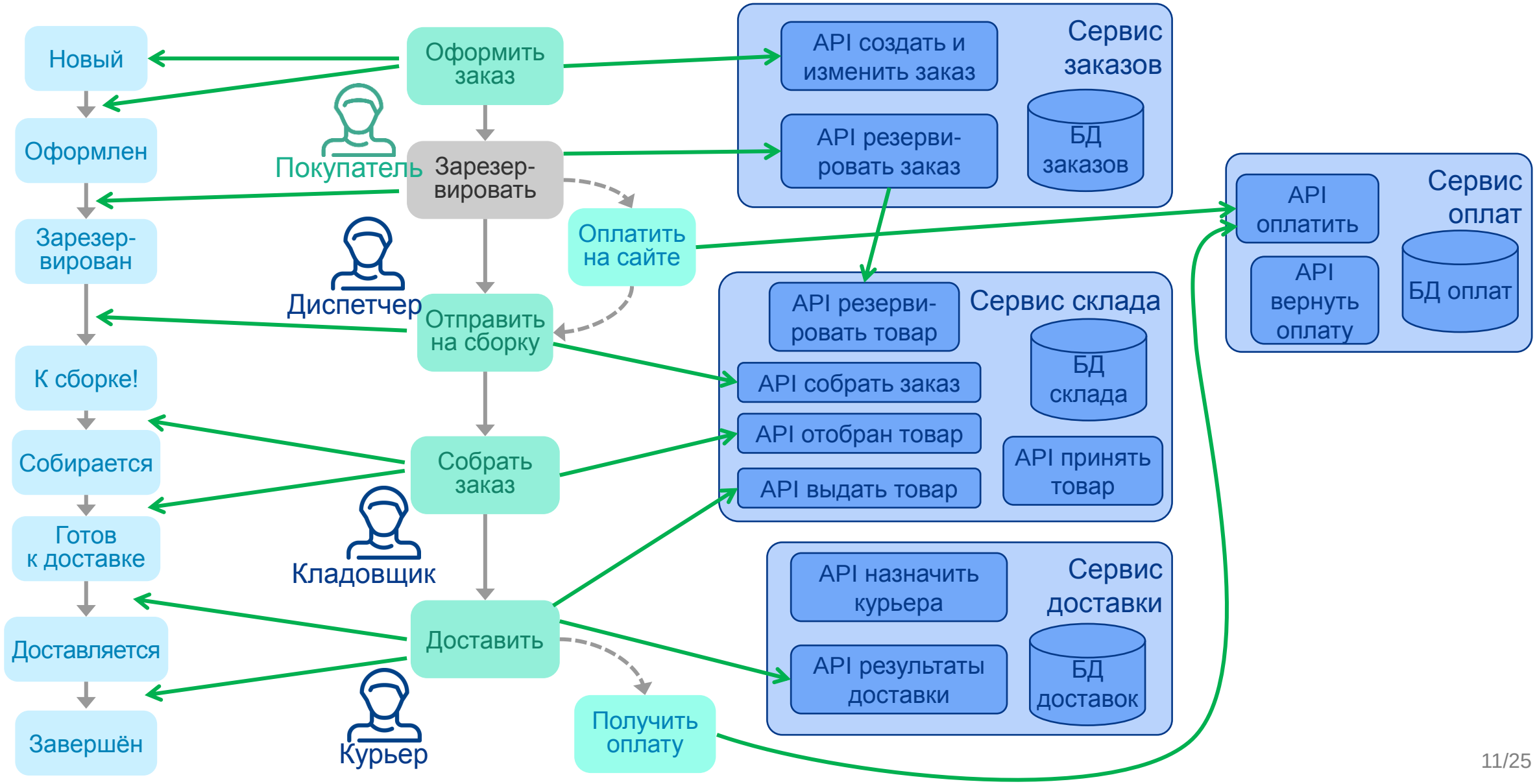
- Собирают несколько кладовщиков.
- Доставка частями.

Подходы к тестированию

- Выполнение запроса пользователя на уровне БД в одной транзакции гарантирует консистентность данных, о ней не надо заботиться при ошибках.
- Нам достаточно проверить:
  - работу интерфейса в 1-2 успешных и ошибочных сценариях;
  - работу на уровне сервера через API сложных сценариев на конкретных шагах workflow;
  - взаимодействие с внешним сервисом оплаты, в том числе разные сложные случаи.
- Проблемы могут быть связаны с кешированием данных в интерфейсе и на сервере приложений, но это не критично и обычно решает базовый уровень.
- Масштабирование обеспечивается железом и СУБД, надо следить за узкими горлами по блокировкам, но всё это обычно задача специалиста по БД.

Невозможность обеспечить масштабирование на уровне БД привела к развитию (микро)сервисной архитектуры в public web.

Монолит vs сервисы: workflow vs API



Что изменилось в сервисной архитектуре

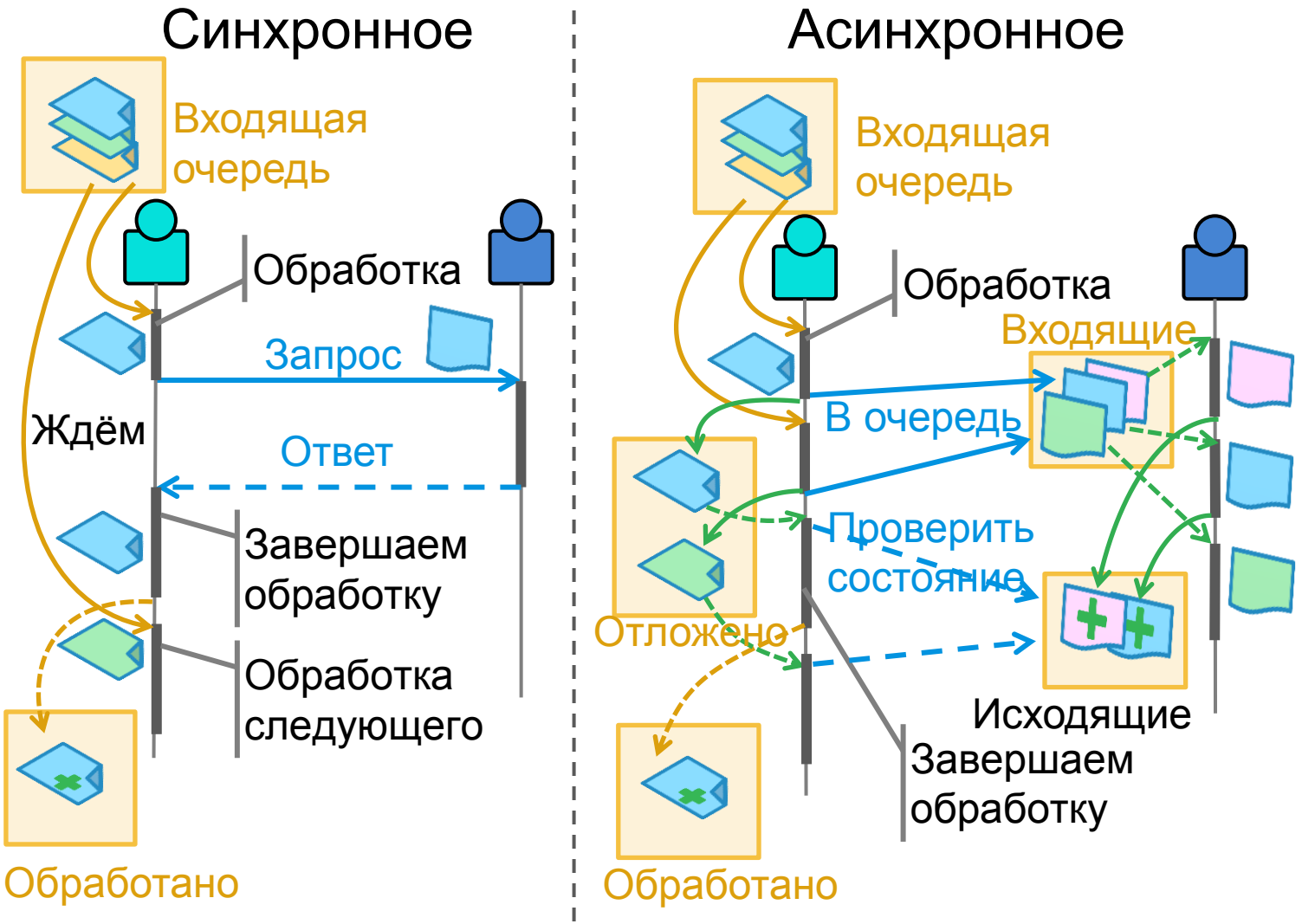
- Каждый бизнес-запрос обрабатывают много сервисов.
- Транзакционность и консистентность обеспечивается в приложении.
- Поднимают много экземпляров сервиса, каждый может упасть по ошибкам или блокировкам, а система должна работать устойчиво.
- Асинхронные сообщения и очереди для выравнивания производительности 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.

Пример: резервирование заказа

- Зарезервировать товар для доставки курьером на одном из складов или собрать с нескольких и проверить возможность доставки к сроку.
- Сценарии тестирования:
  - Основной: весь товар в заказе найден на одном складе.
  - Товар найден на двух складах, требуется консолидация для курьера, её успеем сделать.
  - Товара не хватает, покупатель должен принять решение, берёт ли неполный заказ.
  - Товар найден, но собрать заказ для курьера к сроку не успевают – переносим?
- Монолит: в сценариях достаточно проверять состояние базы данных до и после запроса, предполагая, что при ошибке будет откат состояния.
- Микросервисы вносят усложнения:
  - Товары резервируются по одной позиции, возможно зависание резерва, его надо снимать.
  - Надо проверять состояния базы у нескольких микросервисов.

Варианты межсервисного взаимодействия



Тестировать надо иначе

Классические постановки и тесты рассчитаны на атомарное исполнение: если в интерфейсе и базе данных ожидаемое, то функция работает.

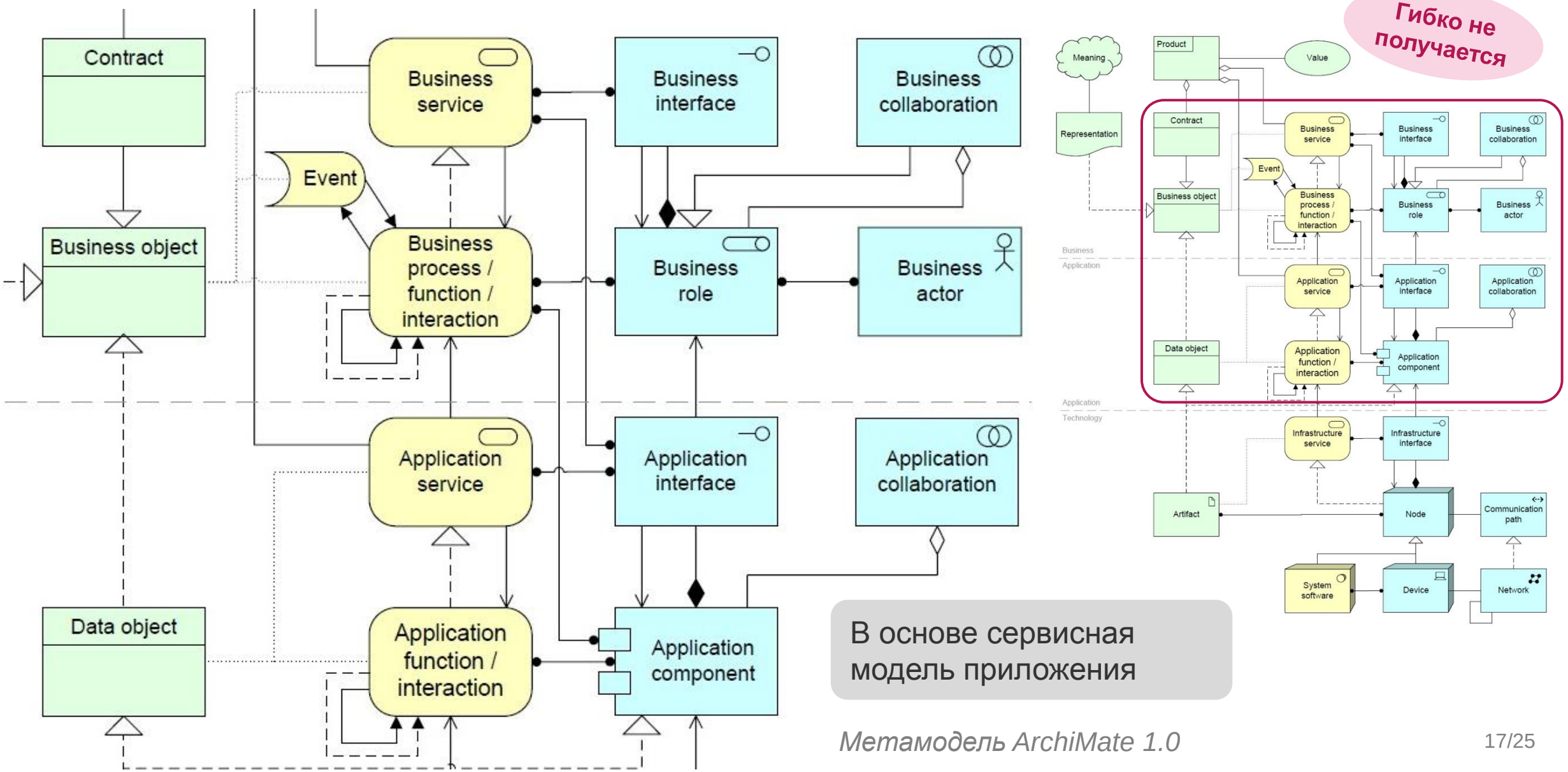
В новой архитектуре приложений это не так:

- Обработка организована через асинхронные сообщения.
- Постановка в очередь не даёт гарантий обработки.
- При распределённой обработке операция может быть выполнена частично.

Метафора гномиков позволяет описывать устройство приложений и прорабатывать вопросы устойчивости и масштабирования.

Масштабирование сервисов

Archimate — гибкая связь бизнеса и софта

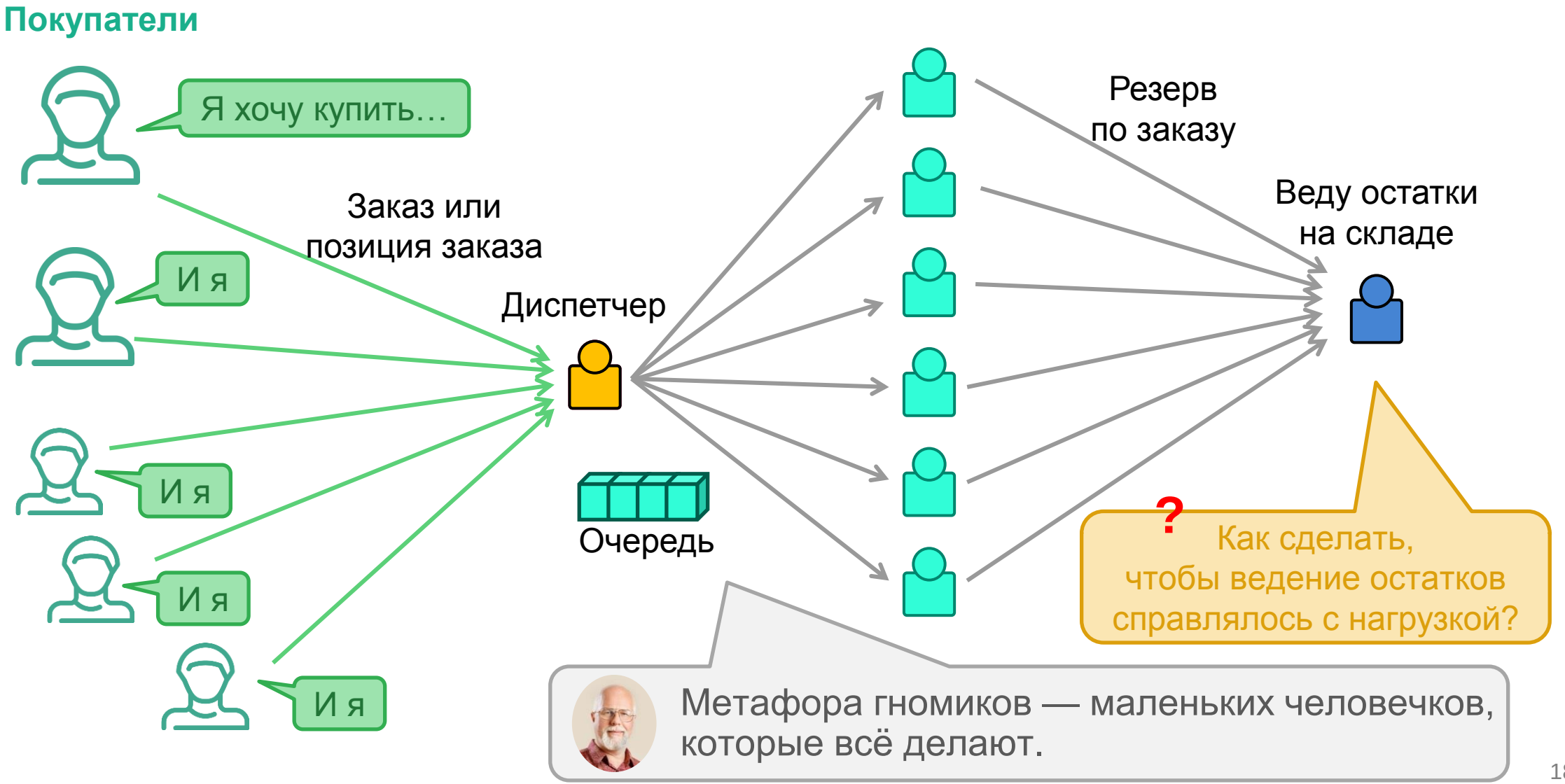


Гибко не получается

В основе сервисная модель приложения

Метамодель ArchiMate 1.0

Масштабирование сервисов



Варианты ведения остатка на складе

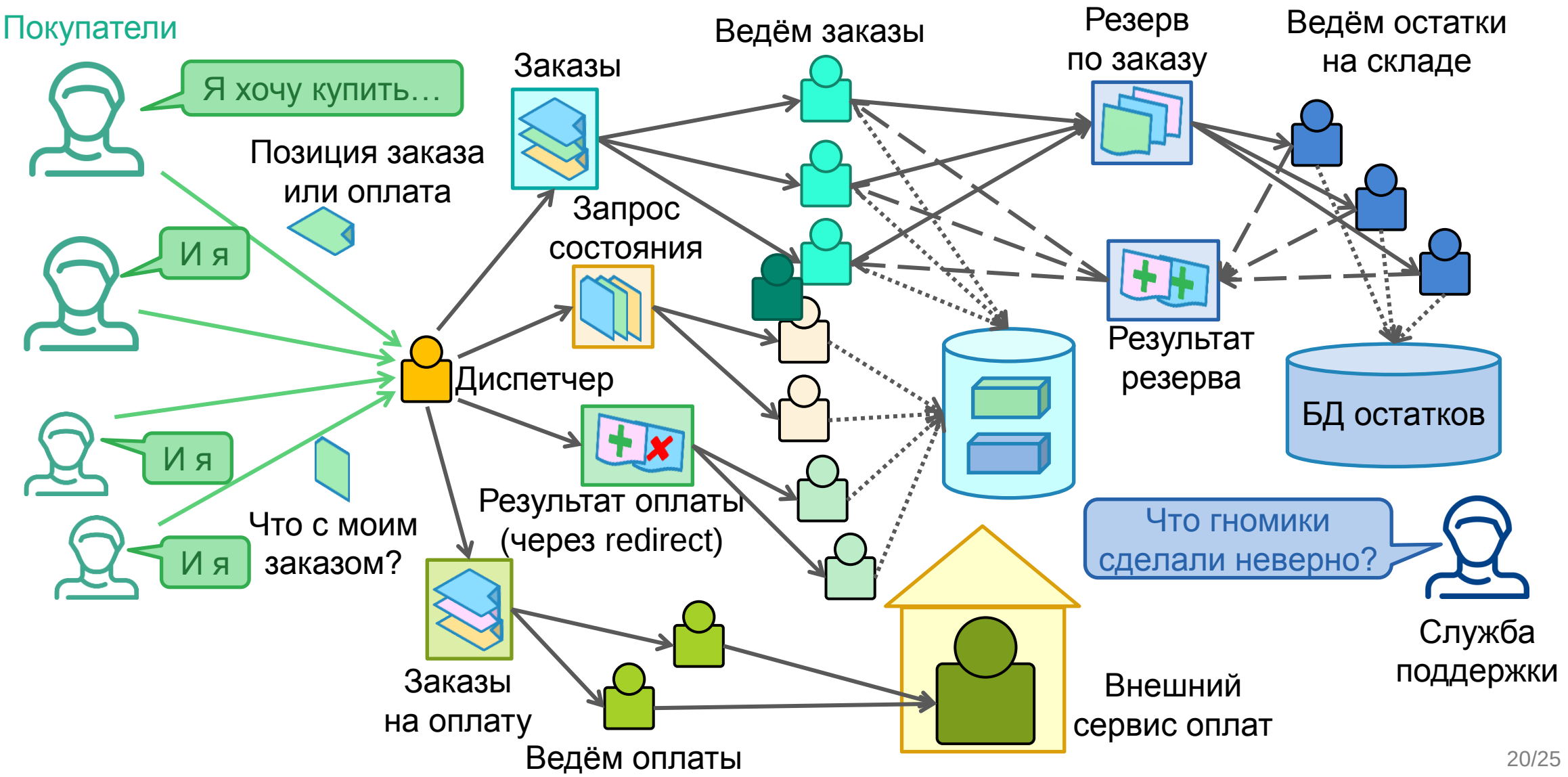
Как обеспечить корректный остаток каждого товара, если сервисов много?

- 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.

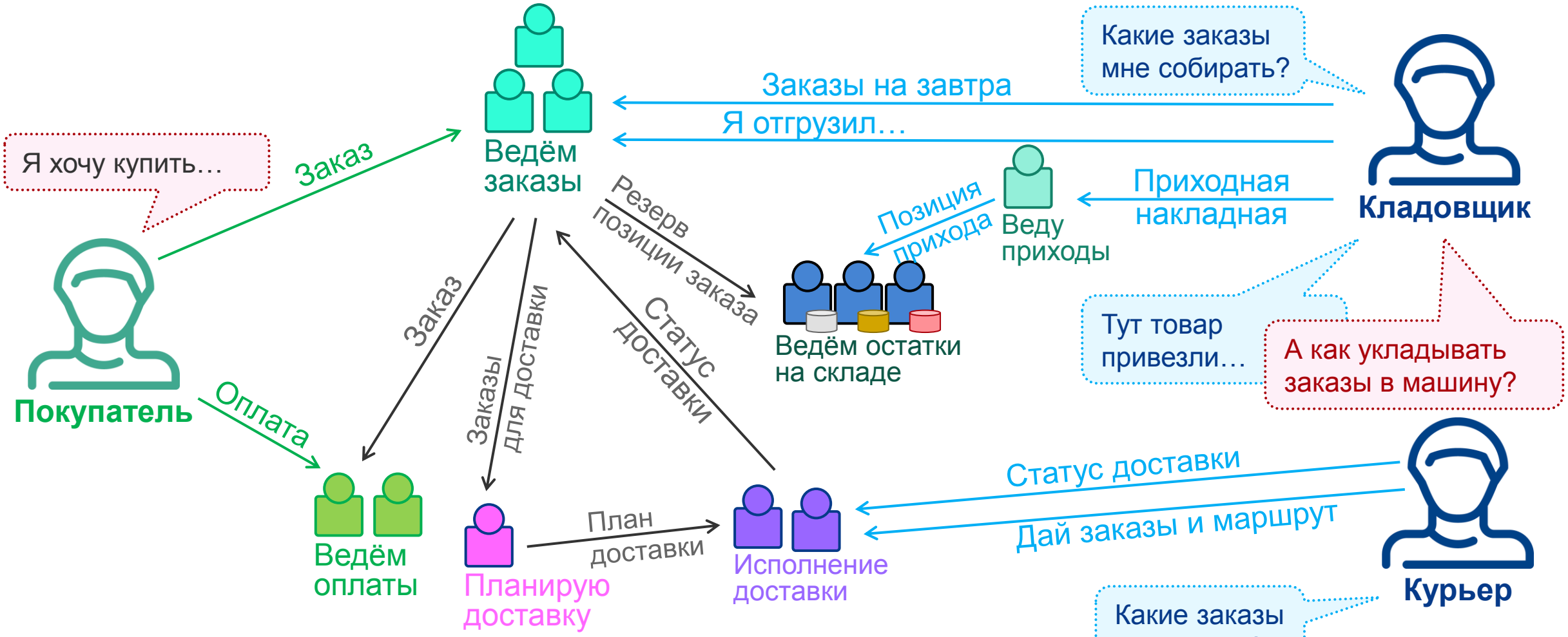
Транзакций нет, а работа асинхронная, и это порождает вопросы:

- 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



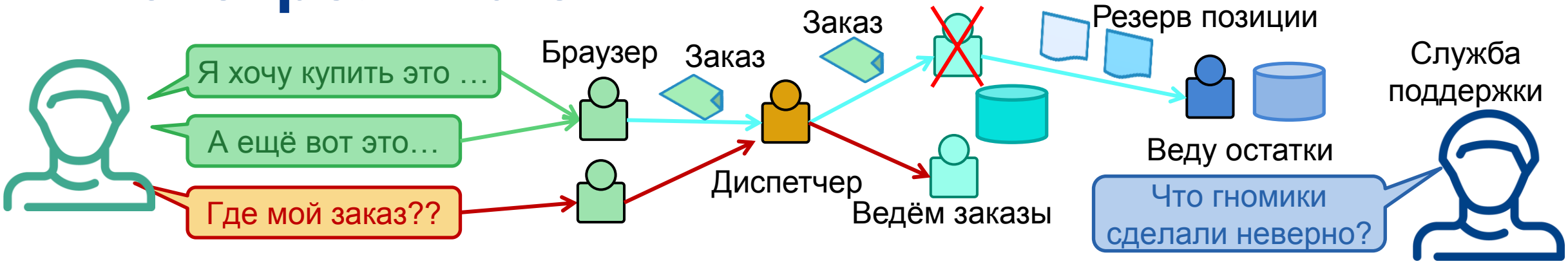
Модель для сервисной архитектуры



Из доклада «Визуальное проектирование масштабируемых приложений»

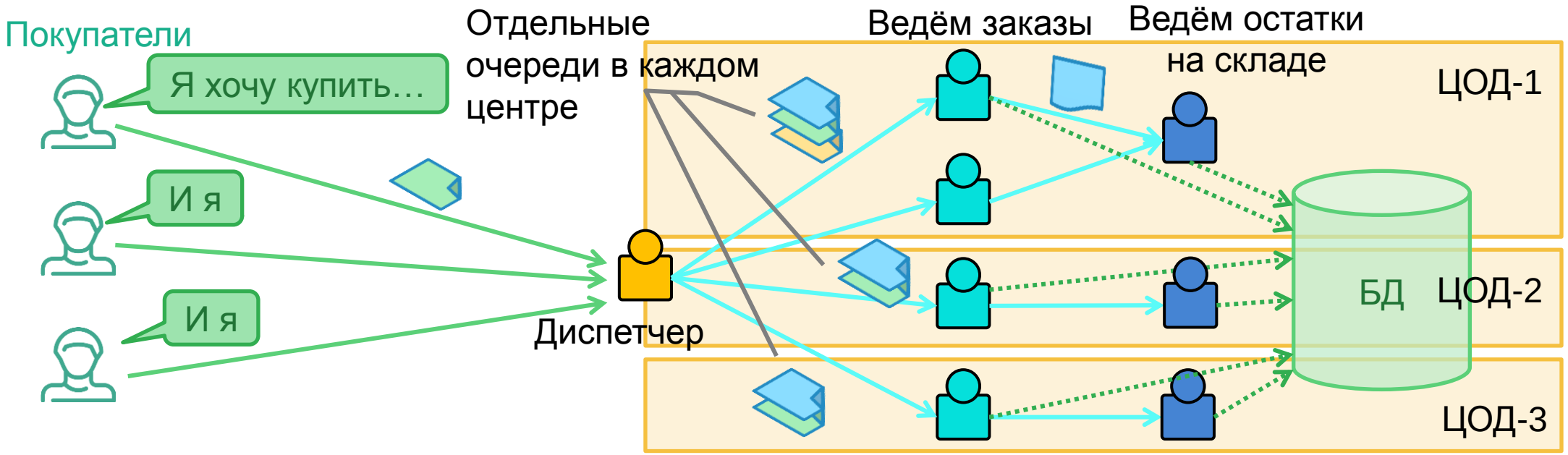
Устойчивость и надёжность в сервисной архитектуре

Устойчивость: гномики 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 проверки:
  - 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.

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

- Монолитная архитектура решала ряд задач «из коробки», хотя проблемы интеграции при сбоях возникали и там.
- В сервисной архитектуре надо хорошо представлять устройство приложения, чтобы его тестировать.
- Тестировщику надо уметь придумывать сложные кейсы с учётом конкретной архитектуры приложения.
- Тестировщику стоит помнить о реальных действиях пользователя в физическом мире при работе с системой.

37 sqa 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!