ANALYST DAYS #11
analystdays.com

Модели предметной области для разных парадигм программирования

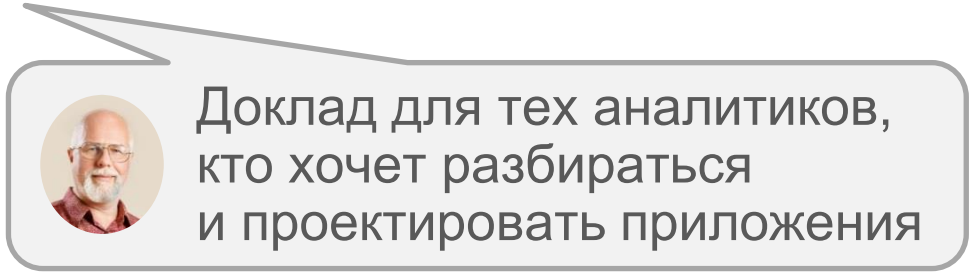
Максим Цепков

IT-архитектор и бизнес-аналитик,
навигатор и эксперт по миру Agile,
бирюзовых организаций и Спиральной динамике
http://mtsepkov.org

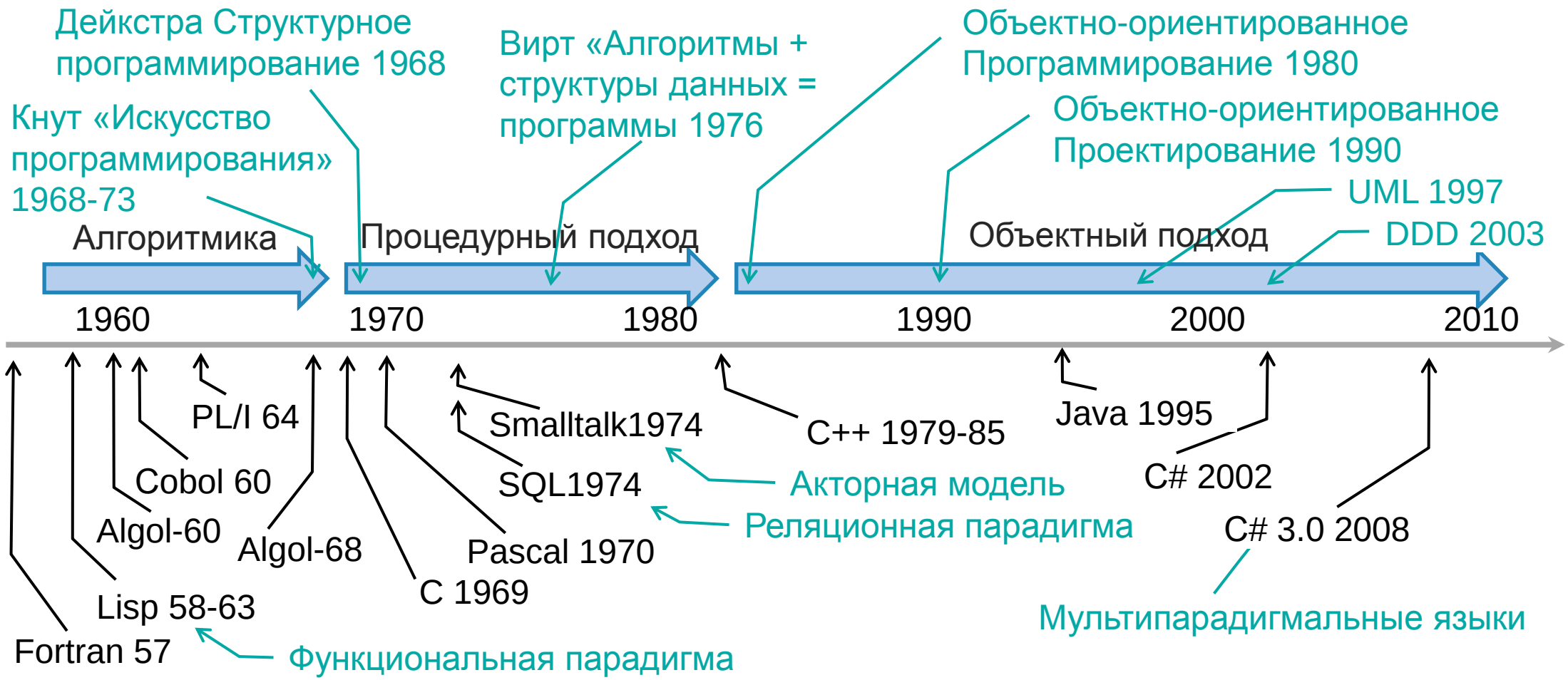
Analyst Days-11 в Москве
9-10 октября 2020

О чем будет рассказ

- История программирования породила много парадигм
- С 90-х основным был ООП и другие были не слишком важны
- Public web все изменил: микросервисы, messaging, акторные модели – надо заново разбираться, чтобы проектировать



История программирования и проектирования



** Мои статьи «История IT. Когда компьютеры были большими...» и «История IT. ООП»*

Процедурный и объектный подход – пример

Задача – интернет-магазин: заказы, оплата, склад, отгрузка, доставка

- Процедурный подход
  - Таблицы товаров, заказов, платежей, остатков на складе, курьеров, доставок
  - Алгоритмы: фиксация оплаты, назначение даты доставки, планирование курьеров
  - Интерфейсы: какие экраны, какие данные показываем, какие из них действия
- Объектный подход
  - Объекты – товар, заказ, платеж, курьер. Доставка – отдельный объект в заказе?
  - Алгоритмы – в методах, инкапсуляция внутренней логики объектов
  - Интерфейсы: витрины каких объектов представляем, какие методы доступны

Если есть доставка самовывозом и курьером, то в процедурном подходе особенности – во всех алгоритмах, а во втором – делаем подтипы

Процедурный и объектный подход

- Процедурный подход: проектируем структуру БД, интерфейсы и алгоритмы обработки. Иногда – процедуры API backend и API RPC
- Объектный подход: типы и статусы объектов, методы бизнес-логики, **naked object** интерфейсы – витрины объектов и методы, REST API
- Разница – где бизнес-логика, в методах объектов или отдельно
- В реализации может быть анемичная модель транспортных объектов и соответствующие тем же объектам контролеры для бизнес-логики
- **DDD** распространил объектный подход на модель предметной области: вместо словаря мы делаем онтологию понятий и связей, декомпозируя область на фрагменты и применяя методы инкапсуляции, наследования и другие – концепция **bounded context**

Почему же процедурный подход живет сейчас?

- C++ появился в начале 1983, а UML – в 1997, 15 лет разницы
- Программисты оценили объекты и внутри приложений они есть
- Но ERP начала 90-х (SAP, 1C) проектировали в процедурном подходе
- Появилась школа такого проектирования, по мотивам классики Вирта
- SAP, 1C и другие до сих пор популярны – подход воспроизводится

В 1990-е были текстовые терминалы и о usability никто не думал.
Поэтому с Usability и UX во многих этих системах проблемы до сих пор.

Public web и мобилки

Развитие public web и мобильных устройств

- Первые системы представляли контент, а гипертекст – не объектный
- Разделили контент и дизайн, добавили бизнес-логику – MVC и MVVM
- Революционное технический прогресс
  - Высокая производительность и доступность интернета породила интернет-магазины и услуги, социальную жизнь в интернете
  - Мобильные сети и смартфоны сделали интернет доступным постоянно, и возникла новая социальная и деловая среда, в которой сейчас живут все
- Изменения в приложениях
  - Сложные интерактивные приложения в браузере
  - Кластерный бек-энд и распределенные базы данных, NoSQL-хранение
  - Мультипарадигмальные языки (C# 2008) и выход из объектной парадигмы

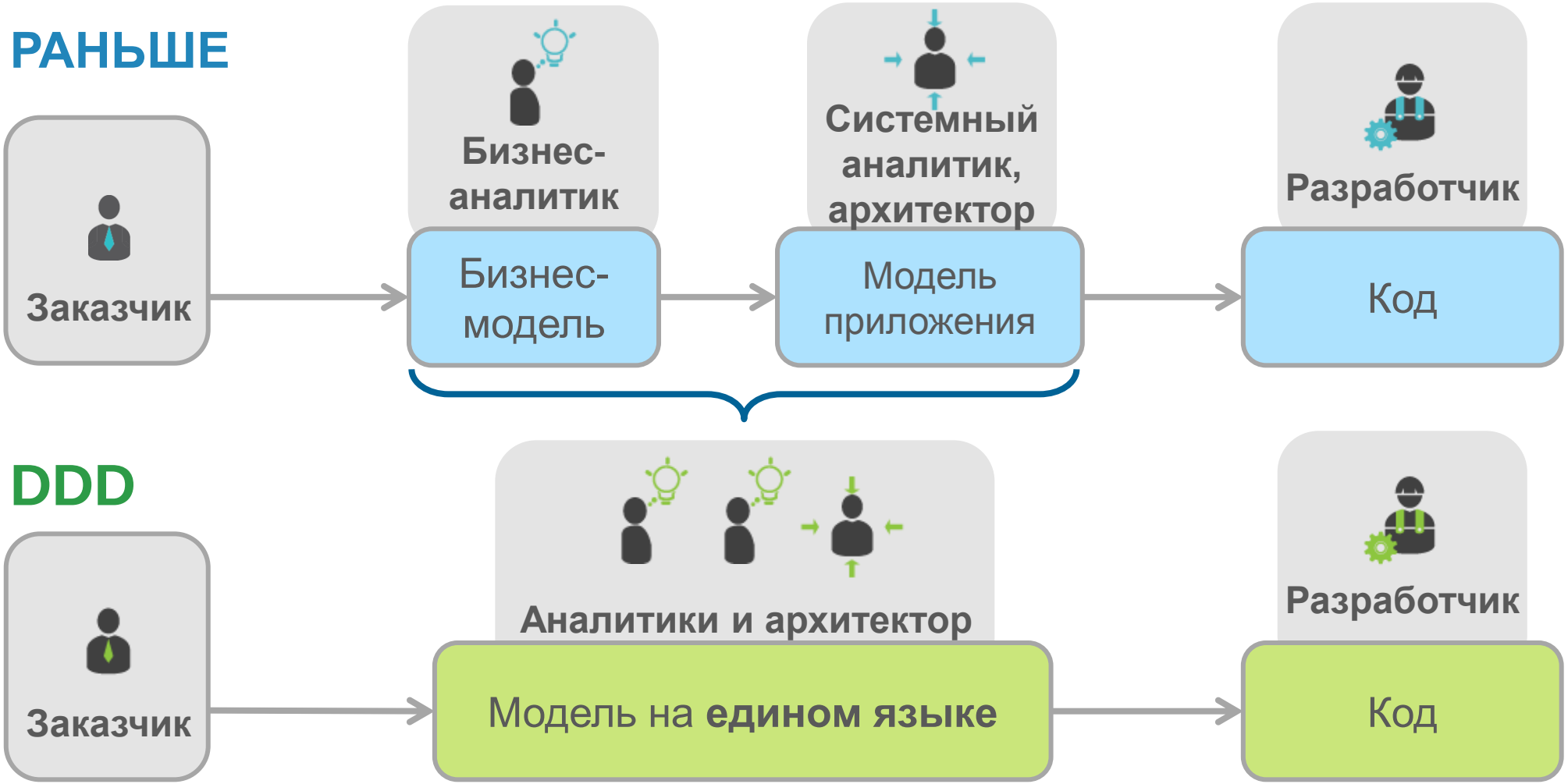
Что поменялось в архитектуре

- Сервисные и микросервисные архитектуры,
  каждый бизнес-запрос обрабатывает 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

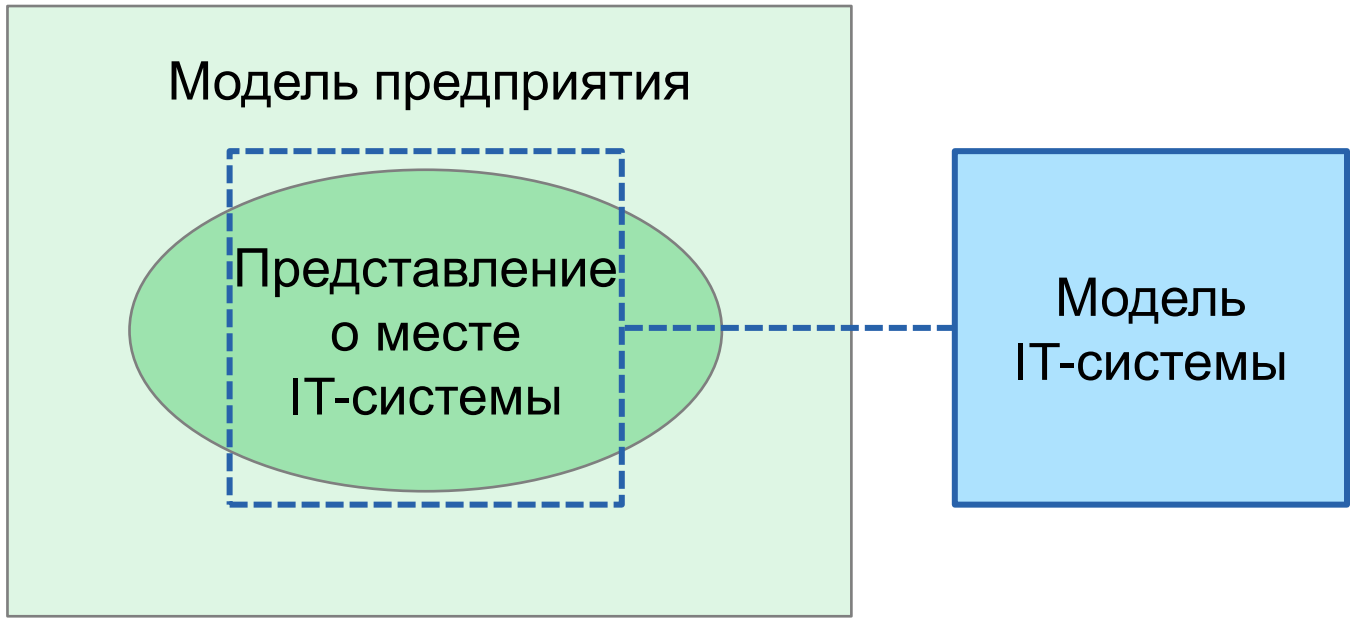
Проектировать надо иначе

- Классические постановки не подходят для нового мира приложений
  - Нет консистентного и связного хранения таблиц или объектов – это не так
  - Процедуры и методы выполняются распределенно, через сообщения
- Надо заново искать место аналитика в разделении труда

DDD – единая язык и единая модель приложения



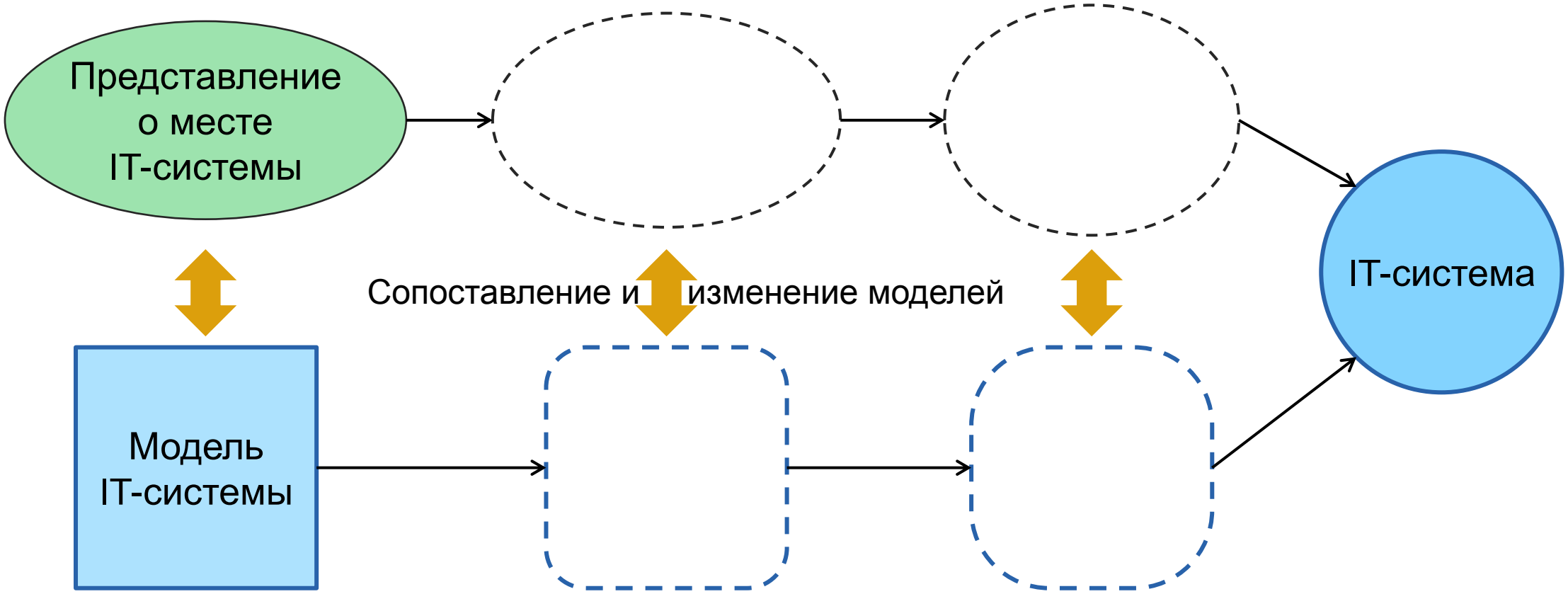
Зачем нужен единый язык?



Модель системы не соответствует представлению бизнеса
о ее месте в модели предприятия.

Не то чтобы совсем не попал,
но только не попал в шарик…

Итерационное развитие модели



Единый язык позволяет совместить модель системы
с представлениями бизнеса о ее месте

Проектировать надо иначе

- Вариант-1 – отдать все разработчикам, писать user story и use case
  - 😐 Возвращаемся к роли бизнес-аналитика
  - ☹ Не можем проверить реализуемость и целостность постановки
- Вариант-2 – использовать постановки только для передачи контекста
  - ☺ Целостность и реализуемость постановки проверяем, пусть по-старому
  - ☹ Постановка передает задачу разработчику и теряет актуальность
- Вариант-3 – придумать новые метафоры и подходы к постановкам
  - ☺ Проверяем целостность и реализуемость постановки
  - ☺ Постановка соответствует фактической реализации и живет дальше
  - ☹ Для этого надо разобраться и понять, как устроены новые технологии

А можно не разбираться? Нельзя!

Зачем я лишь о том всё время думаю,
как сделать, чтоб не думать ни о чём?

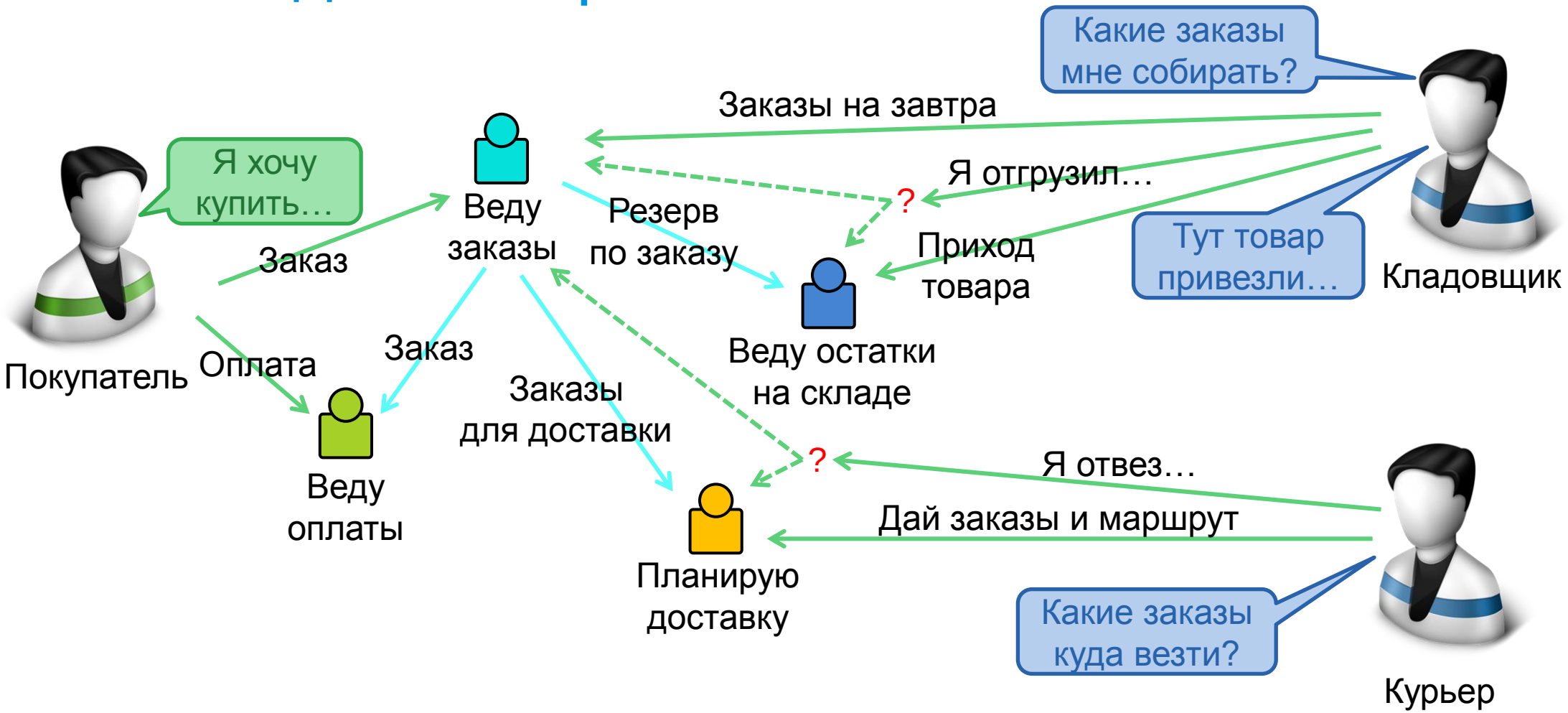
Михаил Щербаков

- Мечта: базу данных визуально проектирует аналитик без кода
- Мечта: универсальный конструктор ERP, который настраивает бизнес
- Или хотя бы универсальный документооборот, BPM и т.п. – UML
- Мечта: делим базу данных по серверам не меняя код – dblink
- Мечта: делим монолит по серверам не меняя код – SOAP
- Мечта: база данных или сервер, масштабируемые без проблем

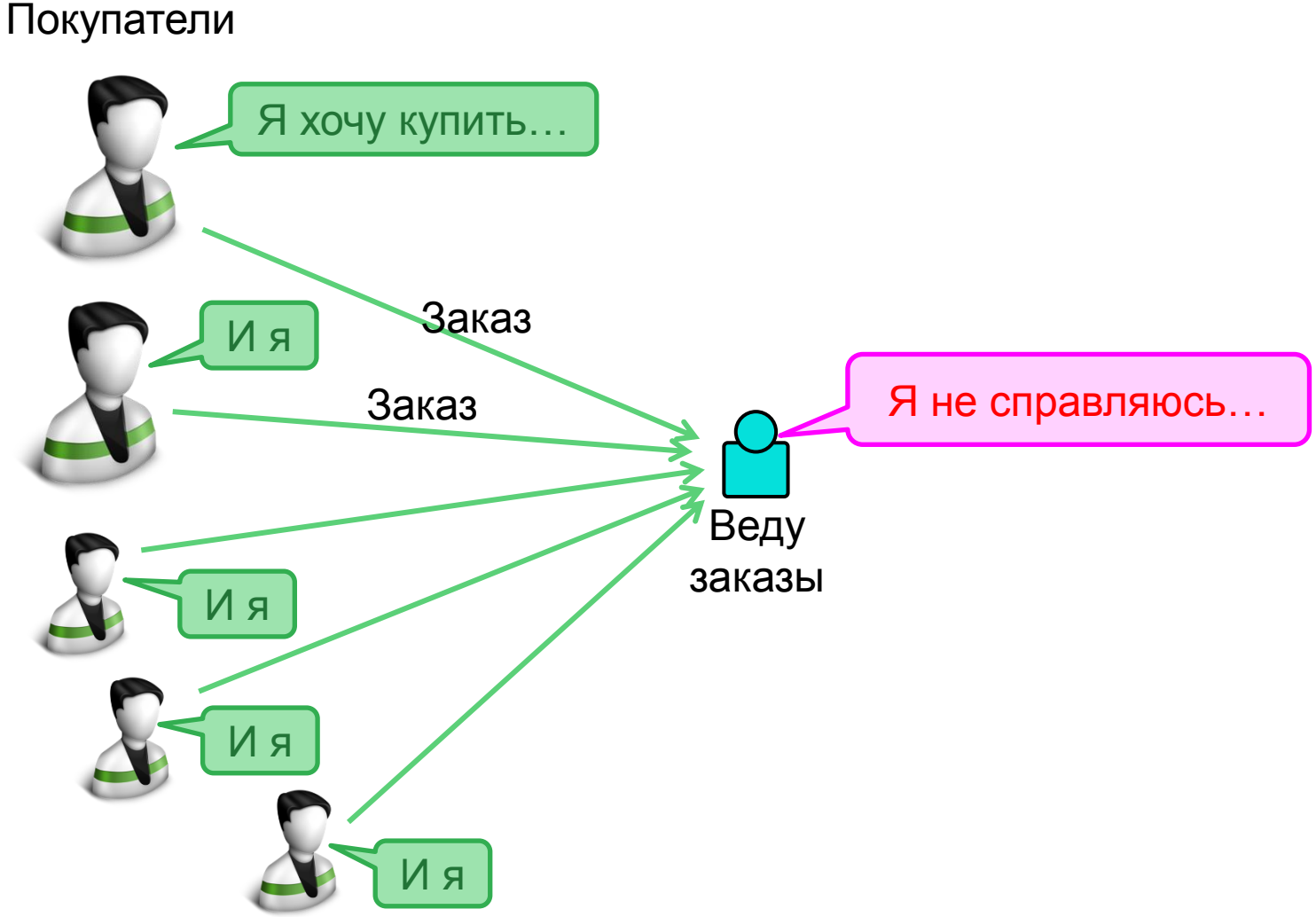
Все эти и многие другие мечты провалились.
Думать – надо!

Метафора гномиков –
маленьких человечков,
которые все делают

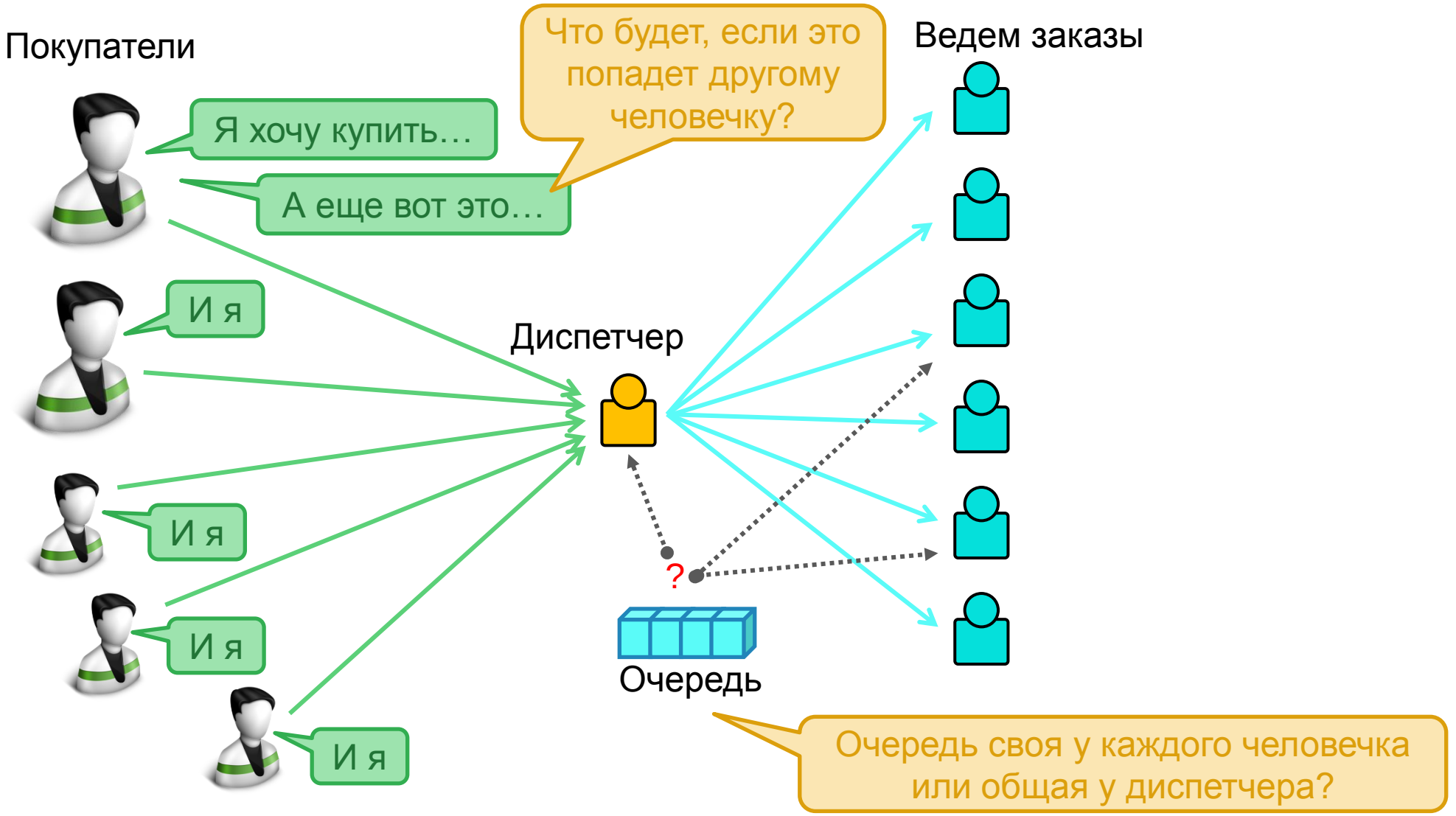
Гномики для интернет-магазина



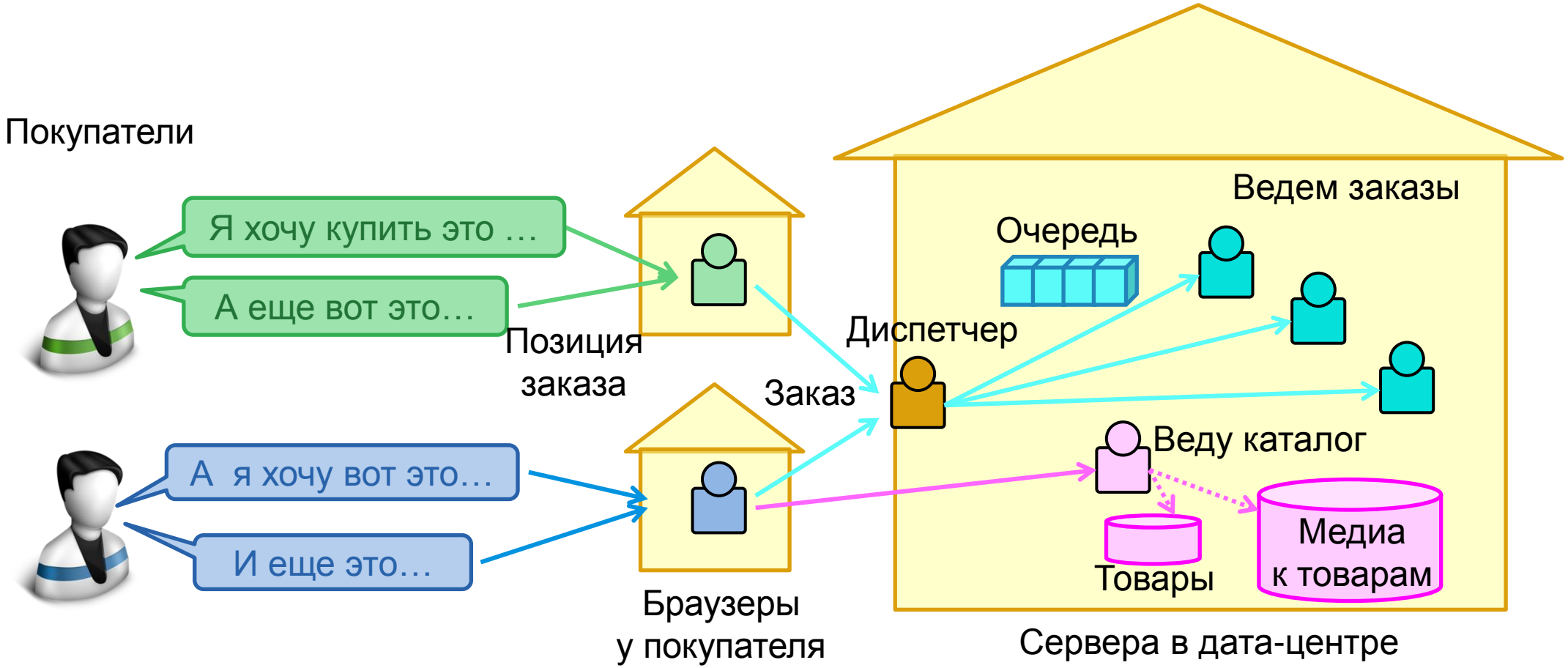
Но у нас много покупателей…



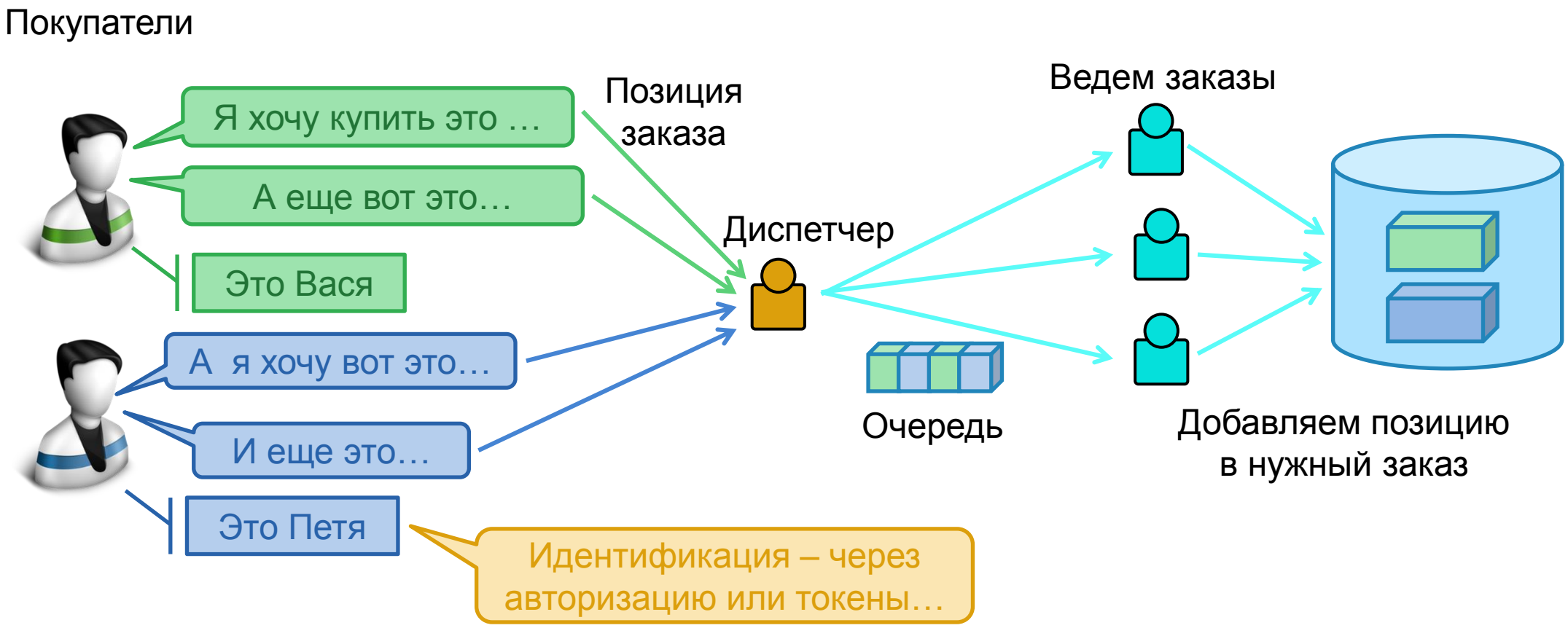
Делаем кластер сервисов приемки заказов



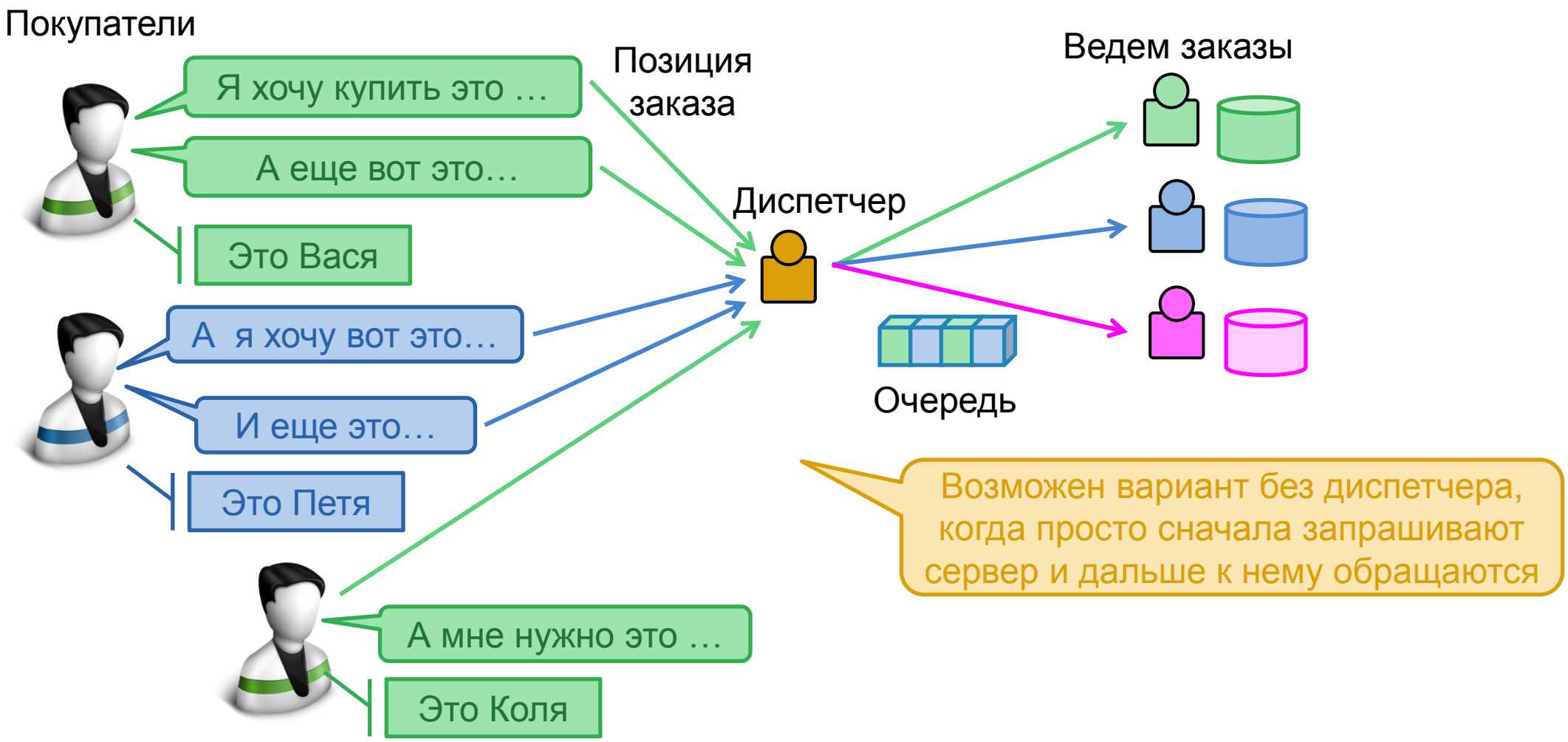
Собираем заказ в браузере



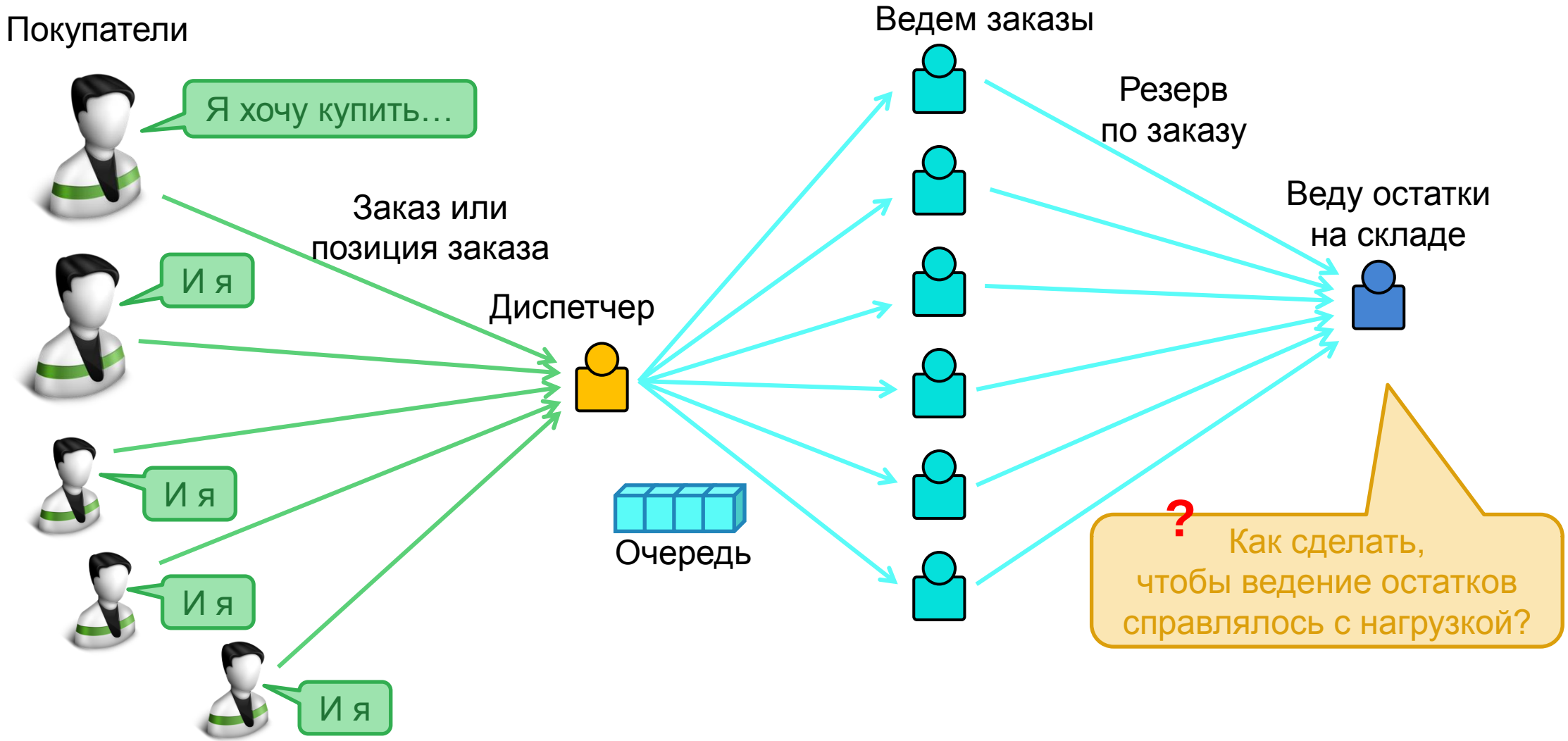
Общая база данных



Шардирование покупателей



Ведение остатка на складе – проблема



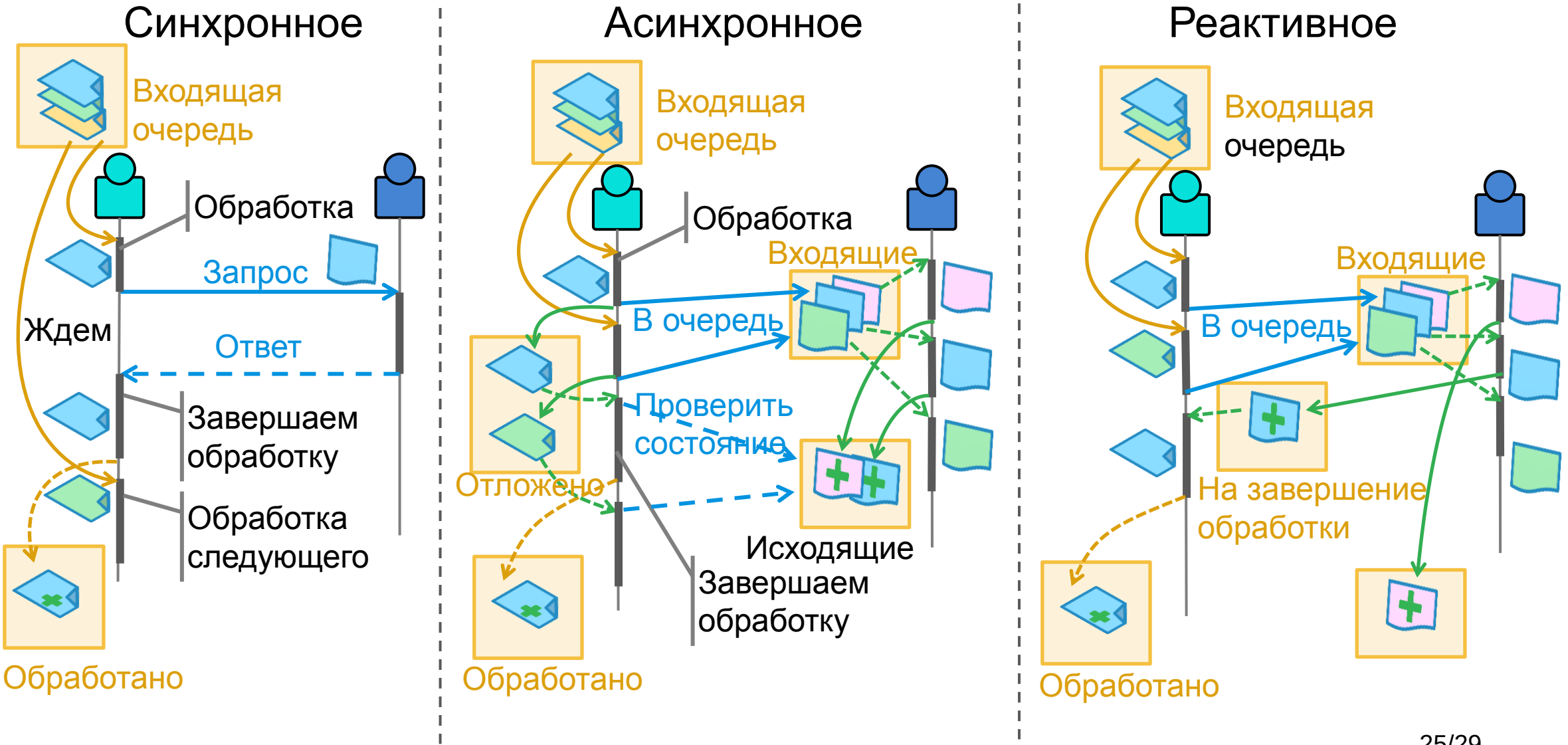
Ведение остатка на складе – варианты

- 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

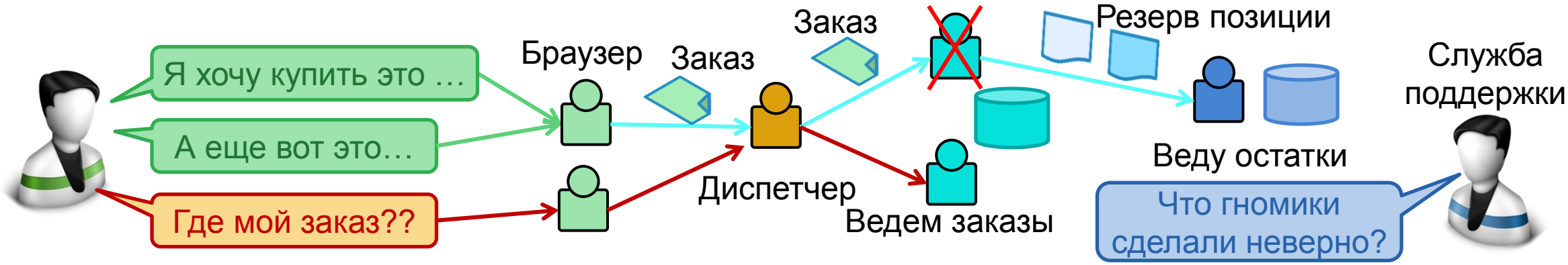
Вопросы:

- Что делать, если зарезервировали не весь заказ?
- Что делать, если резервирование идет долго?
- Вернее так: переводить ли заказ на оплату, если резерва долго нет?

Варианты межсервисного взаимодействия

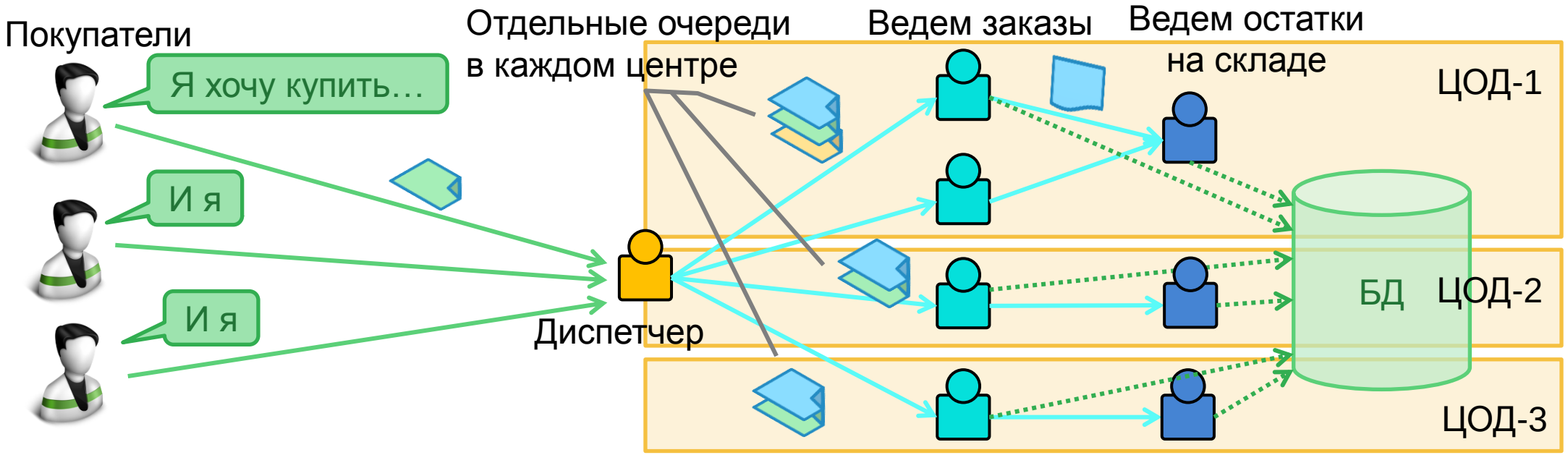


Устойчивость: гномики умирают, их отстреливают



- Ситуация: идет 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…
- Покупатель не обязательно авторизован – до этого могло не дойти …
- Покупатель может ехать в Сапсане или в месте с плохой связью …
- Ситуация может быть и при оплате – там чужой платежный шлюз…

Надежность: ноды в разных датацентрах



- 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

Варианты коммуникационного пространства

- Каждый с каждым: по любому поводу гномик знает к кому обратиться напрямую.
- Messaging: гномики посылают письма через почту и на них реагируют
- Eventing: гномики кричат в пространство и кто-то откликается на услышанное

Это – логическая коммуникация. Может быть на разных физических

- Шина данных – часто eventing, а не messaging
- RabbitMQ и ApachMQ
- Kafka – вечные очереди в памяти

И в заключение…

ANALYST DAYS #11
analystdays.com

- Мир изменился и старые способы постановки не работают в новой архитектуре
- Осваивайте внутреннее устройство современных приложений
- Метафоры надо проверять и подбирать, гномики не всегда помогут

Вопросы? Обращайтесь!

Максим Цепков
http://mtsepkov.org
maks.tsepkov@ya.ru

На сайте много материалов по анализу и архитектуре, Agile, и ведению проектов, управлению знаниями, мои доклады, статьи и конспекты книг.