СТАЧКА

Бизнес и софт как единая система — описываем архитектуру предприятия

Максим Цепков

IT-архитектор и бизнес-аналитик,
Навигатор и эксперт по миру Agile,
методов самоуправления и моделей soft skill

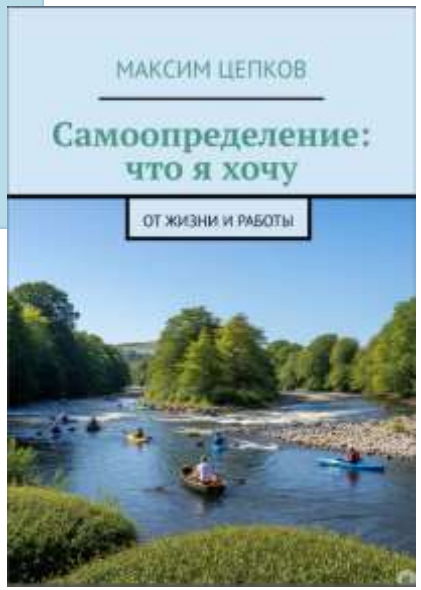
mtsepkov mtsepkov.org

Стачка | Ульяновск
10-11 апреля 2026

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**).

Китай как лидер новой волны технологических революций

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

О чём пойдет речь?

- ИТ-проект начинается с потребностей бизнеса,
  и его результат – софт – встраивается в бизнес и его изменяет
- Надо понимать надсистему и место в ней – так говорит системный подход
- Если же вместо этого описывать границу софта в виде требований,
  то он оказывается не слишком пригоден, требуется долгая доводка
- Поговорим о том, как описывать устройство бизнеса и встройку в него софта
  как единую систему, тем более, что **граница является подвижной**

Мои выступления по этой и смежным темам

- Архитектура софта и бизнеса в сложном ИТ-ландшафте (AnalystDays-2025)
- Постановка от модели бизнеса до детального дизайна требований: как делать и кому (13.03.2025)
- Системное мышление и его место в работе аналитика (AnalystDays-2024a)
- DDD: модели вместо требований 9 лет спустя (ЛАФ-2023)
- Визуальное проектирование масштабируемых приложений (TechLead-2021)
- Обеспечиваем устойчивость интеграции (SQAdays-2025)

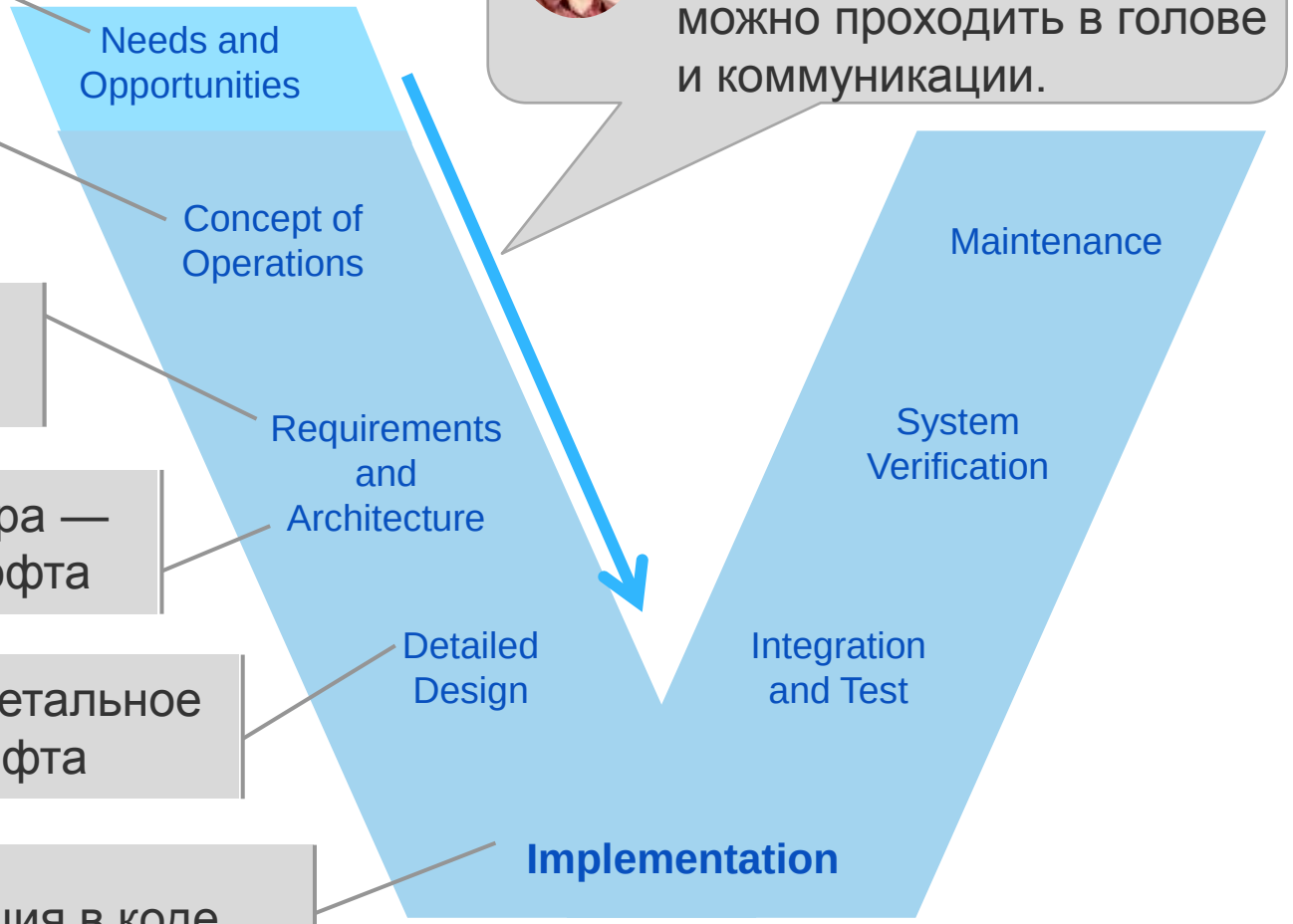
Разрыв между ИТ и бизнесом

Путь ИТ-проекта — V-model

1. Начинается с идеи сделать хорошо некоторым людям с помощью софта
2. Далее надо придумать, как они будут действовать, используя софт
3. И сформулировать требования к софту, исходя из способов работы
4. Затем проектируется архитектура — принципиальное устройство софта
5. Затем — дизайн, детальное устройство софта
6. И реализация в коде

Различный уровень заказа

Этот путь не обязательно требует документов, его можно проходить в голове и коммуникации.

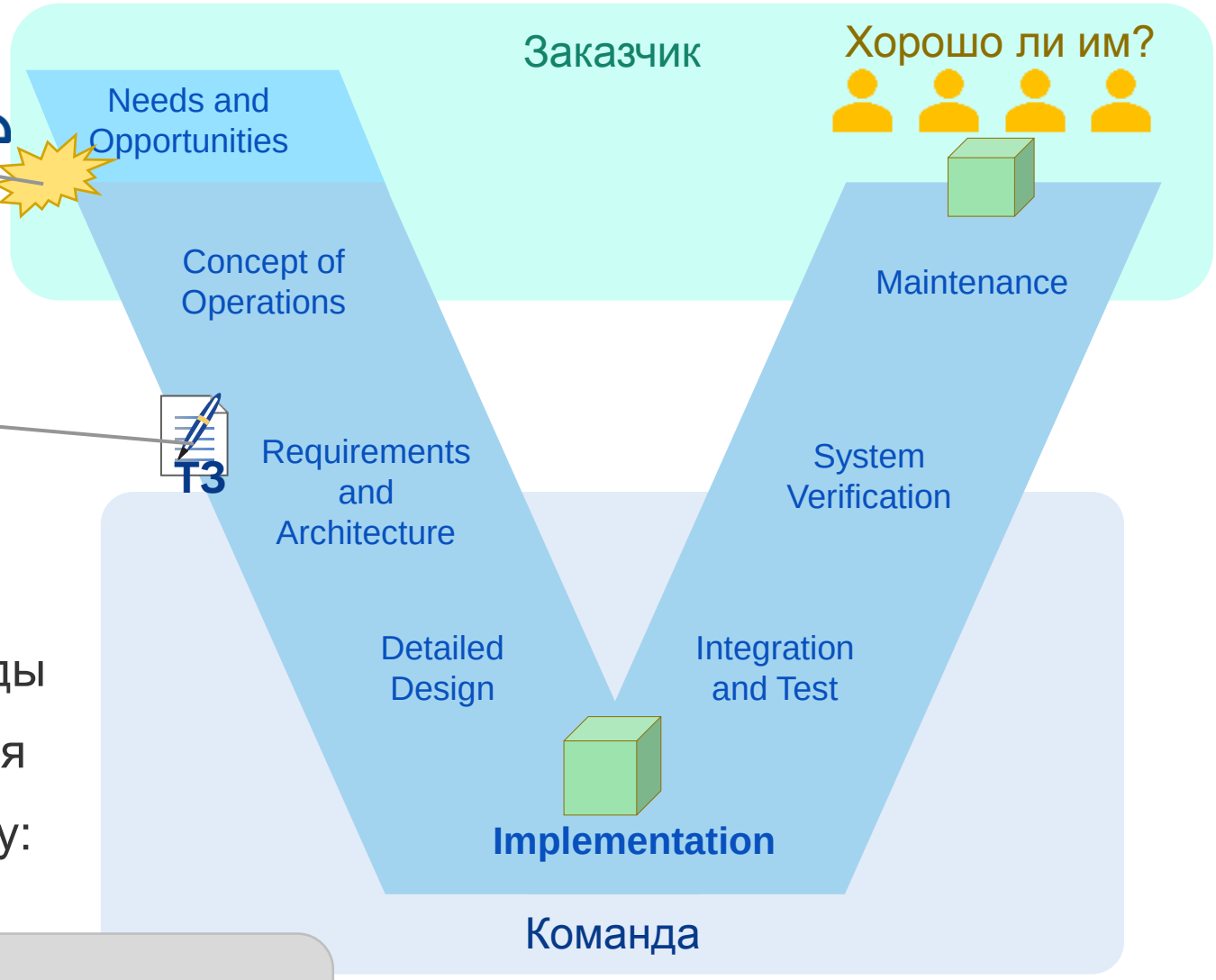


[V-Model (Wikipedia)]

ТЗ — способ сменить ответственность

Идея: сделать хорошо людям

Задача: напишите софт, удовлетворяющий требованиям

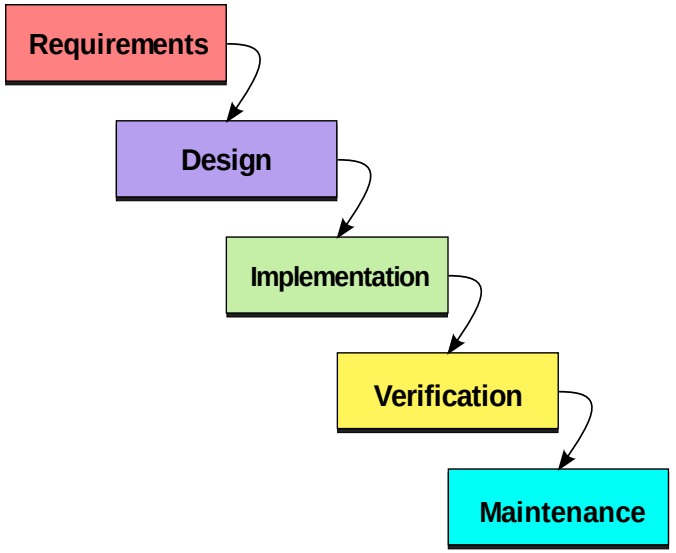


- Ответственность за достижение целей проекта вынесена за пределы ИТ-команды
- Как следствие, цели часто не достигаются
- Выделение этапа ТЗ не решает проблему: ТЗ утверждают, а гарантий оно не даёт

Нужны ли вам требования к системе?
Зависит от границы проекта и контракта с заказчиком

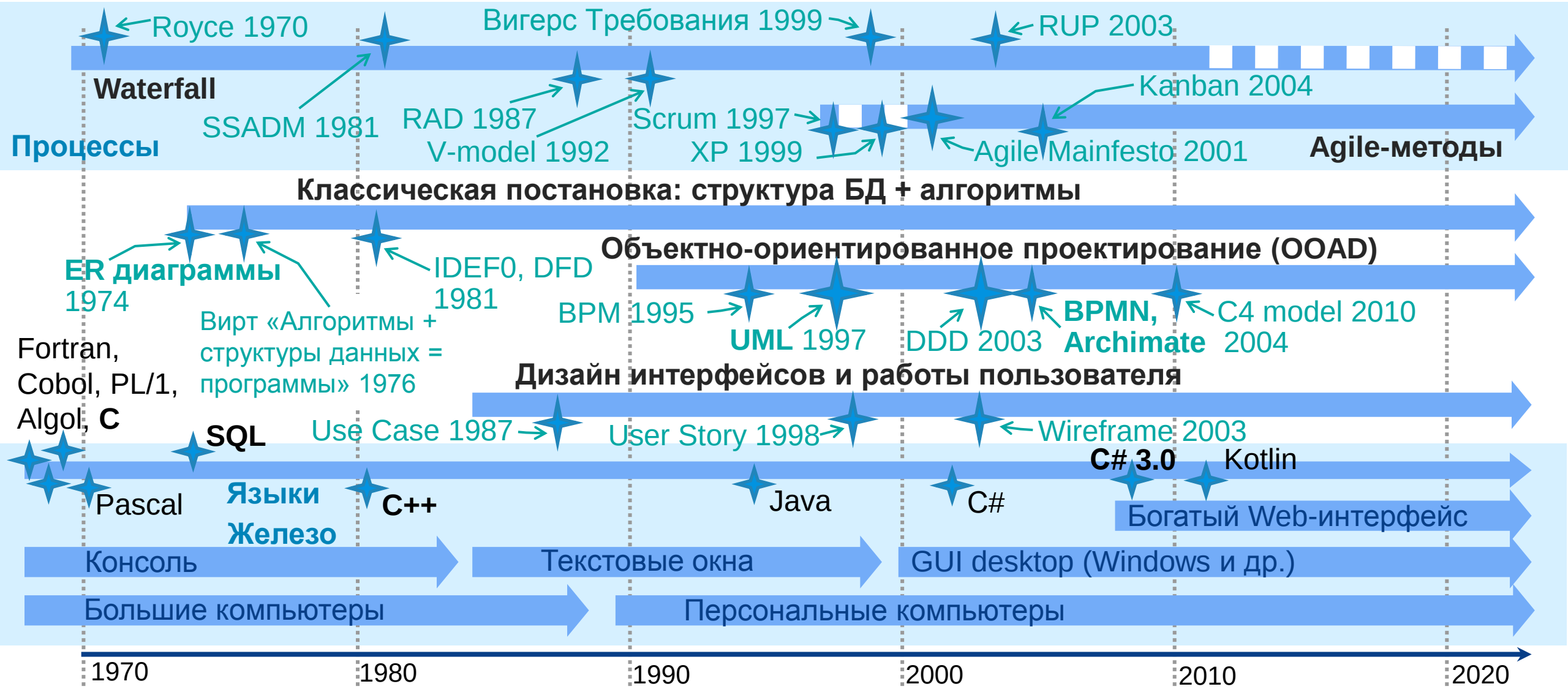
Проблема сложилась исторически

- In-house разработка плотно работала с бизнесом и понимала его, эта модель применяется до сих пор в части компаний
- По мере развития ИТ, для него попробовали применить методы промышленного производства: разделение на операции и организацию процессов:
  - Чтобы обойтись узкими специалистами
  - Чтобы гарантировать успех за счёт регламентов
- В этом случае **ИТ можно не разбираться в бизнесе**
- Успеха достичь не удалось, но учебники остались, и схема воспроизводится

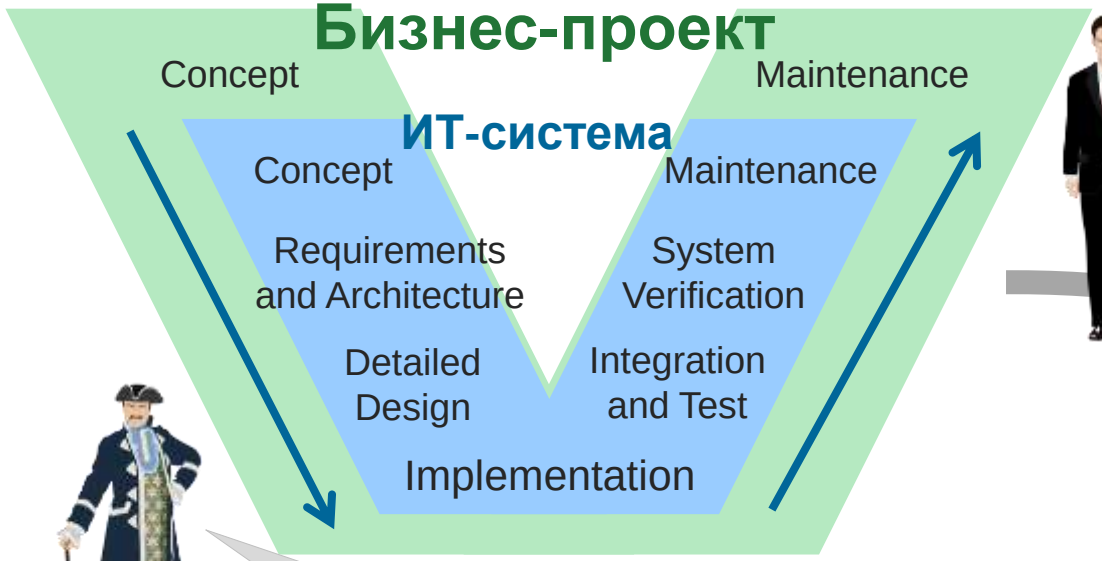


Модель водопада — http://en.wikipedia.org/wiki/Waterfall_model

Проектирование и его окружение



История культур ИТ-проектов



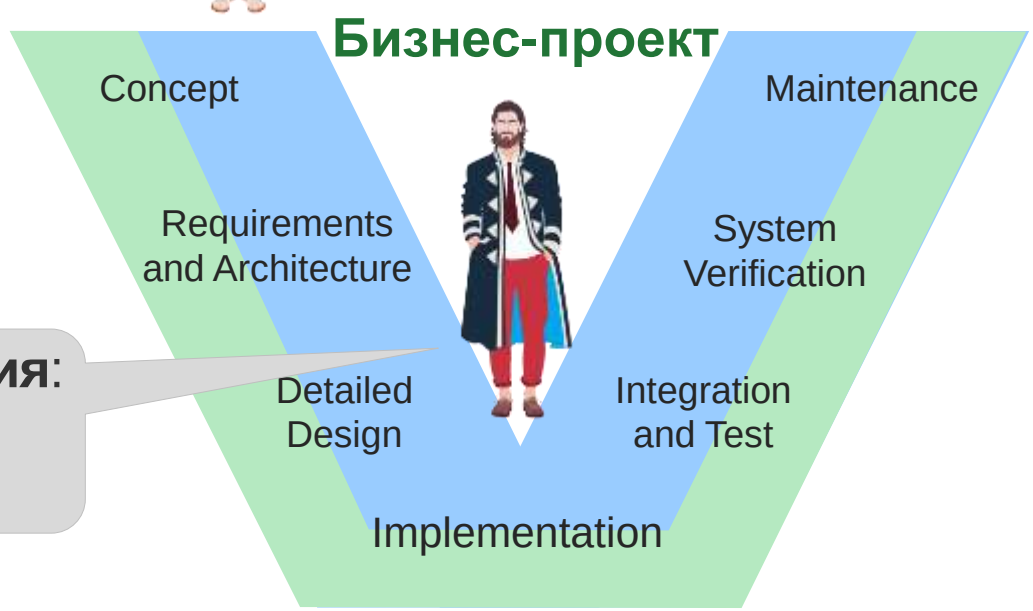
Эпоха **НИОКР** – учимся разработке

Работа по заказу

Эпоха **RUP** – бюджет и сроки, бизнес должен дать требования

Сервис

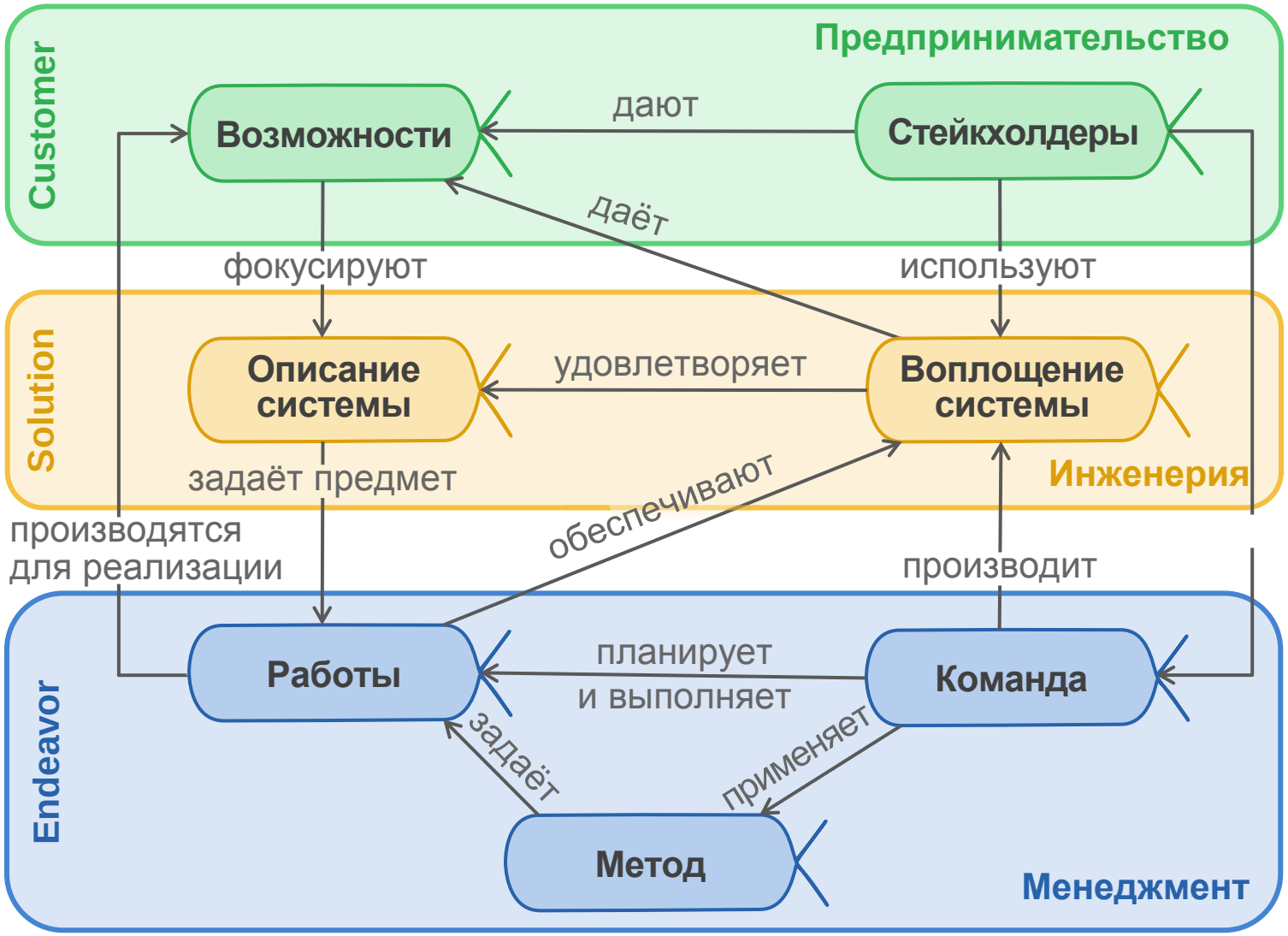
Время **Agile** – меняем ИТ в темпе изменений бизнеса!



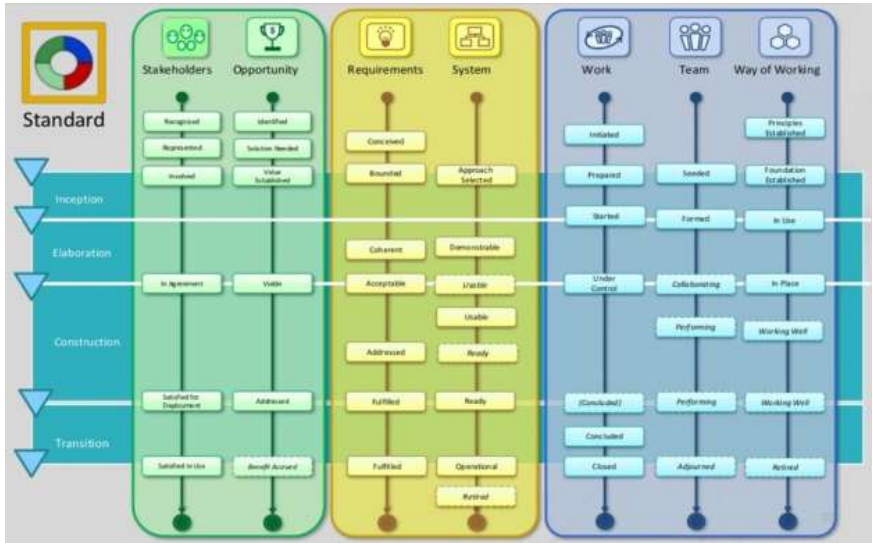
Партнёрство

Public web и продуктовый подход, **цифровизация**: граница между софтом и бизнесом подвижна, **единый проект, а ИТ-часть — главная!**

OMG Essence: бизнес и софт вместе



- **Фокусы внимания** вместо линейной схемы проекта
- Заказчики (предпринимательство) – **внутри деятельности**, а не задают контекст



Продукт как возможность для бизнеса

Пример. Бизнес требует поддержать услугу срочной доставки, так как без неё есть риск проиграть конкурентам, у которых она уже появляется

- Вопросы о возможности: кто будет пользоваться, для каких заказов, что значит «срочная», как мы будем проверять гипотезы и какова мощность в динамике?
  - Справится пеший курьер или нужна машина?
- Будет ли отдельная система срочной доставки? Или надо существующую систему доставки дополнить элементами и наделить способностью так доставлять?
- Какова целевая операционная стоимость доставки, какова стоимость в пилоте, каковы допустимые инвестиции, в том числе для создания ИТ-систем?
  - Если курьер будет брать самокат, стоимость останется допустимой?
- Какие возможны лёгкие решения на пилоте, справится ли Excel + мессенджер?
- Как будем реагировать, когда не справляемся, кого из клиентов обижать?
- И так далее…

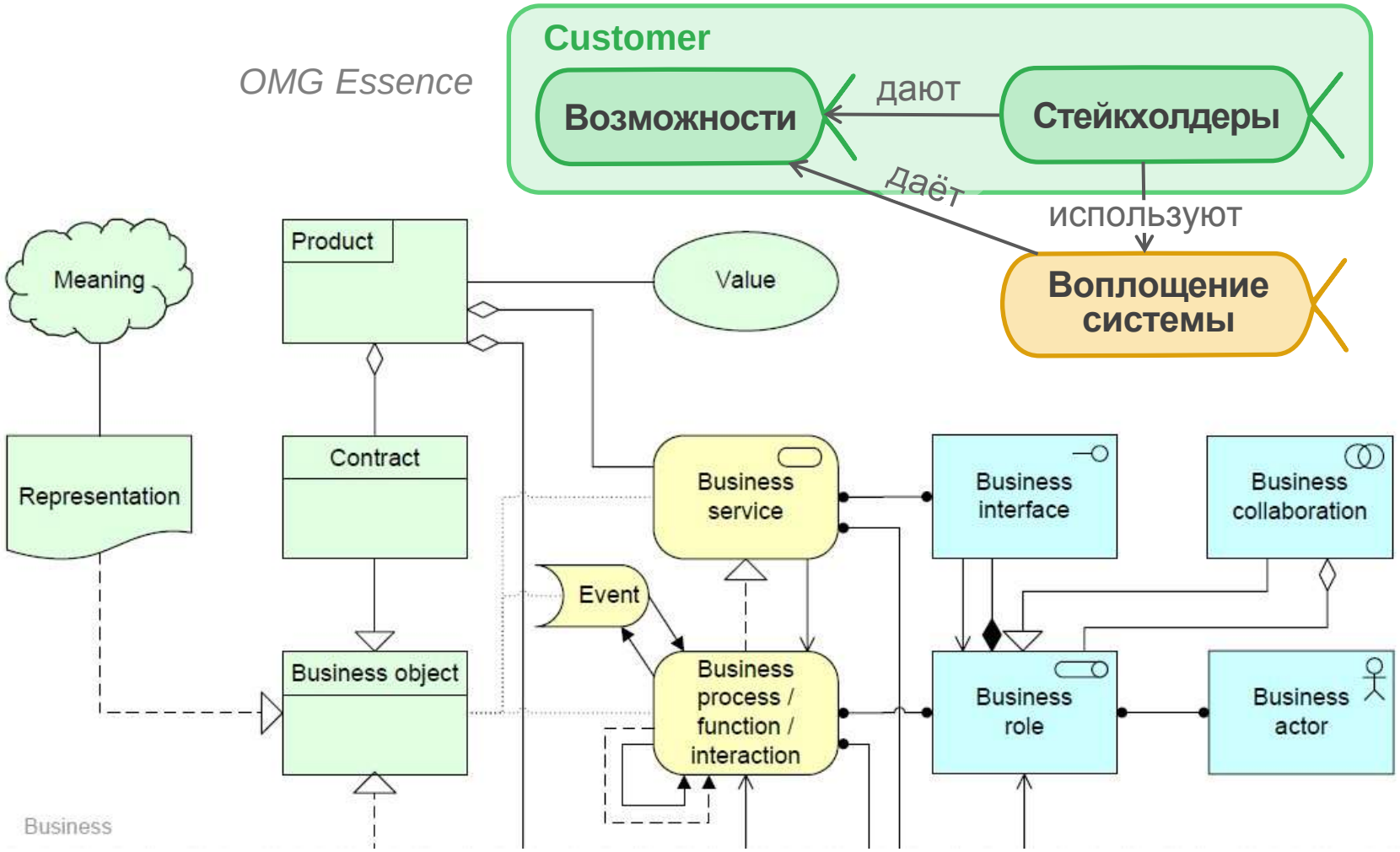
Возможности: impact map или карта гипотез

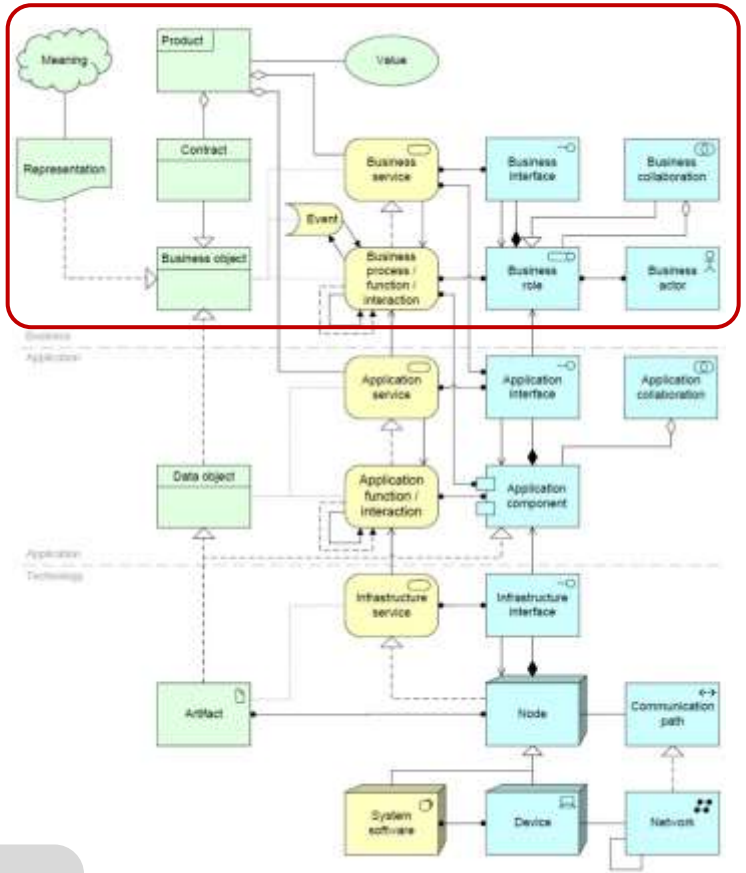
- Методика предложена **Александром Бындю в книге «Карта гипотез»**
- Четыре основных элемента:
  - **Цель**, которую мы хотим достигнуть (включая показатели или критерии продвижения)
  - **Субъекты**, поведение которых должно измениться для достижения цели
  - **Гипотезы** о способах изменения поведения субъектов
  - **Задачи**, необходимые чтобы проверить гипотезу
- Формат гипотезы: **если** мы сделаем А (и Б), **то** субъект <так-то> изменит поведение, **потому что** действие А так-то на него повлияет (объяснение)
- Стартовать создание можно от того, что есть, нужно простроить логическую связь от задач к целям через гипотезы, в этом часто проблемы

Метод – развитие impact mapping. Основное отличие – **формат гипотезы**, обосновывающий влияние выполнения задачи на субъекта.

Бизнес-уровень в ArchiMate

OMG Essence

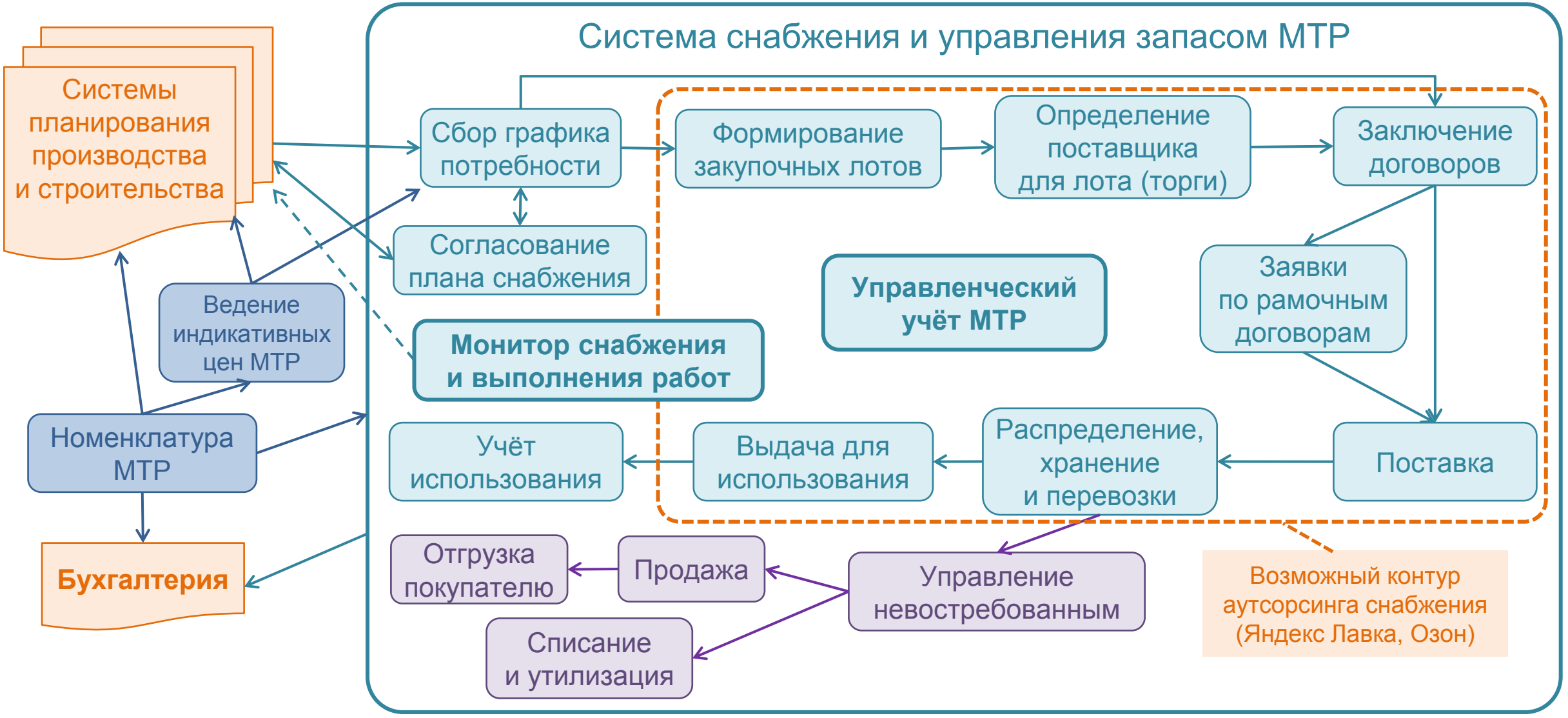




Метамодель ArchiMate 1.0

В основе ArchiMate — продуктовая модель: продукты несут ценность для потребителя, их обеспечивают бизнес-сервисы, включающие процессы и функции, выполняемые агентами.

Архитектура снабжения: процесс из функций



Описание бизнеса – схема бизнес-процессов?

- Схемы процессов описывают бизнес-архитектуру лишь частично:
  - На них нет целей бизнеса и цепочек ценности
  - Они не показывают сопряжение бизнеса и софта
  - Далеко не всю деятельность можно описать языком бизнес-процессов
- Они быстро устаревают, жизнь меняется быстрее:
  - Схемы слабо соответствуют реальным процессам в компаниях
  - Описания конфликтуют с инструкциями пользователей по работе с системами;
  - Об этом я говорил ещё в 2011 году: «Описание бизнес-процессов — waste?».
- Есть много разных способов описания бизнес-архитектуры, надо представлять спектр вариантов и выбирать подходящие:
  - Не стоит использовать описания бизнес-процессов лишь потому, что «так принято»
  - для описания архитектуры ИТ-компании часто используют task flow, это альтернатива
  - но если основа автоматизации — BPMN-движок, то бизнес-процессы надо использовать

Формальная бизнес-модель и реальность

Оптовые продажи магазинам и торговым сетям

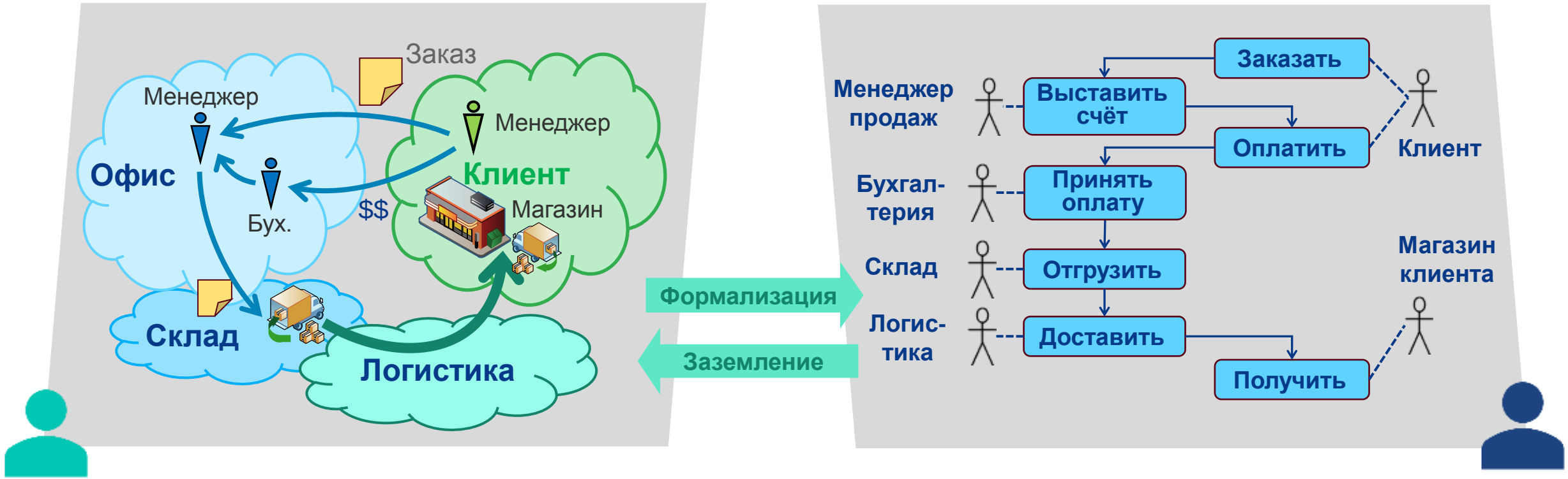
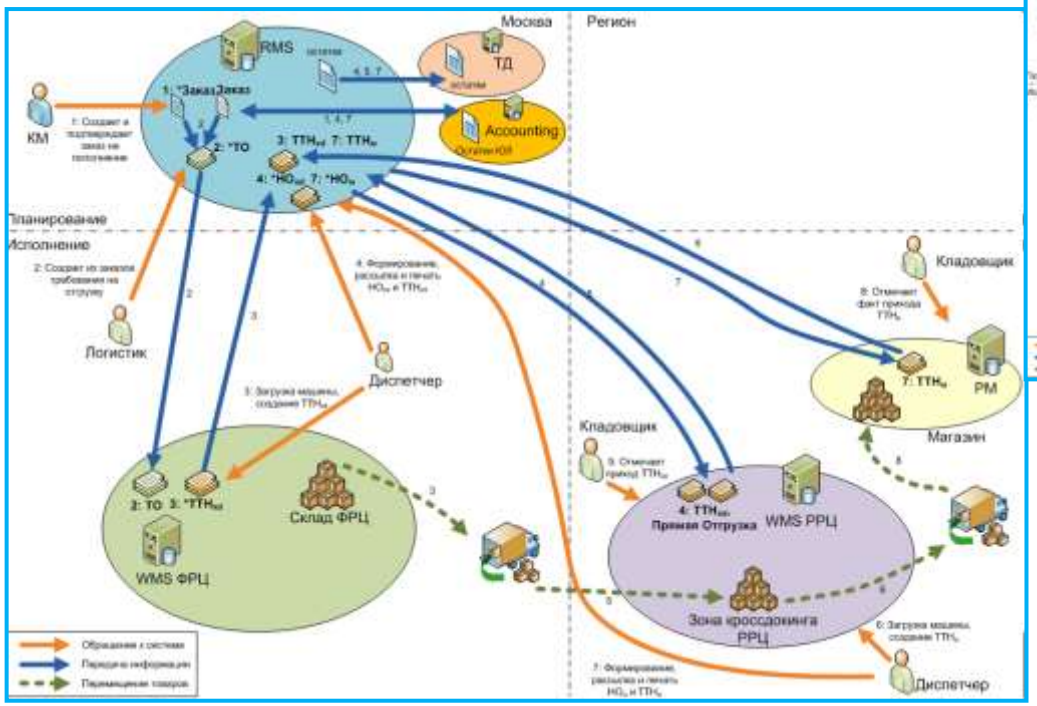


Схема бизнес-процессов — это модель происходящего в реальности.
Видим повседневную деятельность за формальной бизнес-моделью!

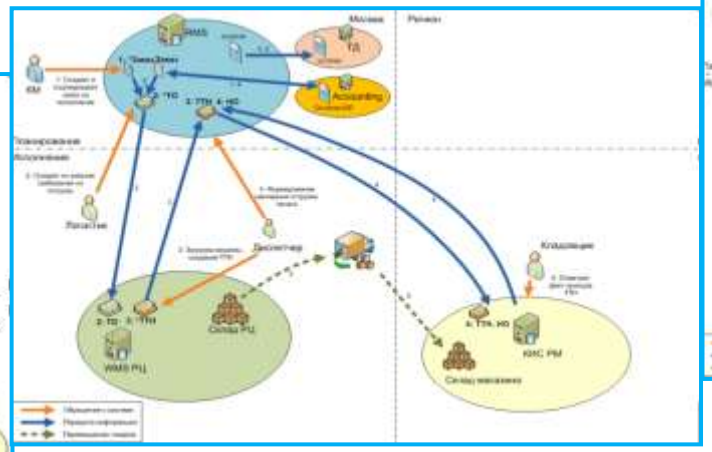
Проверка в неформальной модели

Мы можем построить формальную модель процесса и его реализации, но не всегда заказчик может её проверить. Часто он доверяет, а на внедрении вскрываются проблемы. Решение — вернуться в неформальную модель или показывать прототипы.

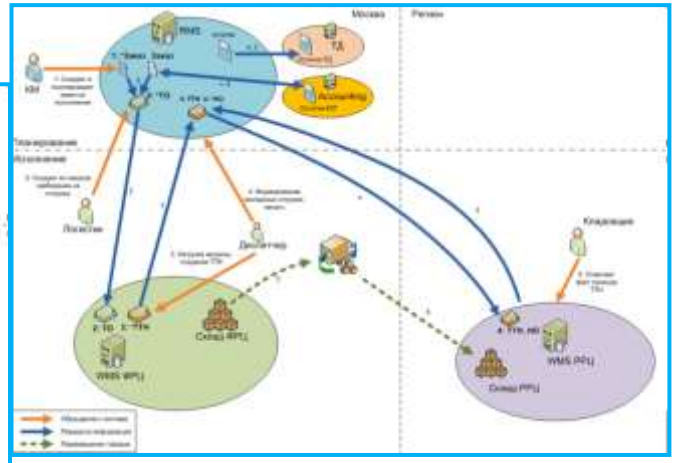
1



2



3



Снабжение магазинов: на неформальных моделях показывали кейсы работы будущей системы.

Где не работает описание процессами?

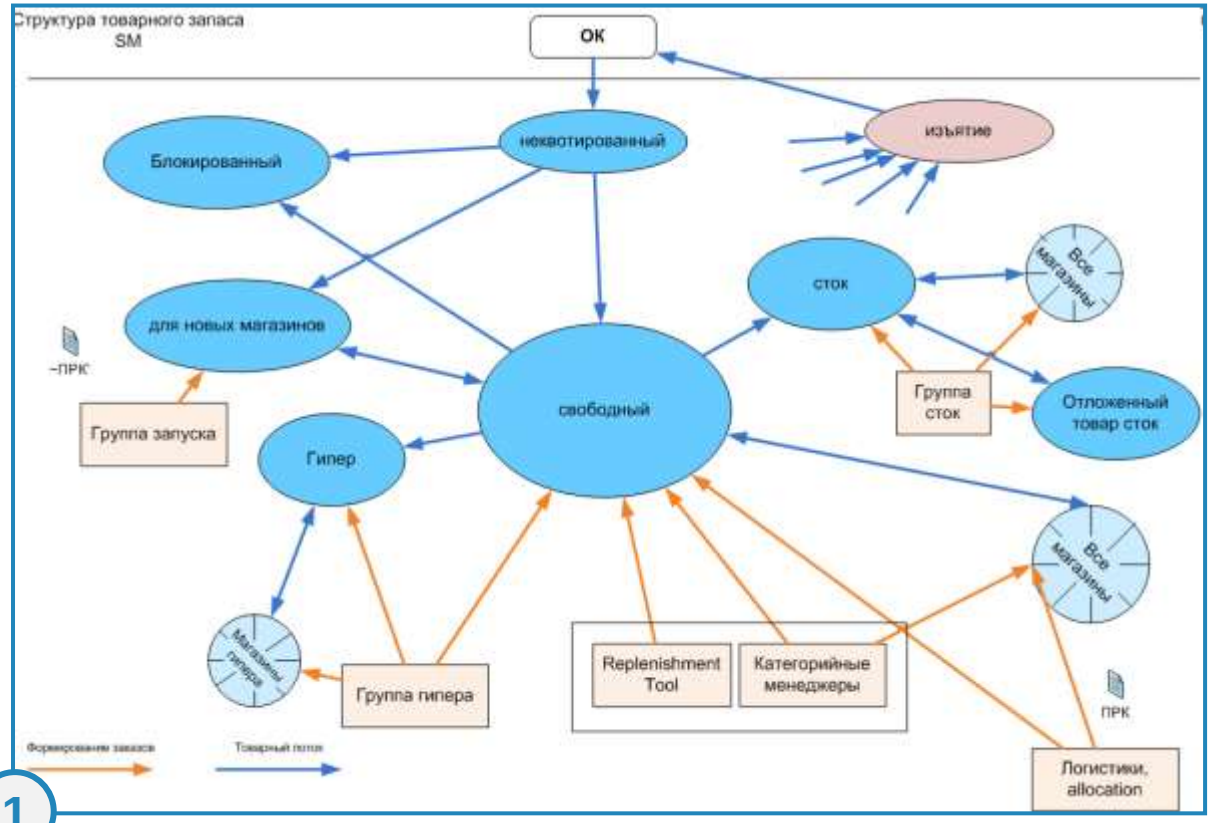
- **Справочники**
  - Структура справочных данных: товары, цены, акции, покупатели, точки выдачи.
  - API и алгоритмы вычисления: цены со скидками, доступные даты доставки.
  - Способы настройки параметров алгоритмов на основе справочных данных.
- **Планирование** деятельности с учётом ограничений требует 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.

Не применяйте описание процессов там, оно не работает,
даже если эту деятельность процессами называют. **Ищите альтернативы!**
Для любого метода: надо уметь видеть границы применения.

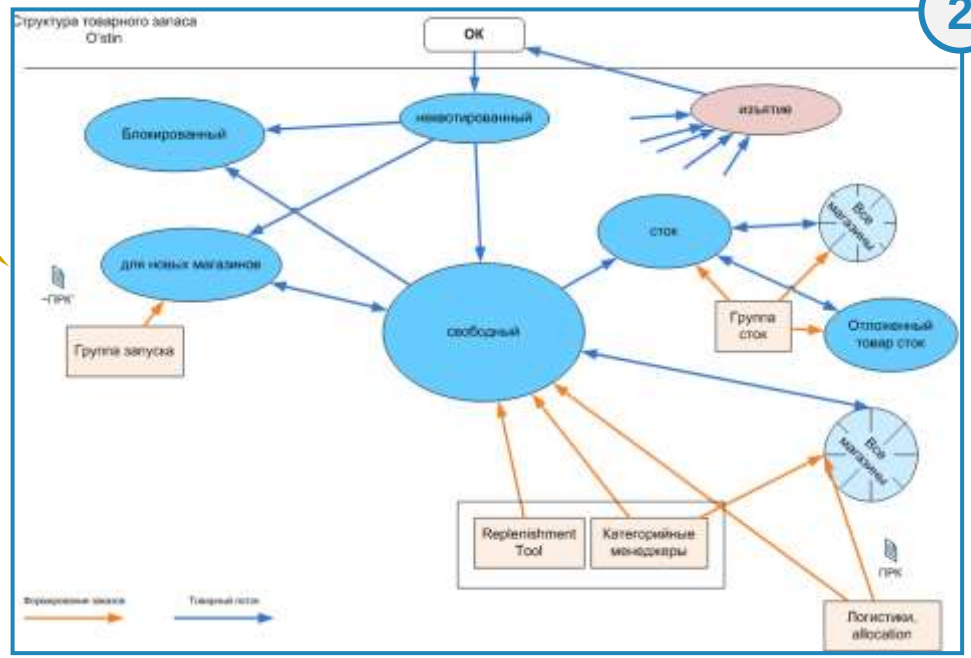
Визуальный образ для назначения системы

Пример:
деление товара

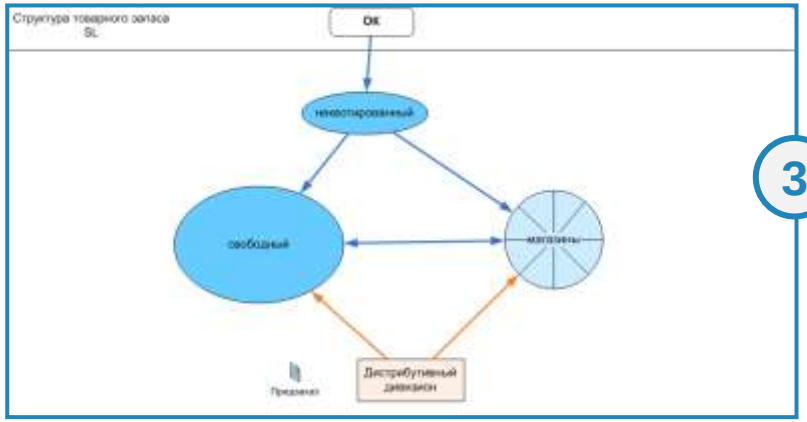
Варианты процессов
в простой нотации,
схемы понимаются на лету



1



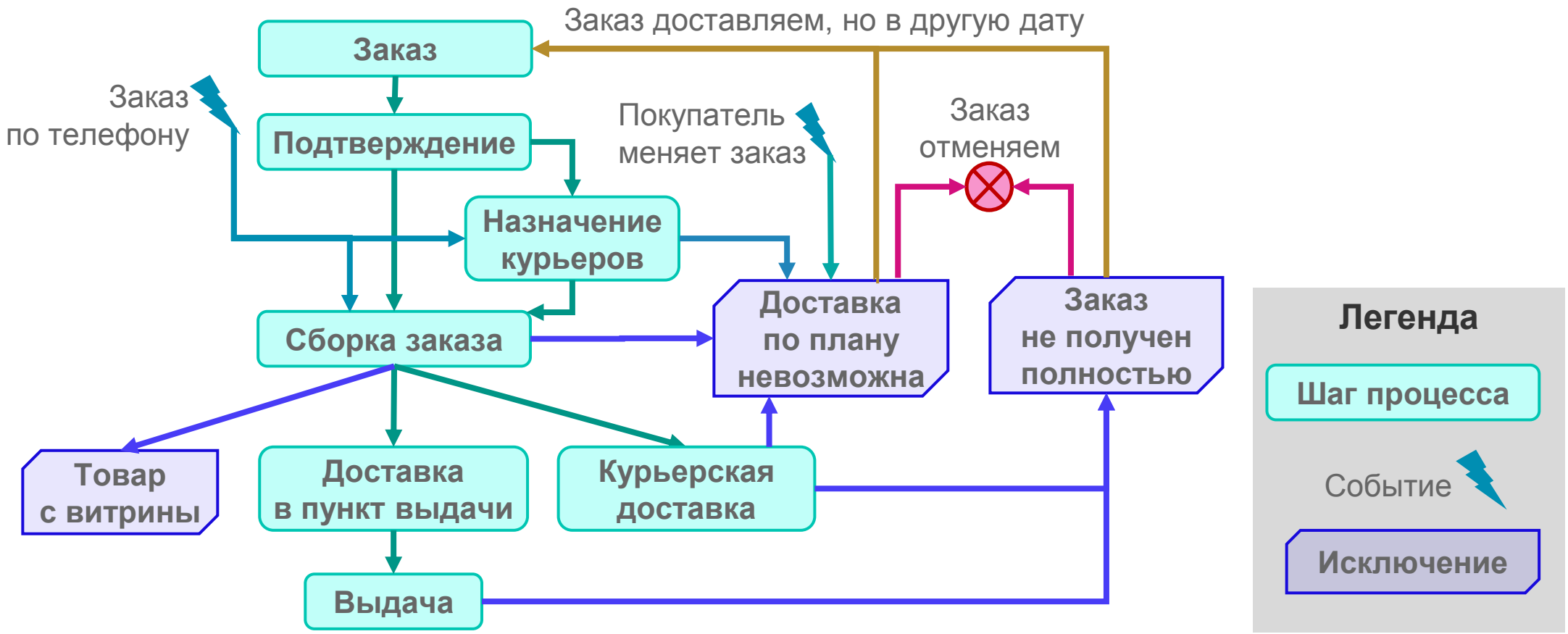
2



3

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».



Итак, что нужно знать о бизнесе?

- Для успешного создания софта надо понимать архитектуру предприятия как системы, в которой **люди и софт работают совместно**
- Есть два взгляда на главное в организации, это — различие мышления:
  - Главное — правильно организовать процессы, и всё будет работать
  - Нужно понимать цели бизнеса и работать на них, а процессы применять где уместно
- В современном мире работает второй способ, а первый уместен ограниченно, но его часто применяют, руководствуясь старыми учебниками
- Есть много альтернативных способов, чтобы описать работу компании
- Если в компании описания процессов создаются «чтоб было», то так же следует относиться к тем описаниям, которые требуют от вас

От бизнеса к софту

Проект чёрного ящика: user story + экраны

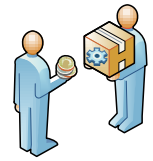

Сделать заказ


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*

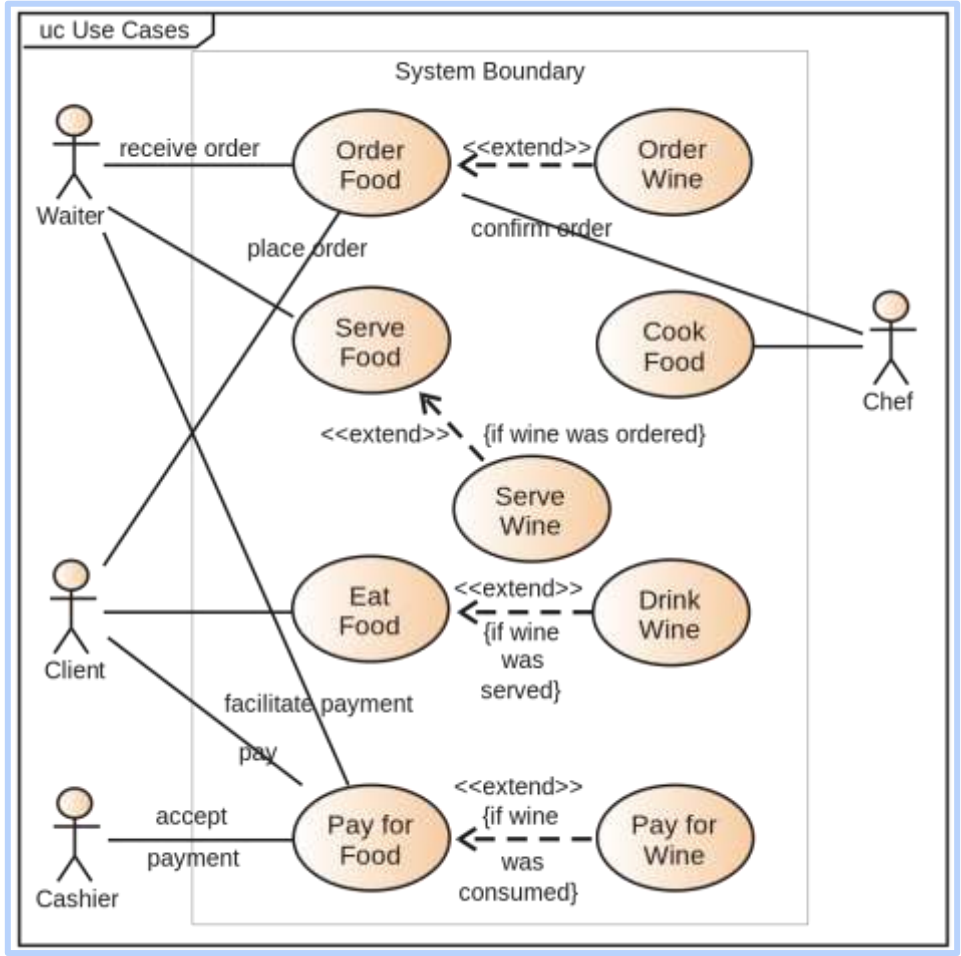
Контекстная диаграмма C4 Model — система в окружении

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

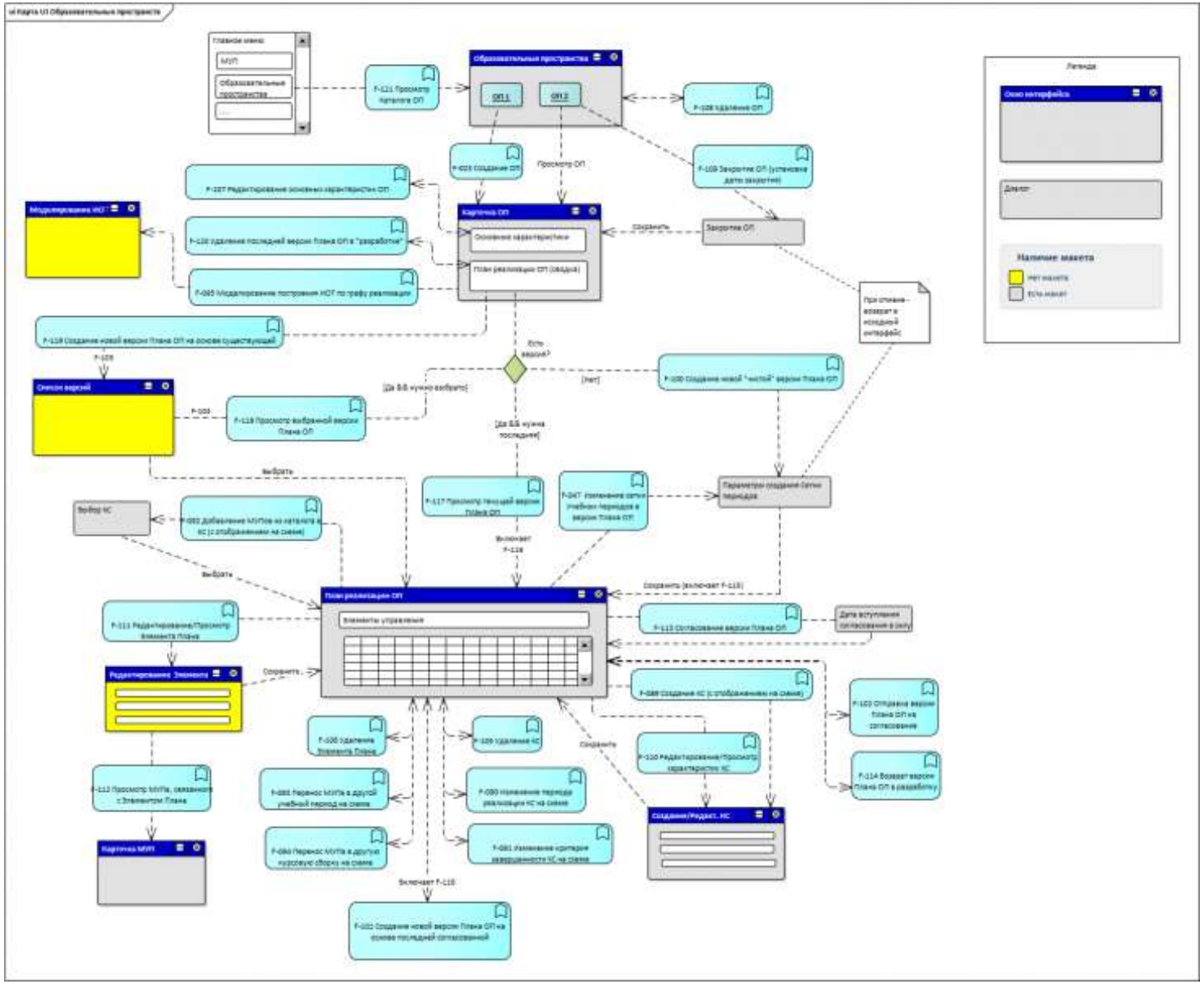
Стрелки — акты взаимодействия

Множество внешних систем — смена парадигмы на сервисы

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



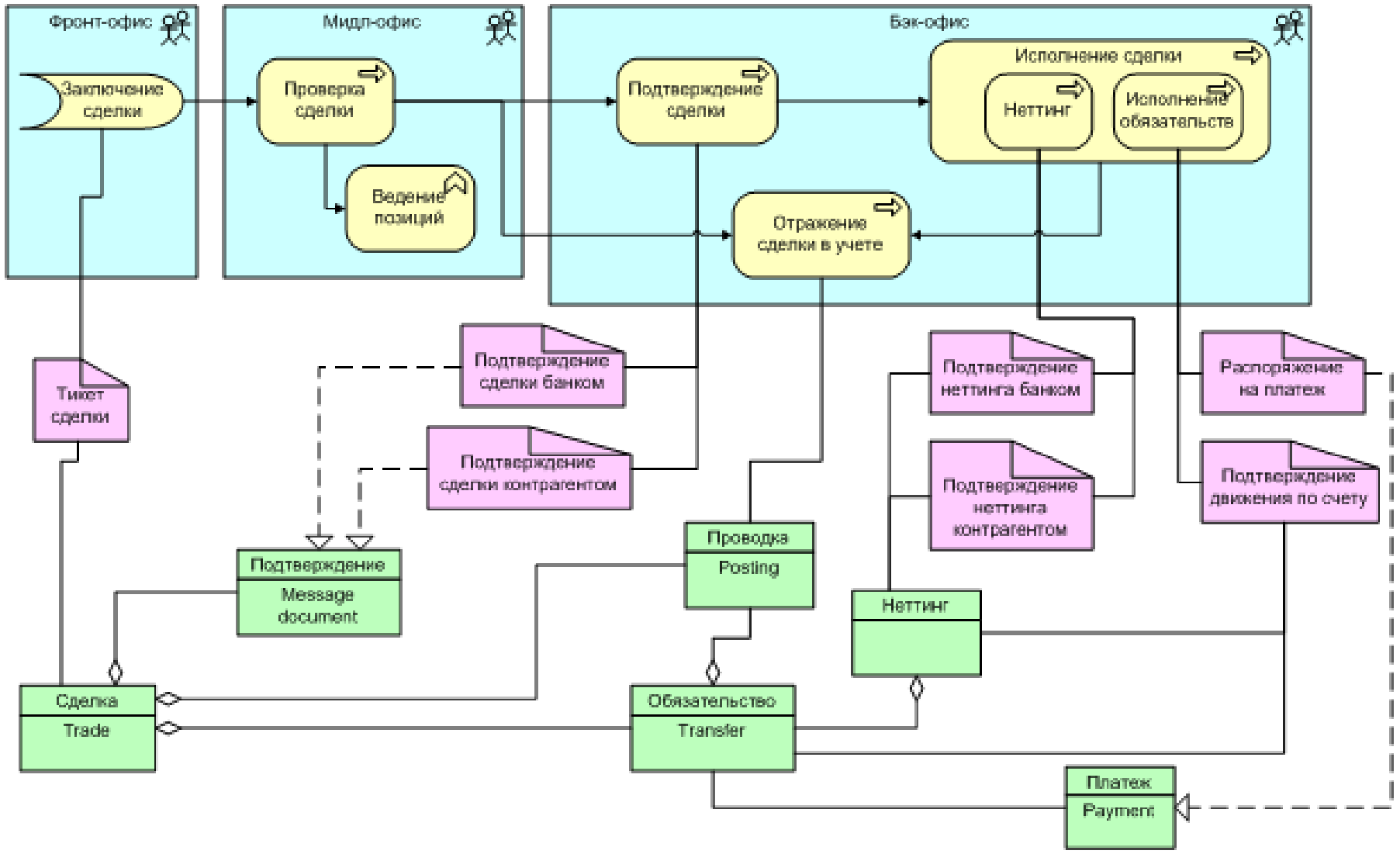
Поддержка бизнес-функций экранами приложения



Use case или user story?

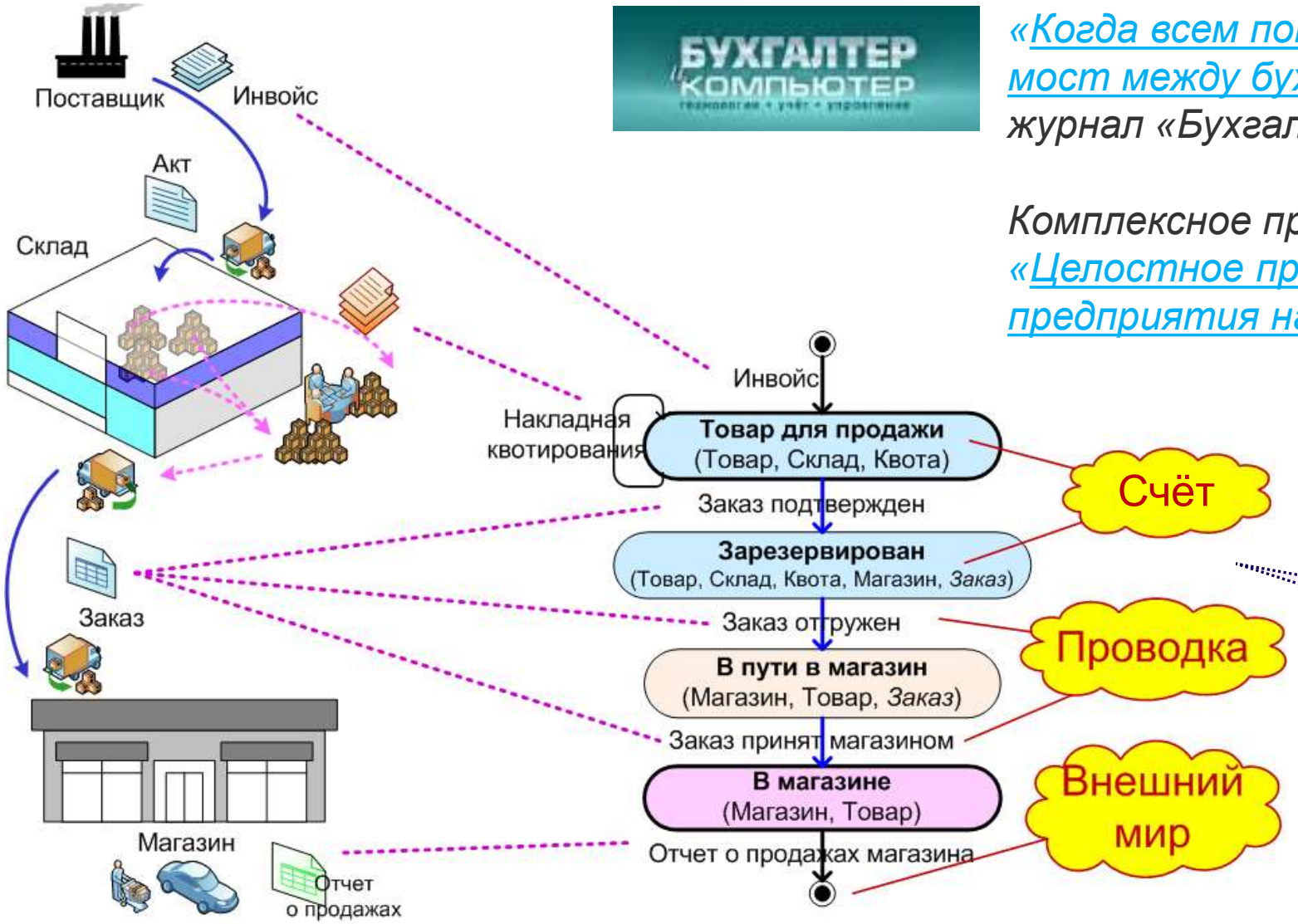
- Оба служат для описания работы пользователя для реализации интерфейсов, на их основе легко делать тест-кейсы
- User story:
  - ☺ фиксируют цели пользователей, а не только взаимодействие с системой
  - ☺ хорошая единица работы: можно менять порядок для реализации по важности
  - ☹ однако, сложные истории могут потребовать существенных переделок
- Use case:
  - ☺ комплексно описывает взаимодействие, это можно заложить в реализацию
  - ☹ плохая единица работы: в одном use case смешаны частые и редкие сценарии
  - ☺ можно делить use case на slice — смотри Use case 2.0 Ивара Якобсона
- Проект целиком – **story mapping**, развитие **Customer journey map** (**CJM**) и **Jobs To Be Done** (**JTBD**) – использование софта группами пользователей

Представление через объекты данных



Нотация ArchiMate 1.0

Учёт: бизнес как поток ресурсов



«Когда всем понятно. Диаграммы учёта: мост между бухгалтером и разработчиком» — *журнал «Бухгалтер и компьютер», №5-2011.*

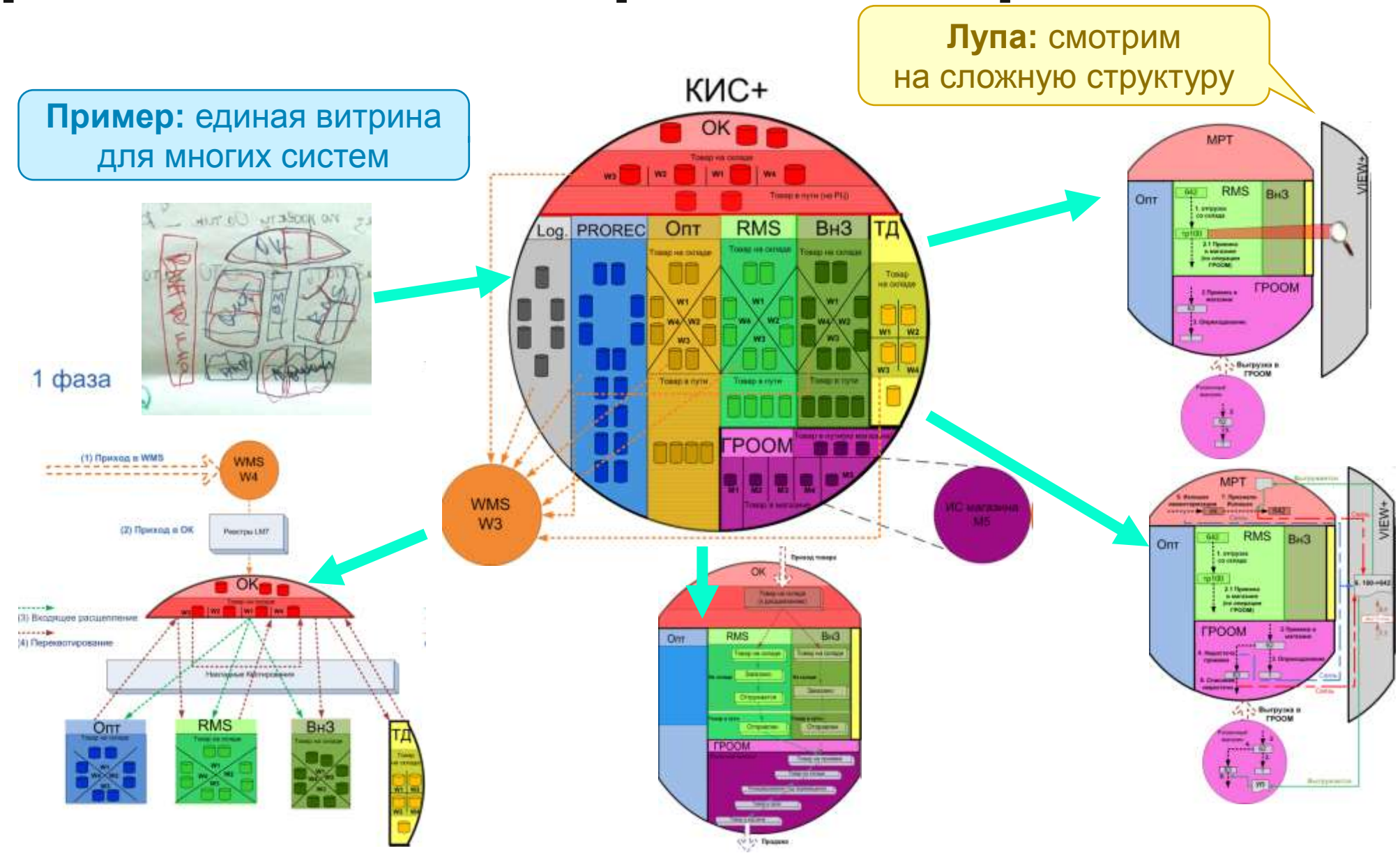
Комплексное представление — мое выступление «Целостное представление деятельности предприятия на диаграммах учёта».

Движение товаров
и других ресурсов отражается
как проводки между счетами

Оригинальный образ для отражения систем

Пример: единая витрина для многих систем

Лупа: смотрим на сложную структуру

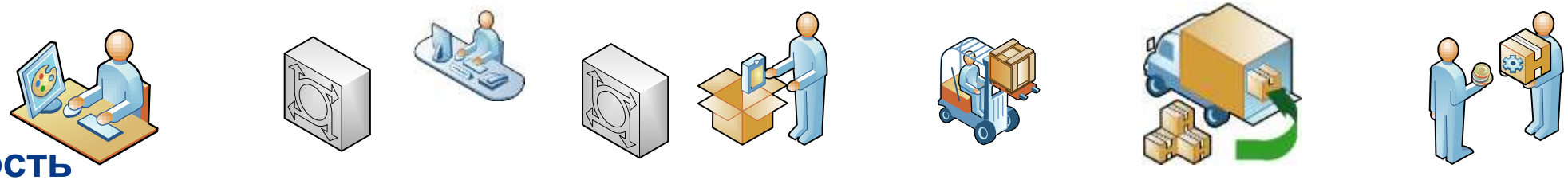


Передача информации о бизнесе и пользователях

- Есть ли лица, которые готовы заказать или нужно исследования?
- Каков способ коммуникации для получения знаний:
  - интервью и работа с регламентами и формирование документа-описания
  - интерактивные встречи — story mapping или event storming
- Способ передачи общей информации о бизнесе разработчикам:
  - участие в общей интерактивной встрече с заказчиком
  - краткий документ и лекция
  - метафора системы, если получилось придумать — эффективная практика XP
  - модель процессов или событий
- Способ передачи детальной информации по конкретной фиче:
  - описания бизнес-процессов
  - описания сценариев работы пользователя
  - user story с целями пользователей
  - контекстная диаграмма в C4-Model

Представляем бизнес-процесс в ИТ-системе

Уровни представления (интернет-заказ)



Деятельность

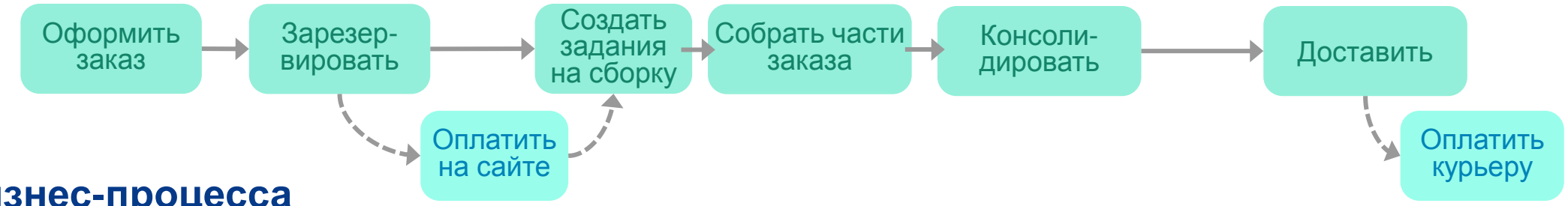


Схема бизнес-процесса

Объекты и их состояния

Для покупателя: как создать и оплатить заказ

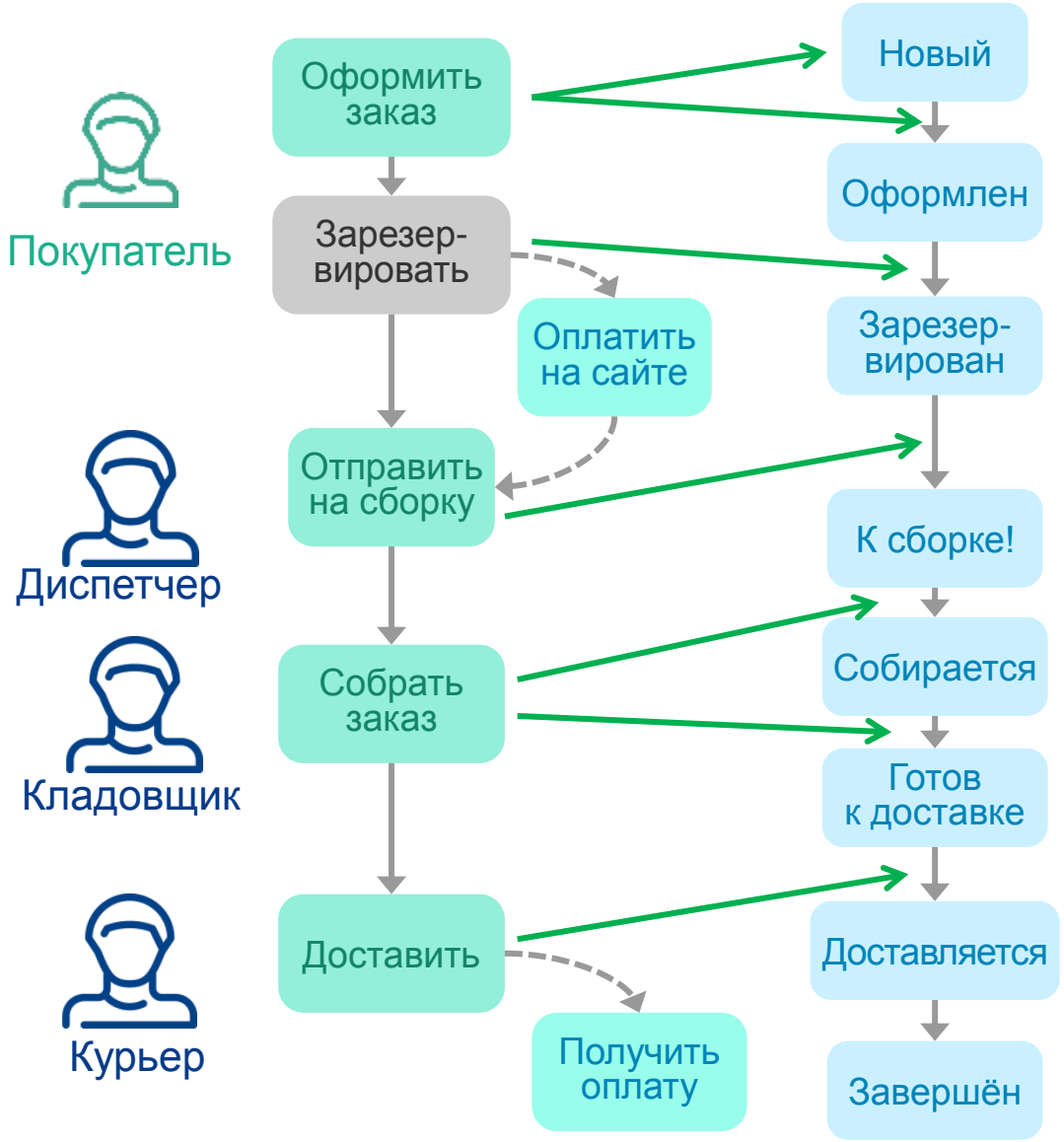
Для оператора: обработка заказа в офисе

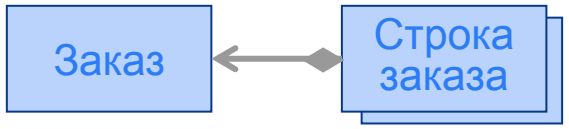
Для кладовщиков: сборка заказов и выдача курьерам

Для курьеров: передача покупателю

Руководства по системе

Монолит — процесс через workflow

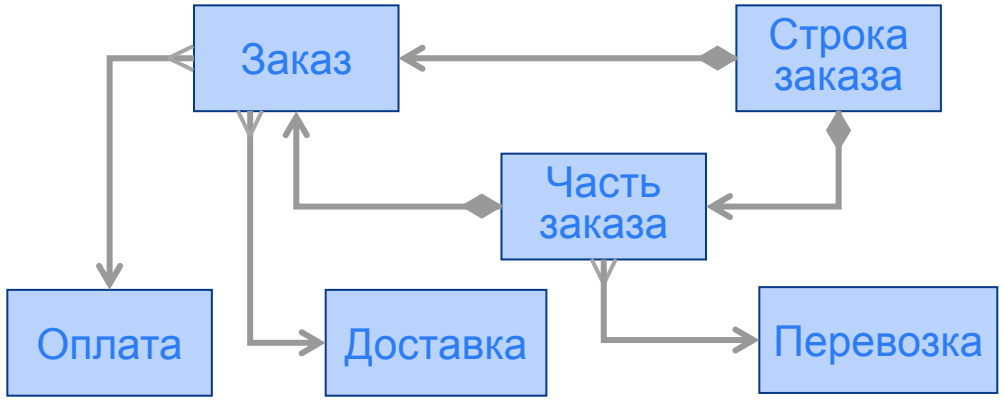




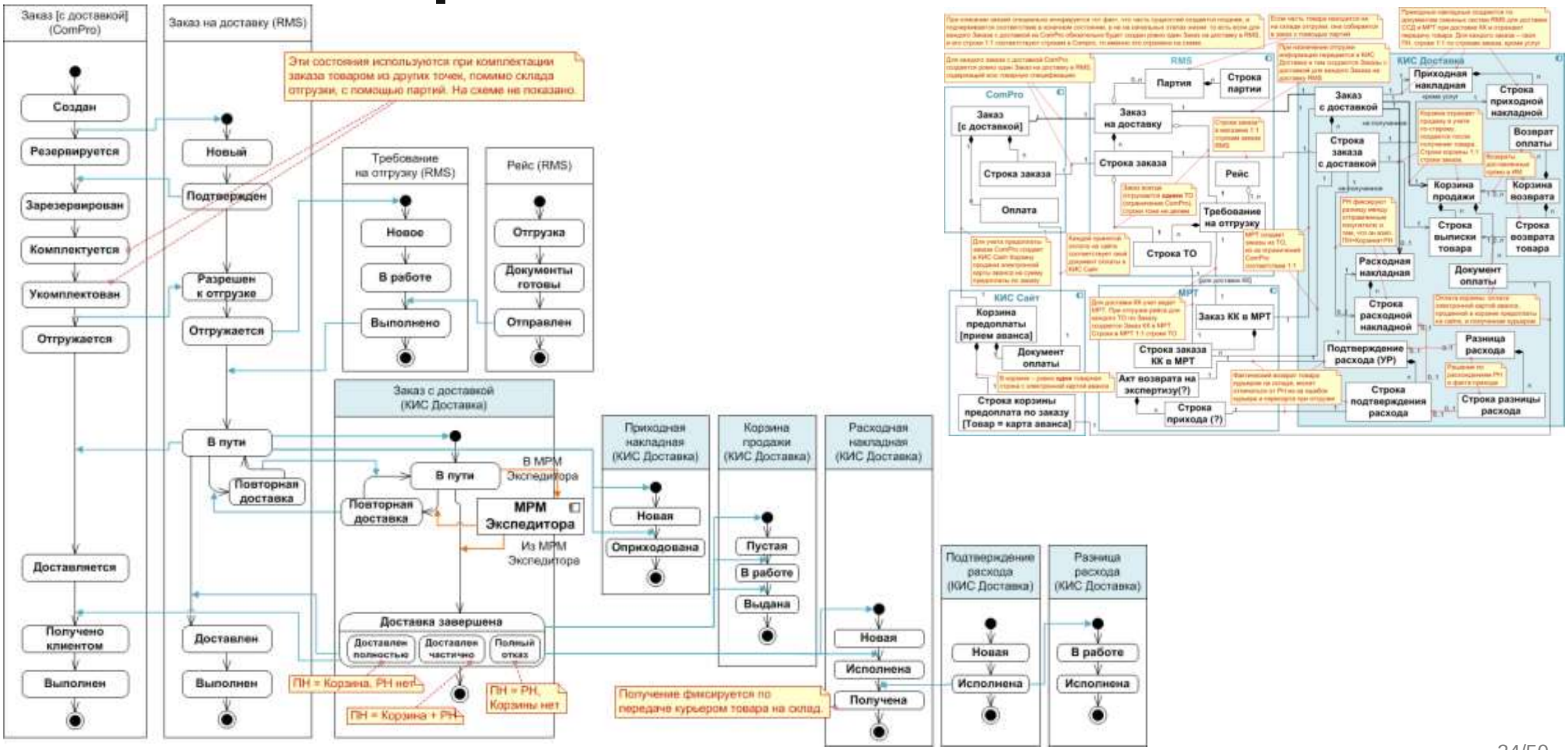
- Состояние заказа — единственный атрибут.
- Отражение оплаты — две суммы в заказе.

Усложнения:

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



Workflow через множество систем



Не забываем о пользователе за workflow!

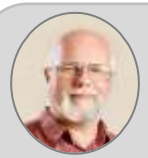
Запрос на доработку. Сейчас при вводе оплаты система автоматически привязывает её к первому не закрытому договору клиента. Надо уметь указывать договор при вводе платежа, так как клиент может внести предоплату по новому договору, не закрыв старый, чтобы ему отгрузили новый товар.

Предложено решение: убрать заполнение договора в сервисе создания платежей, добавить поле договора на форму ввода платежа со списком незакрытых договоров, и если там один договор, то заполнять поле автоматически.

Вопрос: а как бухгалтер, вводя платёж, узнает, по какому он договору?

- Если это написано в тексте основании платежа, то можно определить. А если нет?
- Можно спросить у менеджера, но платежи бухгалтеру надо ввести быстро — он не спросит.
- В любом случае, решение, когда для ввода надо спросить другого пользователя, сомнительное.

Хорошо, если вопрос возникнет на этапе оценки постановки, а не после реализации!



Указание пользователя конкретизирует встройку в надсистему, указывая конкретный элемент, и мы проверяем работоспособность связи — компетенции пользователя, необходимые для её работы.

Процесс в целом: выходим за рамки системы

→ Если работать только со взаимодействиями пользователя, явно относящимися к системе, то не все шаги бизнес-процесса могут быть поддержаны.

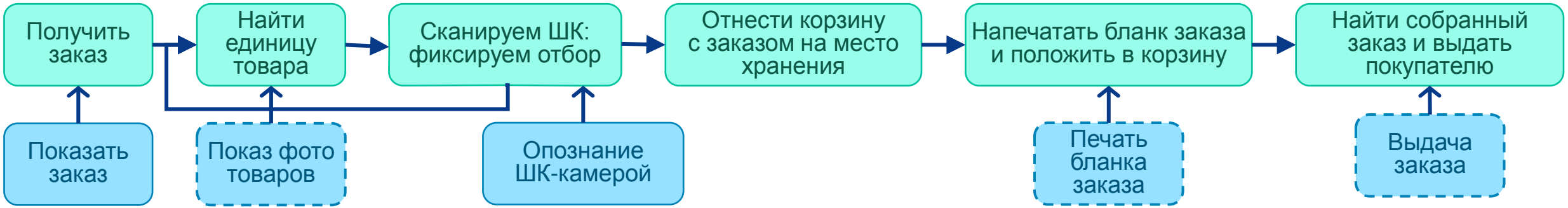
Запрос на доработку. Поддержать на мобильном рабочем месте отбор одежды по заказу в торговом зале магазина.

Очевидное решение — два шага: получить заказ и фиксировать товар поштучно.

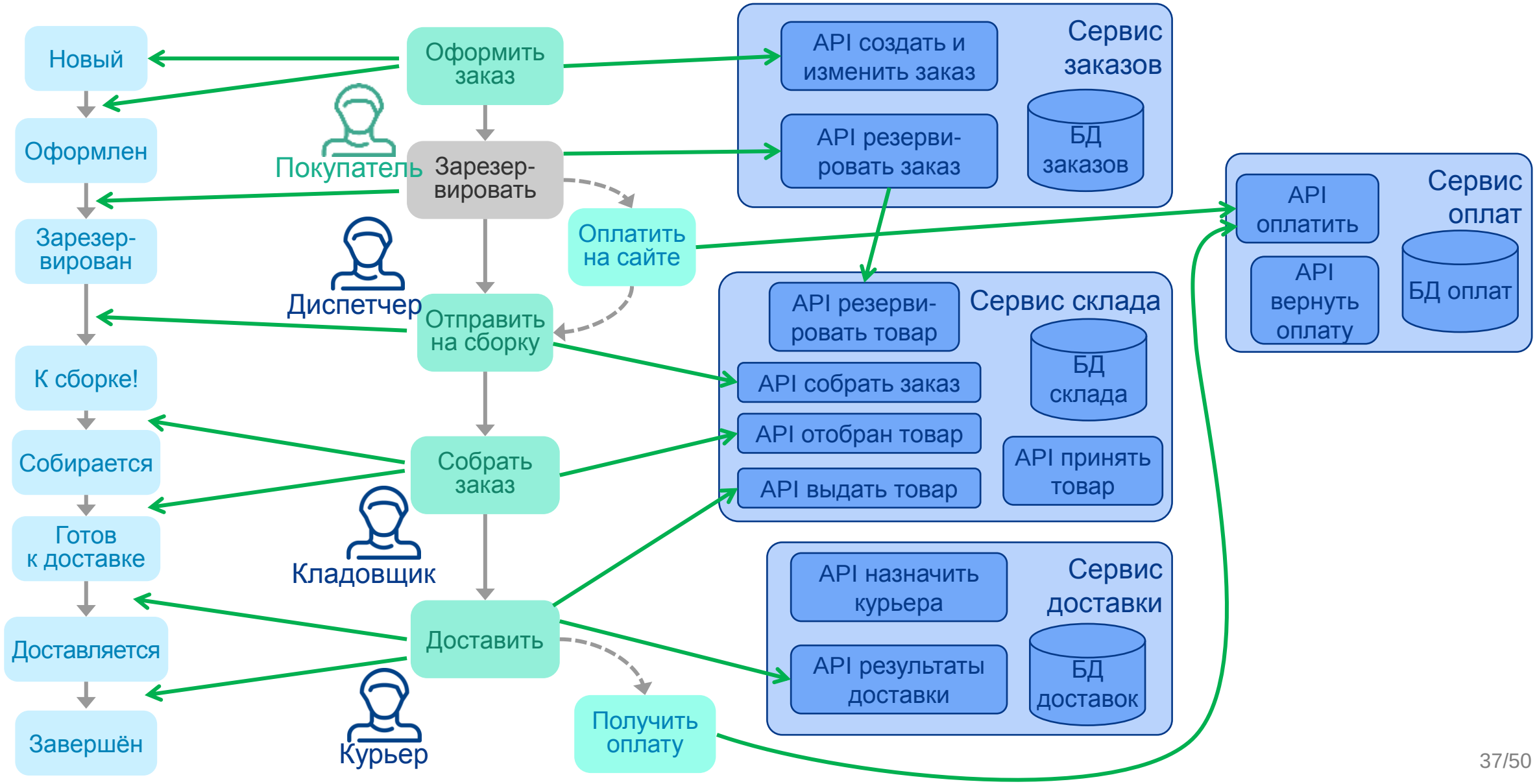
Проблемы:

- Как продавцу быстро найти нужное? Модели и цвета похожи, смотреть этикетки — долго.
- После отбора заказ надо куда-то положить, чтобы потом выдать. Как?

Проблемы видны, если представить процесс отбора по шагам и подумать, какая поддержка нужна в системе для каждого шага процесса.



Сервисы вместо workflow



Сервисы: 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 возможны 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?

Что изменилось в сервисной архитектуре

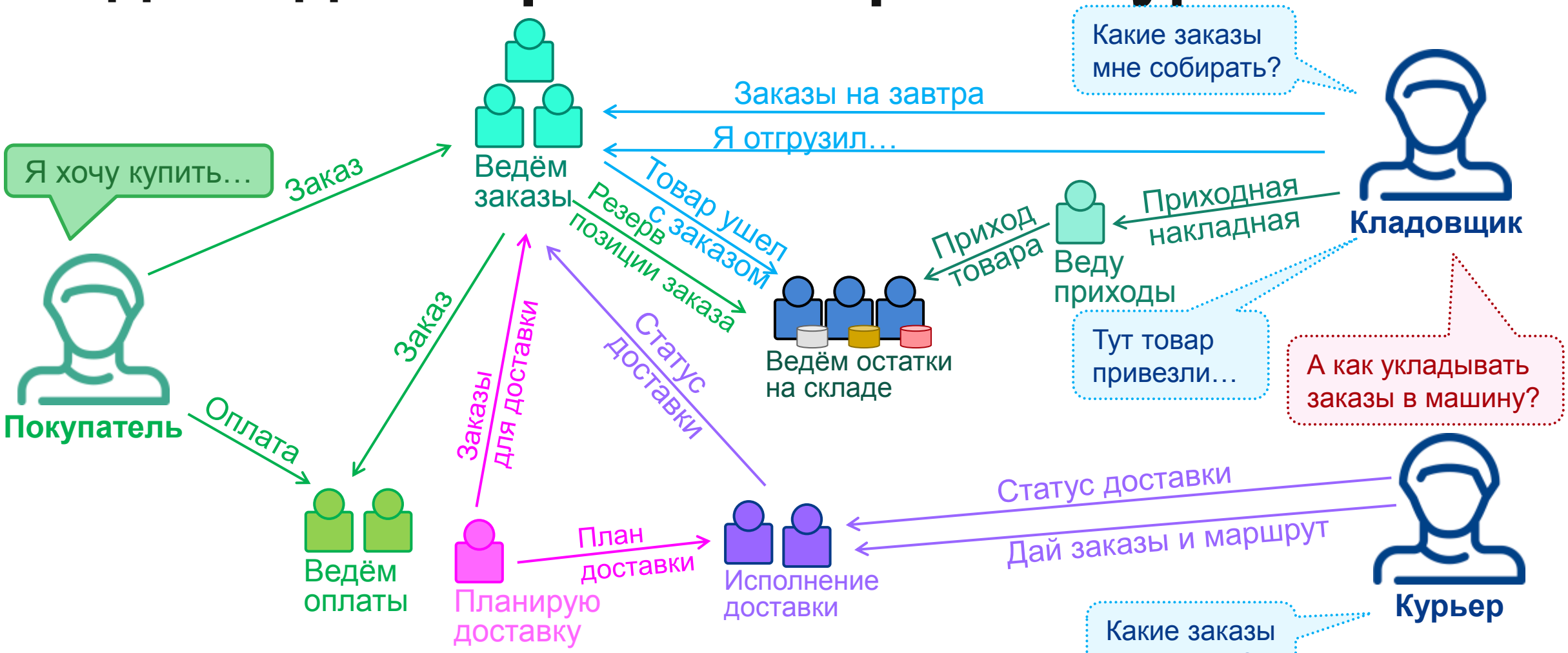
Микросервисы: масштабируемость и быстрые доработки приложений

- Каждый бизнес-запрос обрабатывает много сервисов, нет общей транзакции
- Много экземпляров одного сервиса для масштабирования, каждый может упасть по ошибкам или блокировкам, а система должна работать устойчиво
- Асинхронные сообщения, очереди выравнивают производительность

Отказ от единой реляцинной СУБД – она не справляется

- NoSQL-базы данных, использование многих БД одним приложением, In-memory хранение в БД и очередях, отложенный сброс в хранилища
- Кластерное развертывание с независимым хранением на узлах
- Транзакционность и консистентность обеспечивает приложение, а не БД
- При восстановлении узла кластера данные неконсистентны

Модель для сервисной архитектуры



Из выступления «Визуальное проектирование масштабируемых приложений»

Event Storming и бизнес-процессы

BPM — моделирование бизнес-процессов:

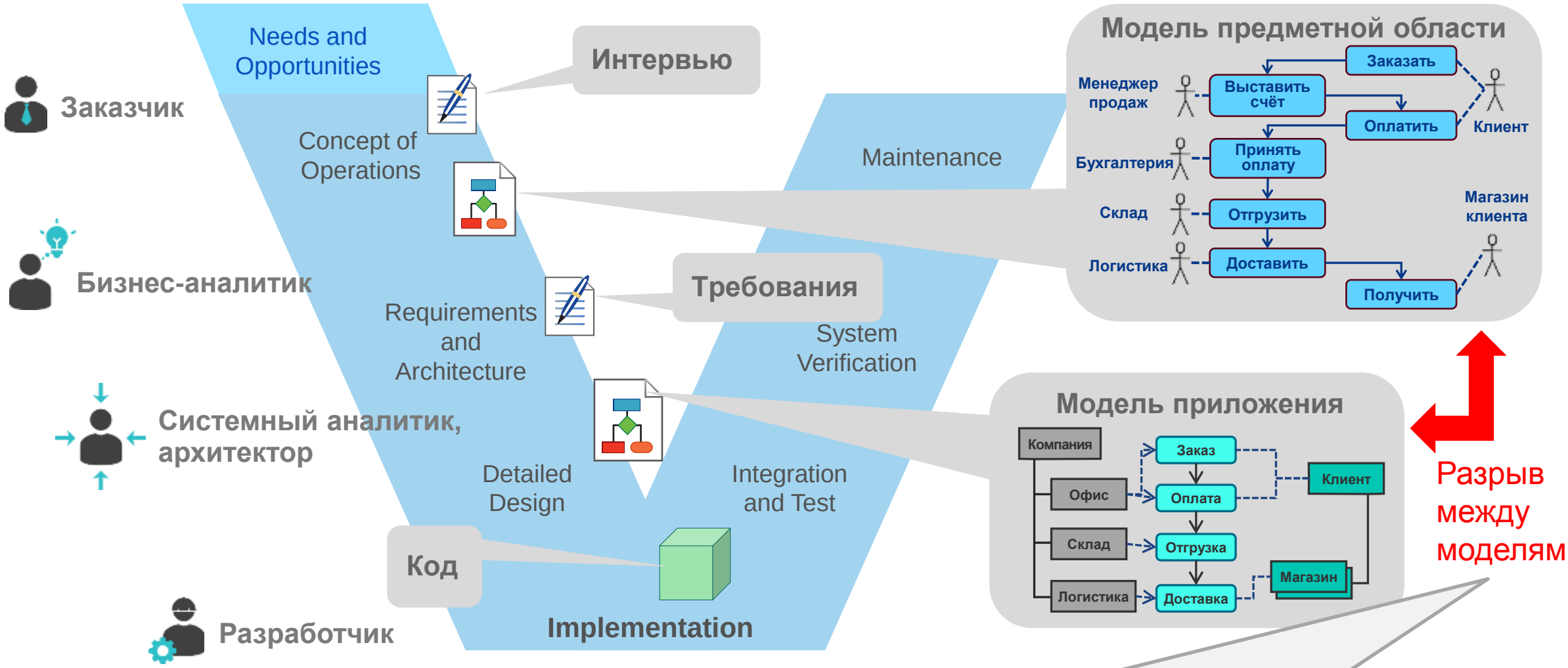
- Представляет деятельность как последовательность шагов с ветвлениями
- Хорошо подходит для основного потока операций, плохо — для исключений
- Есть работы, которые «состоят из исключений»: выверка отчётов, увязка планов, для них подходит Case Management, а не Process Management

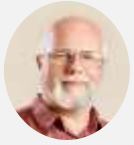
Event Storming:

- Выдаёт событийную, а не процессную модель устройства деятельности
- Хорошо подходит для автоматизации «от событий» в сервисной архитектуре
- Помогает разобраться в бизнесе, но не даёт целостного представления
- Восстановить целостное представление, связав разные события, можно по-разному: через схему бизнес-процессов, систему целей или учёт

DDD – единая модель бизнеса и софта

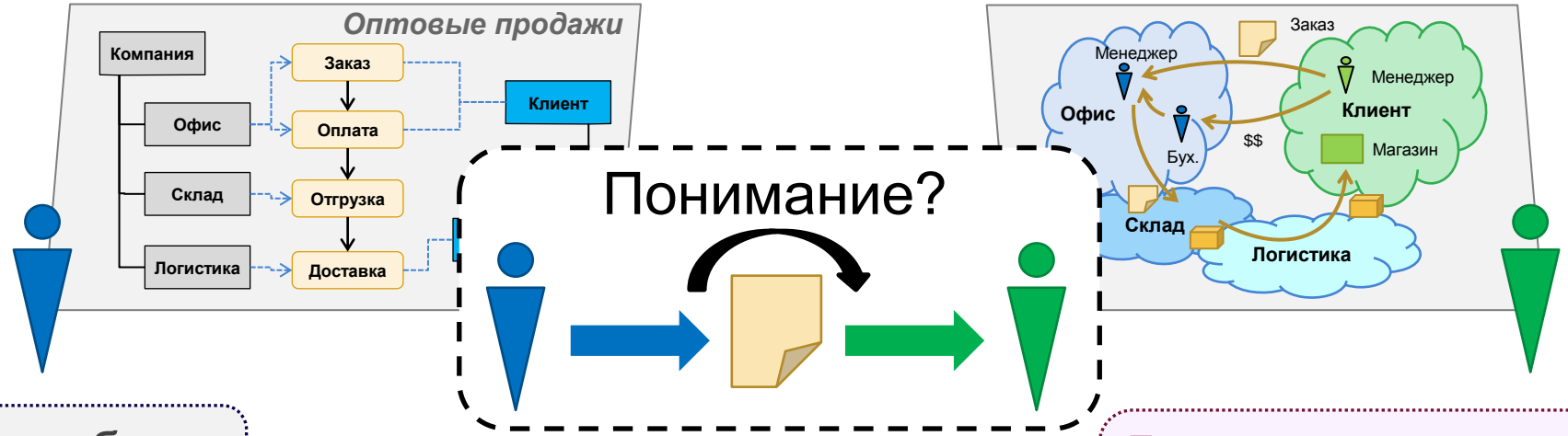
Поддержка артефактами (классика)

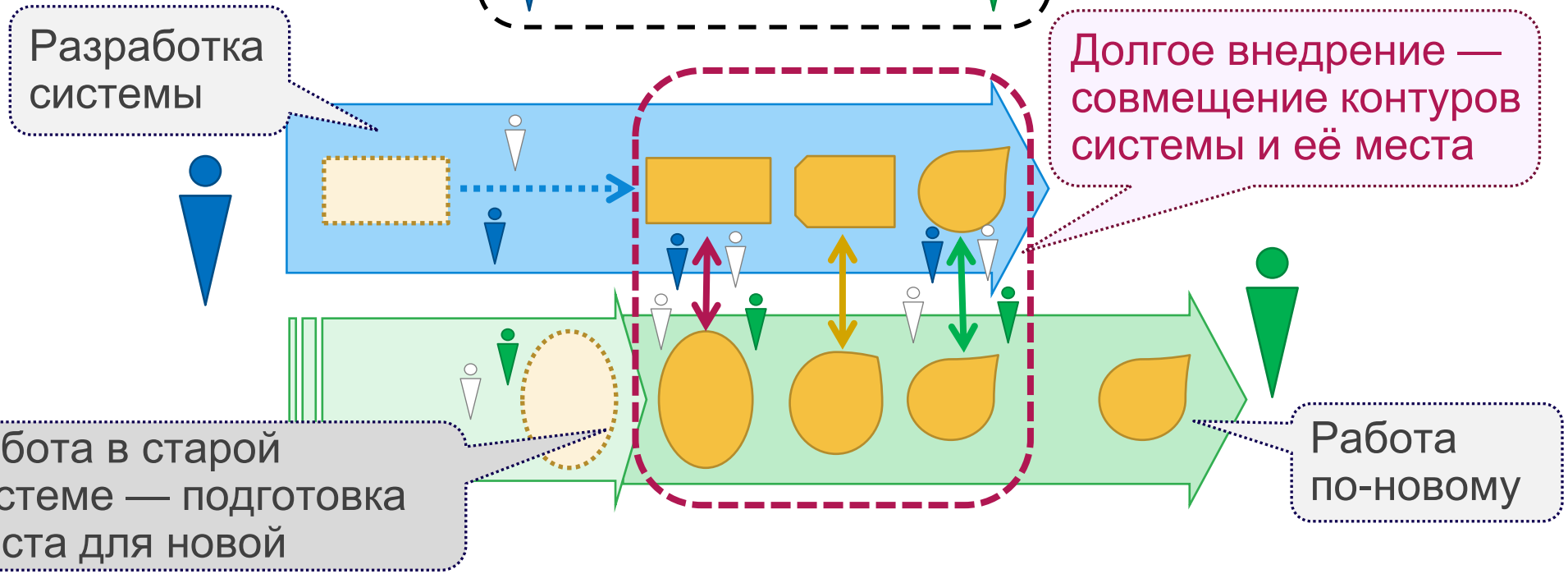




Проблема: каждый уровень отдельно сложно изменять и поддерживать соответствие.

Непонимание влечёт долгое внедрение





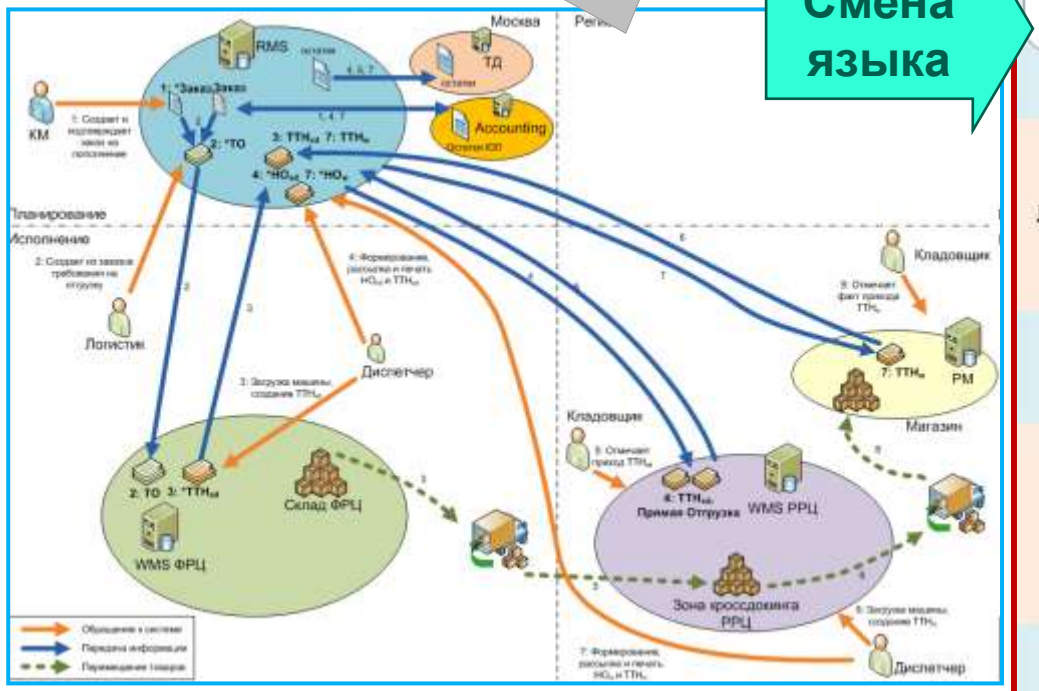
Формализация снабжения магазинов

Объекты — Class Diagram

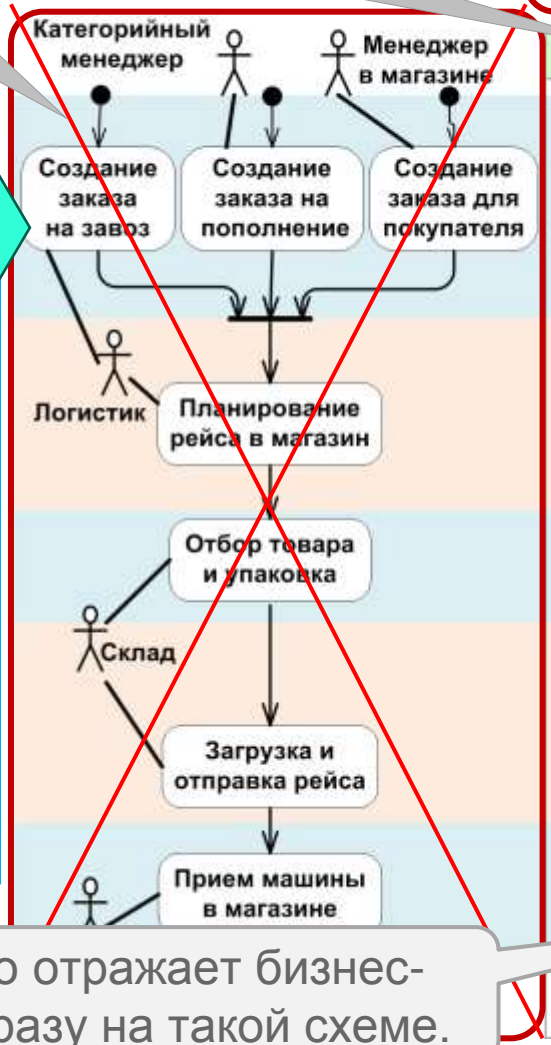
Бизнес-процесс — Activity Diagram

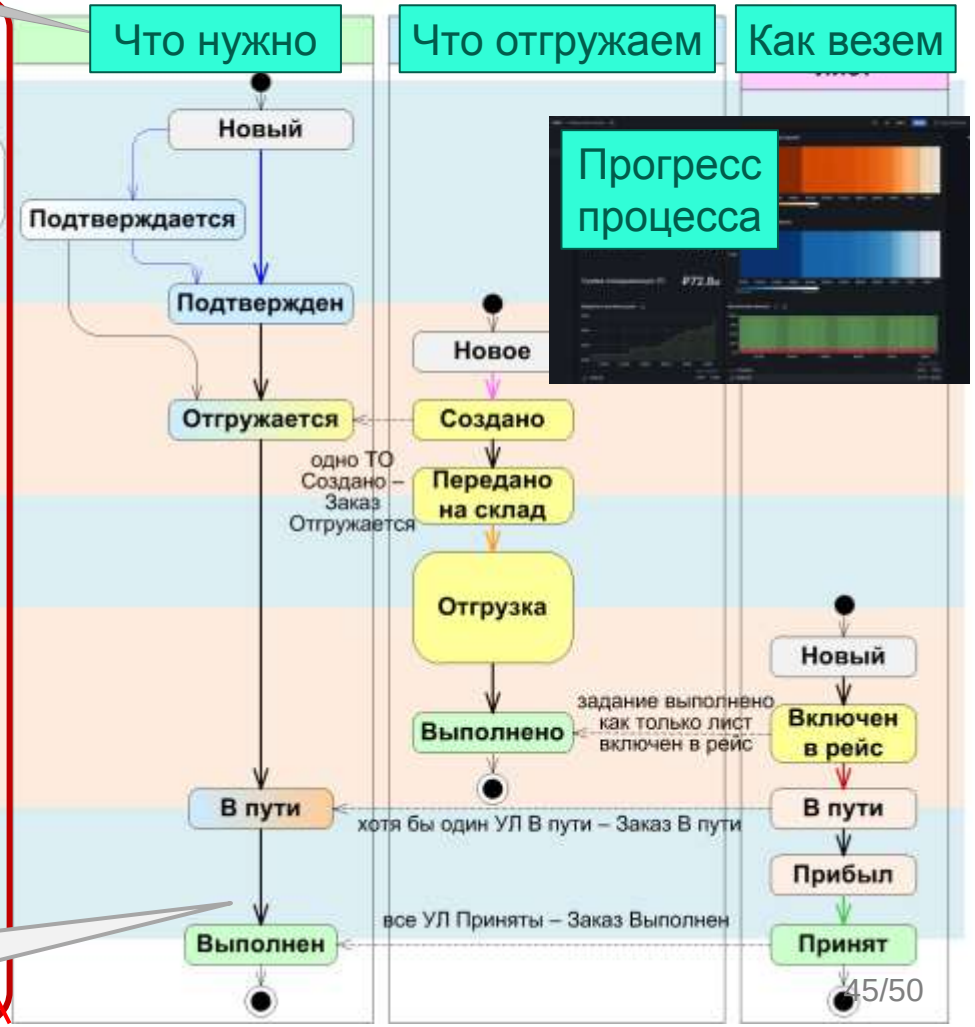
Состояния документов — State Diagram

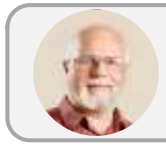
Неформальная схема деятельности и её отражение в существующих системах



Смена языка



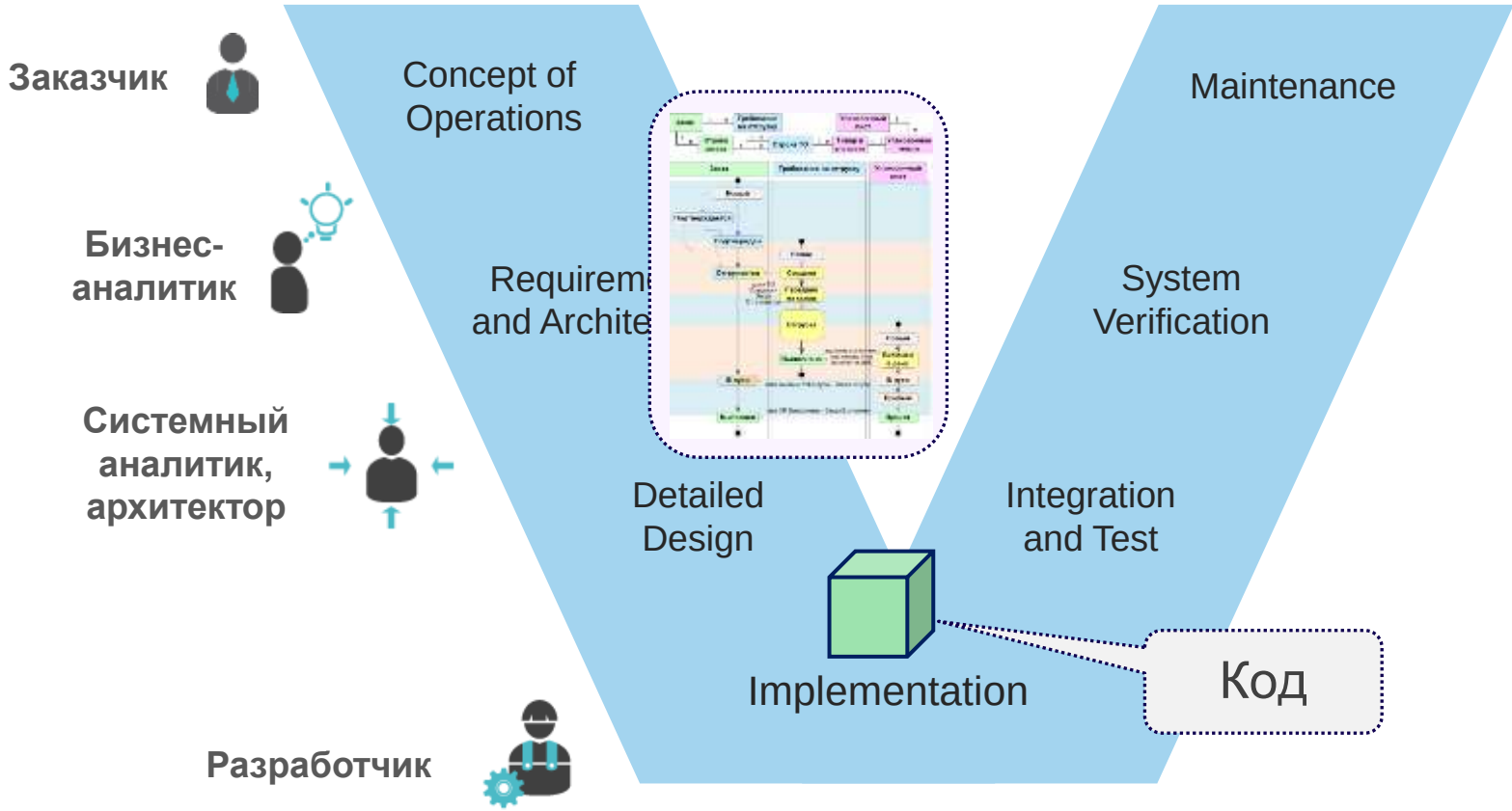




Если workflow документов прозрачно отражает бизнес-процессы, то можно обсуждать их сразу на такой схеме.

Domain Driven Design

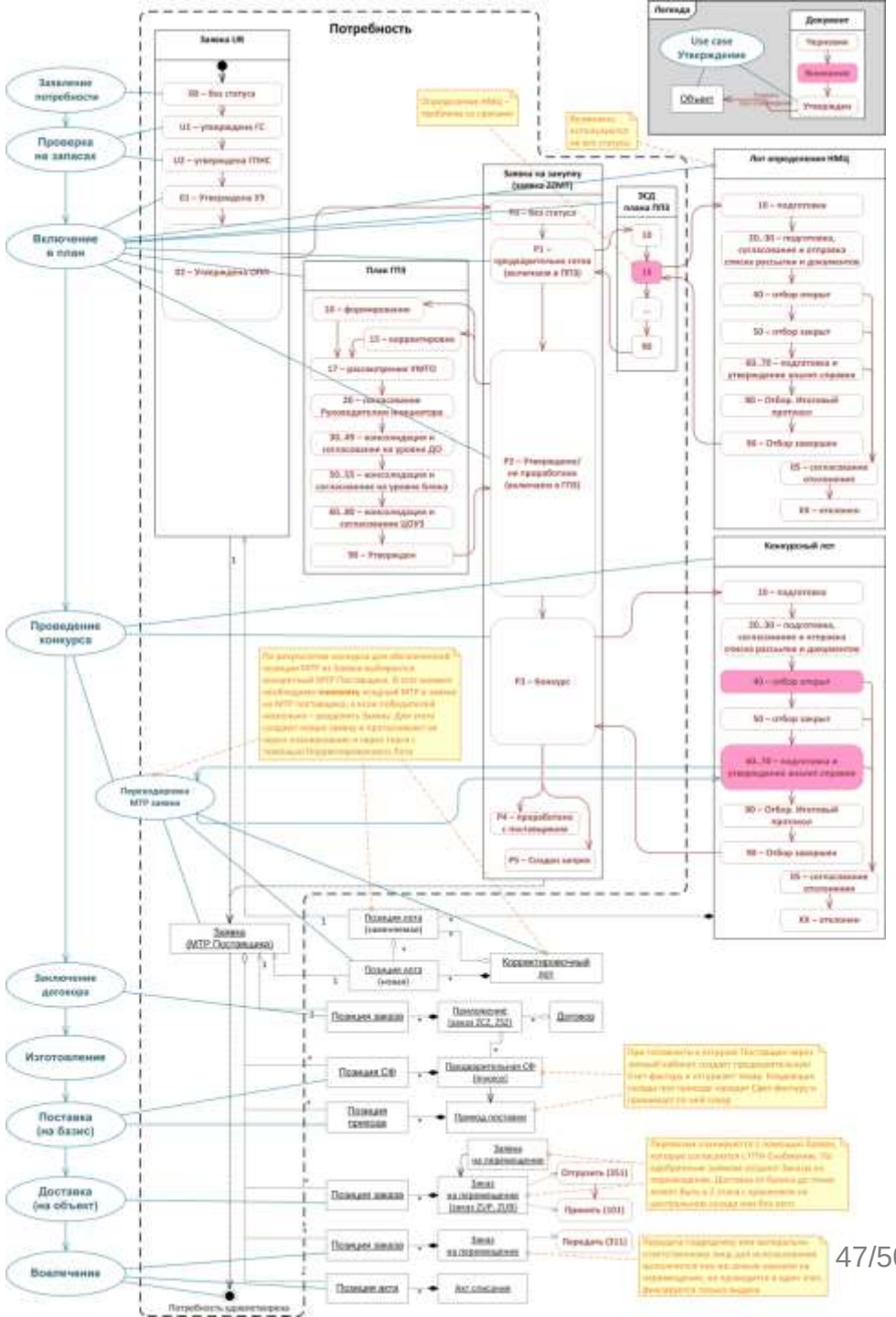
- Единый язык:
  - на основе терминов предметной области;
  - понятен всем участникам проекта.
- Единая модель, приложения и его встройки в бизнес.
- Прозрачное отражение модели в код.



У меня есть много выступлений о DDD, последнее — «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.



Три категории постановок

1. Разработка новой системы поверх фрагментарной и малой автоматизации:
   - выясняем модель бизнес-процессов,
   - создаём модель системы,
   - работаем над её встройкой в процессы, изменяя их.
2. Доработка существующей системы: проектируем изменения модели существующей системы и её встройки в процессы.
3. Разработка новой системы, заменяющей существующую:
   - существующие процессы несут отпечаток старой системы, **его надо снять,**
   - проектируем новую систему и процессы,
   - проектируем работу на переходном этапе, обеспечивая мягкую замену.

Большинство методов анализа и проектирования создавались, когда бизнес-процессы были слабо автоматизированы, в расчёте **на первую ситуацию.** Сейчас мы имеем дело **со второй и третьей.**

Как фиксируется контракт бизнеса и ИТ?

- Классика: требования в виде объектов и функций и ПМИ
  - Мина: при внедрении окажется, что софт не подходит пользователям
- Требования в виде макета интерфейсов и сценариев работы в виде user story или use case
  - Макеты интерфейсов согласовывать легче, чем требования
  - Мина: окажется, что для функций в цепочке не будет хватать входных данных.
- DDD: согласованная модель приложения
  - Мина: проектирование — существенная часть проекта, сложно оценить заранее
  - Мина: заказчик не всегда может разбираться в модели
  - Модель — очень хорошая основа для развития системы
- Уровень бизнеса: обеспечение возможностей и решение проблем, при ограничениях на время, стоимость и сроки
  - Мина: что всё-таки будет решено — неясно, итерации этот риск уменьшают

Итоги: архитектура предприятия

СТАЧКА

- **Думайте о предприятии как о системе бизнеса и софта**, ищите схемы, которые хорошо проявят связь между ними и покажут важное
- Не забывайте о **целях проекта по изменению бизнеса**
- Нужны объекты, с которыми работают пользователи: словарь или полноценная объектная модель
- Описание бизнес-процессов не обязательно, особенно когда оно существует лишь формально, возможны альтернативы:
  - user story или use case и story mapping или CJM или JTBD
  - event storming
  - workflow документов с ролями пользователей

Обратная связь

Максим Цепков

http://mtsepkov.org

@MaximTsepkov

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