ARCHDAYS 2024

Бизнес и софт как единая система — описываем архитектуру предприятия

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

http://mtsepkov.org

CUSTIS

1 ноября 2024, Москва

О чём этот доклад?

- ИТ-проект начинается с потребностей бизнеса.
- И его результат — софт — встраивается в бизнес и его изменяет.
- Системный подход учит, что проектирование системы всегда начинается с понимания надсистемы и места системы в ней.
- Попытка работы от границы ИТ-системы, от требований, приводит к тому, что софт оказывается не слишком пригоден, требуется долгая доводка.
- А по мере развития ИТ, бизнес и софт всё больше интегрируются, граница между ними становится подвижной и условной.
- Поэтому необходимо понимать и описывать устройство бизнеса и встройку в него софта как единую систему.

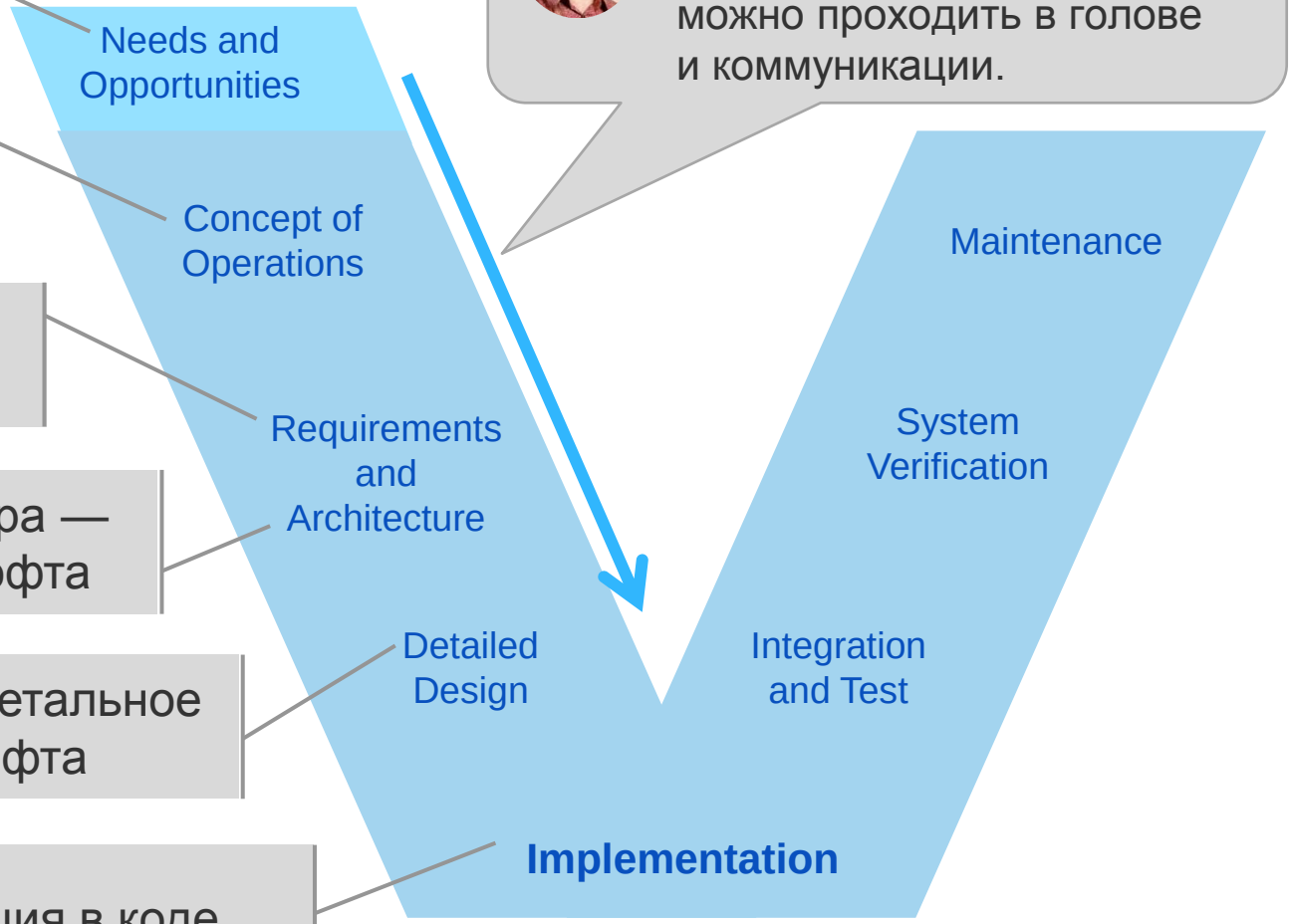
Разрыв между ИТ и бизнесом

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.

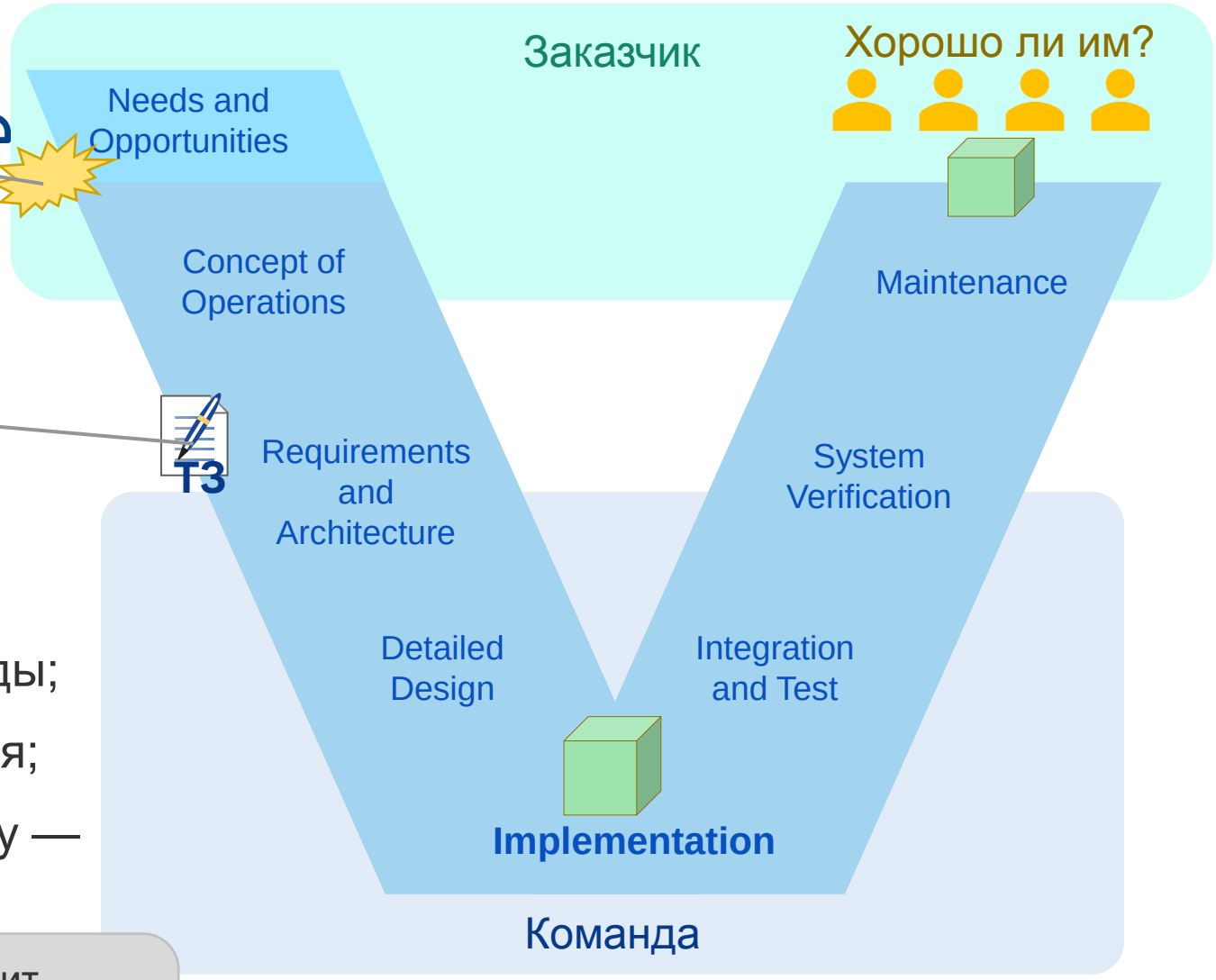


[V-Model (Wikipedia)]

ТЗ — способ сменить ответственность

Идея: сделать хорошо людям

Задача: напишите софт, удовлетворяющий требованиям

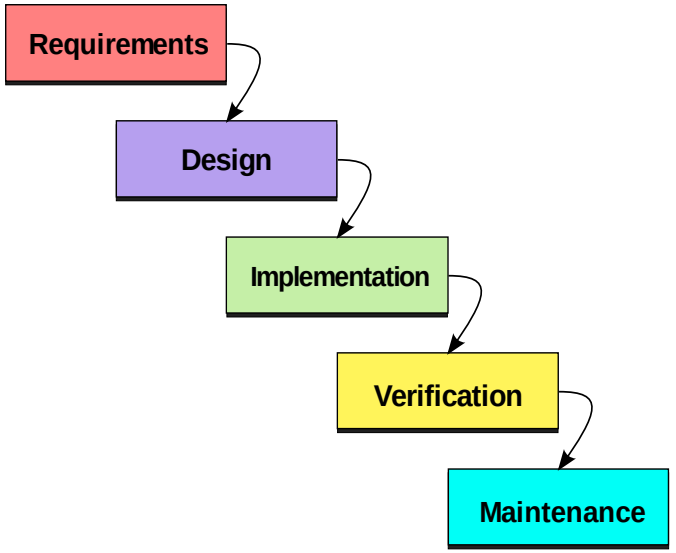


- Ответственность за достижение целей проекта вынесена за пределы ИТ-команды;
- Как следствие, цели часто не достигаются;
- Выделение этапа ТЗ не решает проблему — ТЗ утверждают, а гарантий оно не даёт.

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 процедур.
- 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

История культур ИТ-проектов

Рамка проекта:

ИТ-система…

обеспечивает бизнес

делает то, что нужно

сделана вовремя

работает

1960 1985 2003 2013

Эпоха НИОКР — создаём софт для решения заданной задачи так, как создают другие инженерные изделия, например автотехнику.

Большие компьютеры

Эпоха RUP — учимся создавать софт по проекту в заданные бюджеты и сроки, как в производстве других отраслей.

Провал

Время персоналок и Scrum: софт для бизнеса, задача меняется, пока идёт разработка.

Public web и продуктовый подход: софт становится основой для бизнеса.

Каждая эпоха порождала свои учебники, эпоха ушла, а они — остались.

ИТ- и бизнес-проект объединяются

В современных проектах **граница** между ИТ-частью проекта и изменением бизнеса **подвижна,** и о ней надо принимать решения в ходе проекта. **Диджитализация** означает, что **проект будет единым, а ИТ-часть — главной!**

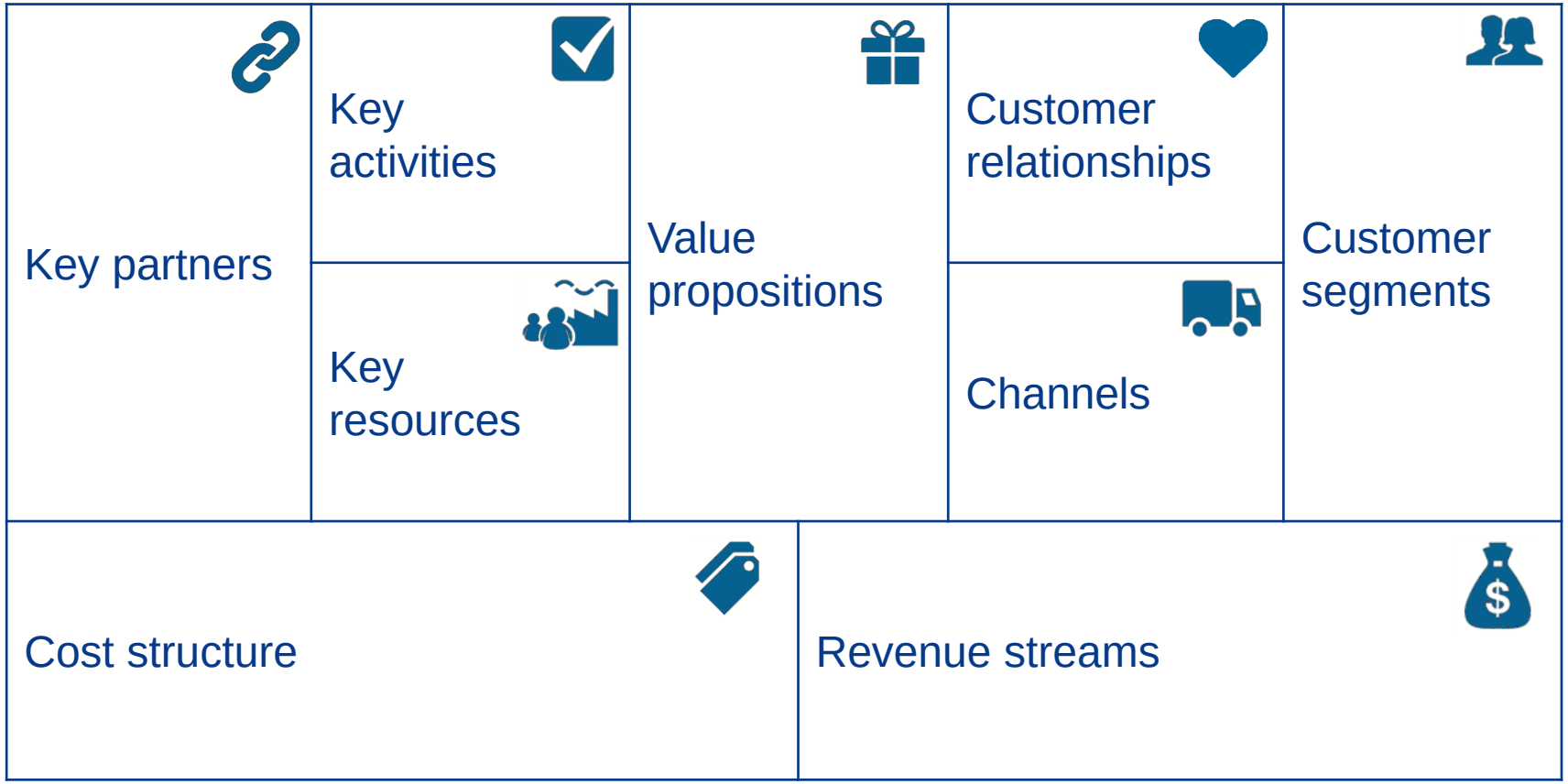
Big Picture бизнеса — бизнес-уровень в целом

Продукт как возможность

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…

Описание создания ценности — Canvas Александра Остервальдера



Система и её окружение

Целевая система — та, над которой мы работаем в настоящий момент:

- различаем описание системы и её саму в реальном мире.

Целевая система встроена в **надсистему**:

- выполняет определённые функции в ней,
- отвечает на интересы агентов надсистемы.

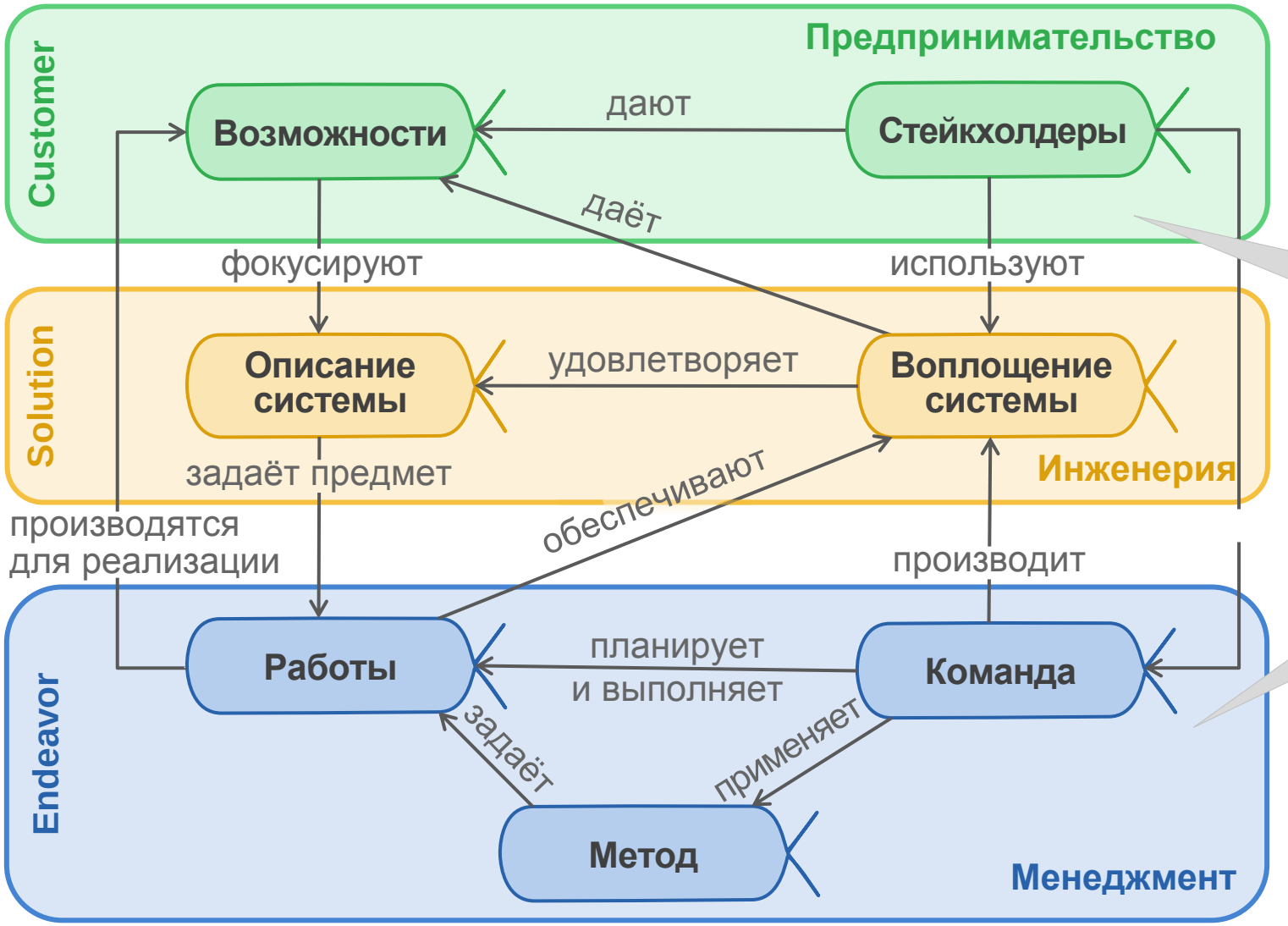
Эксплуатирующая (обеспечивающая) система обеспечивает работу целевой, обычно включает **команду** и другие части, живые и неживые.

Создающая система — та, что создаёт и развивает целевую.

Окружение системы — смежные системы, с которыми целевая взаимодействует в процессе своей работы.

Нетривиальные связи — закон Конвея: отражение в структуре создаваемой ИТ-системы орг. структуры команд создающей системы.

OMG Essence: сопряжение систем



Три системы:
надсистема, целевая и создающая;
для ИТ — бизнес, софт и команда.

Сопряжение с надсистемой:
ценность и люди

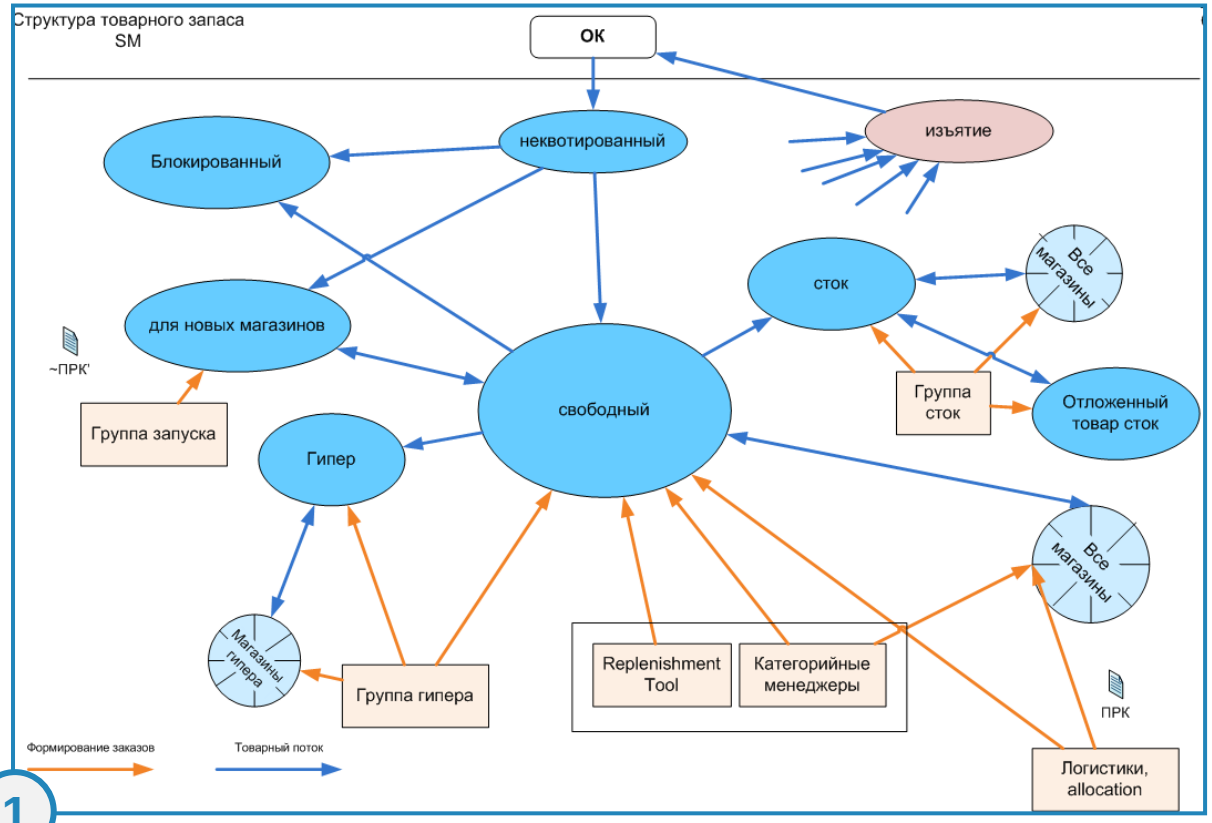
Создающая
и эксплуатирующая
система

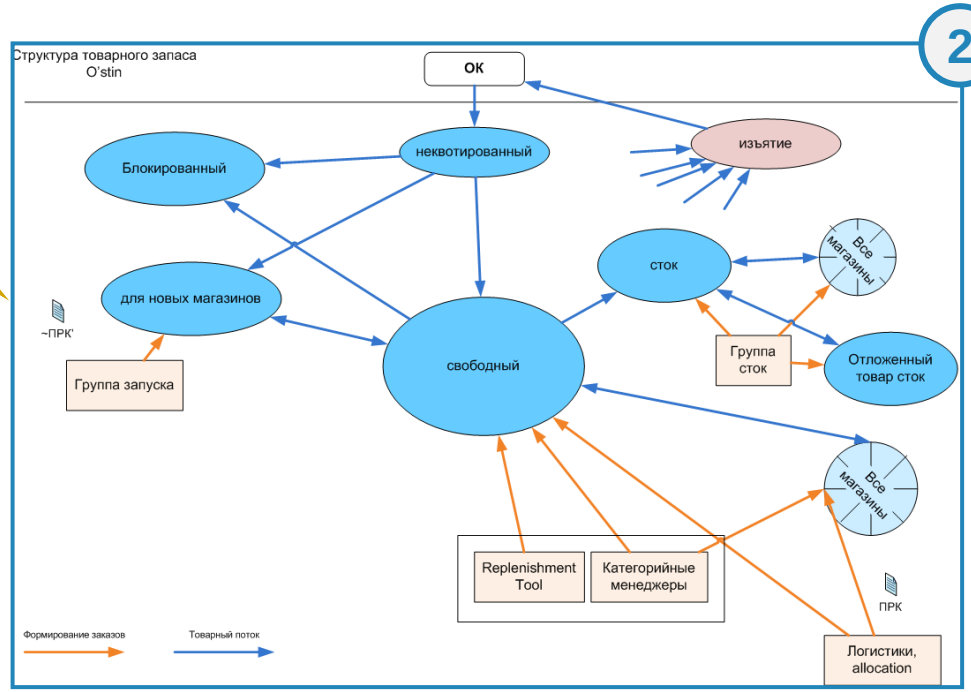
Каждую из систем изучает своя дисциплина: предпринимательство, инженерия и менеджмент.

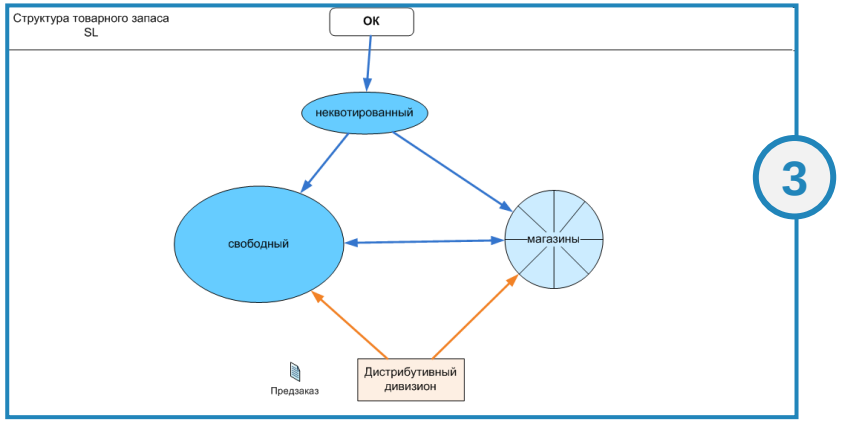
Визуальный образ описывает назначение

Пример:
деление товара

Варианты процессов
в простой нотации,
схемы понимаются на лету

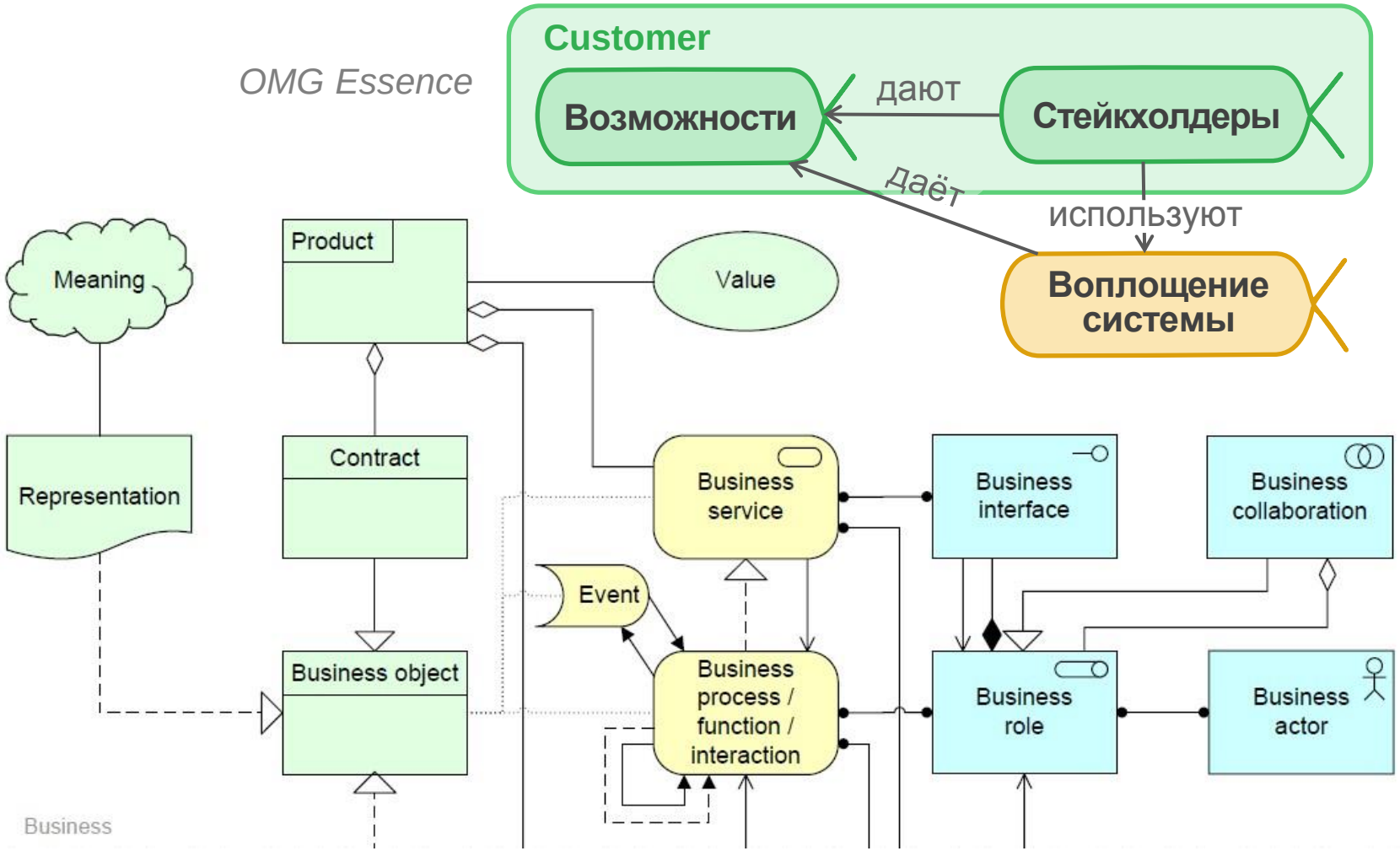


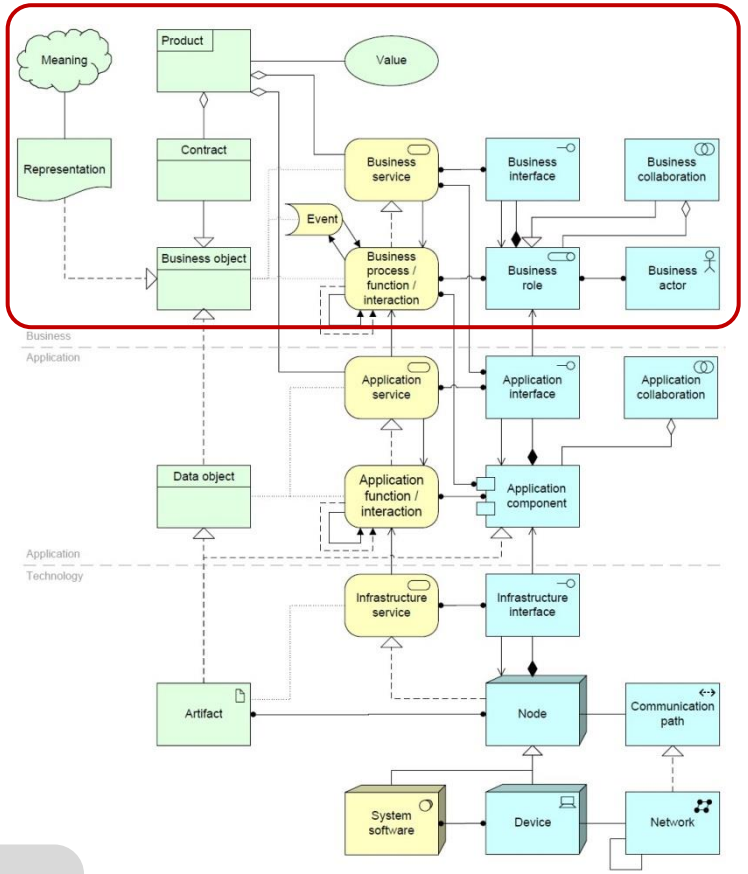




Бизнес-уровень в ArchiMate

OMG Essence

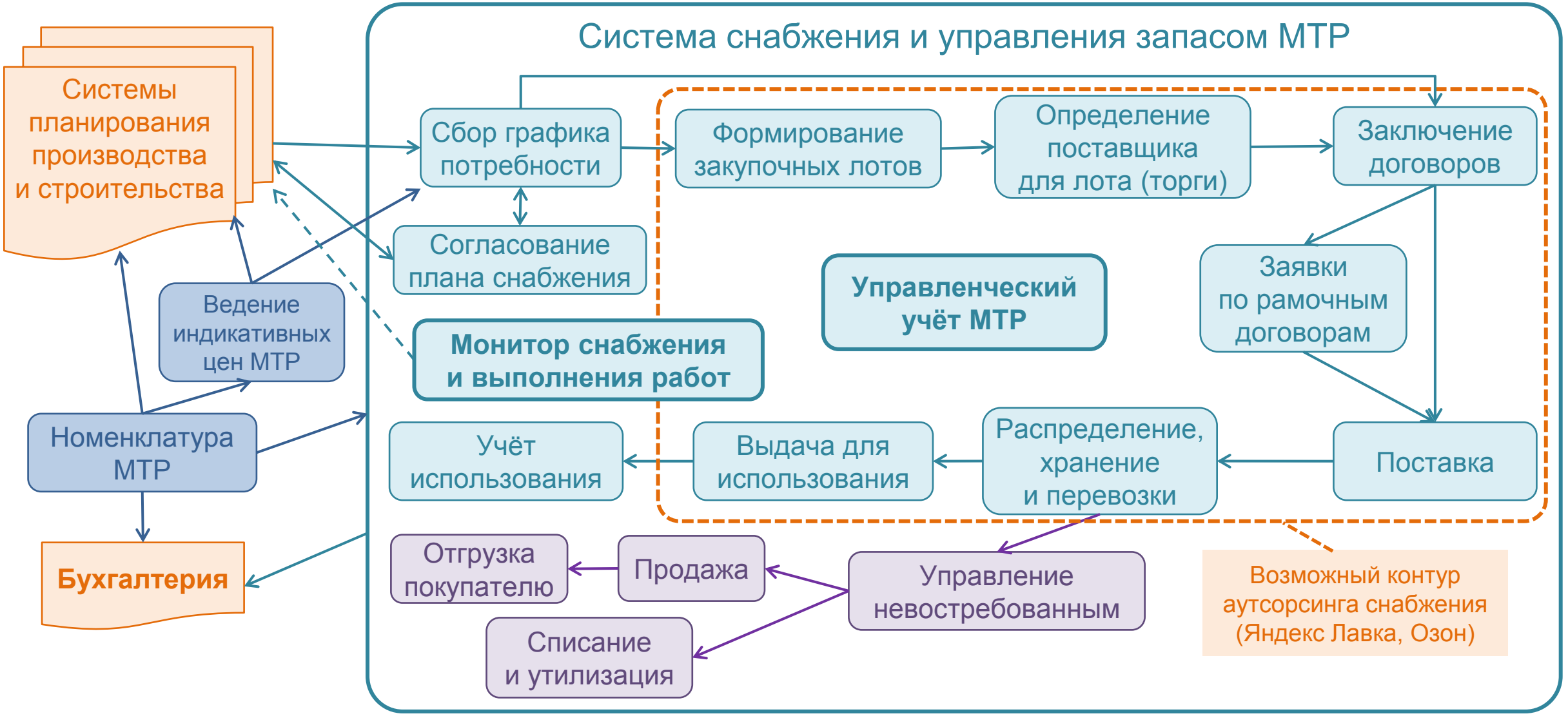




В основе ArchiMate — продуктовая модель: продукты несут ценность для потребителя, их обеспечивают бизнес-сервисы, включающие процессы и функции, выполняемые агентами.

Метамодель ArchiMate 1.0

Архитектура снабжения: крупные функции



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 в докладе «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.

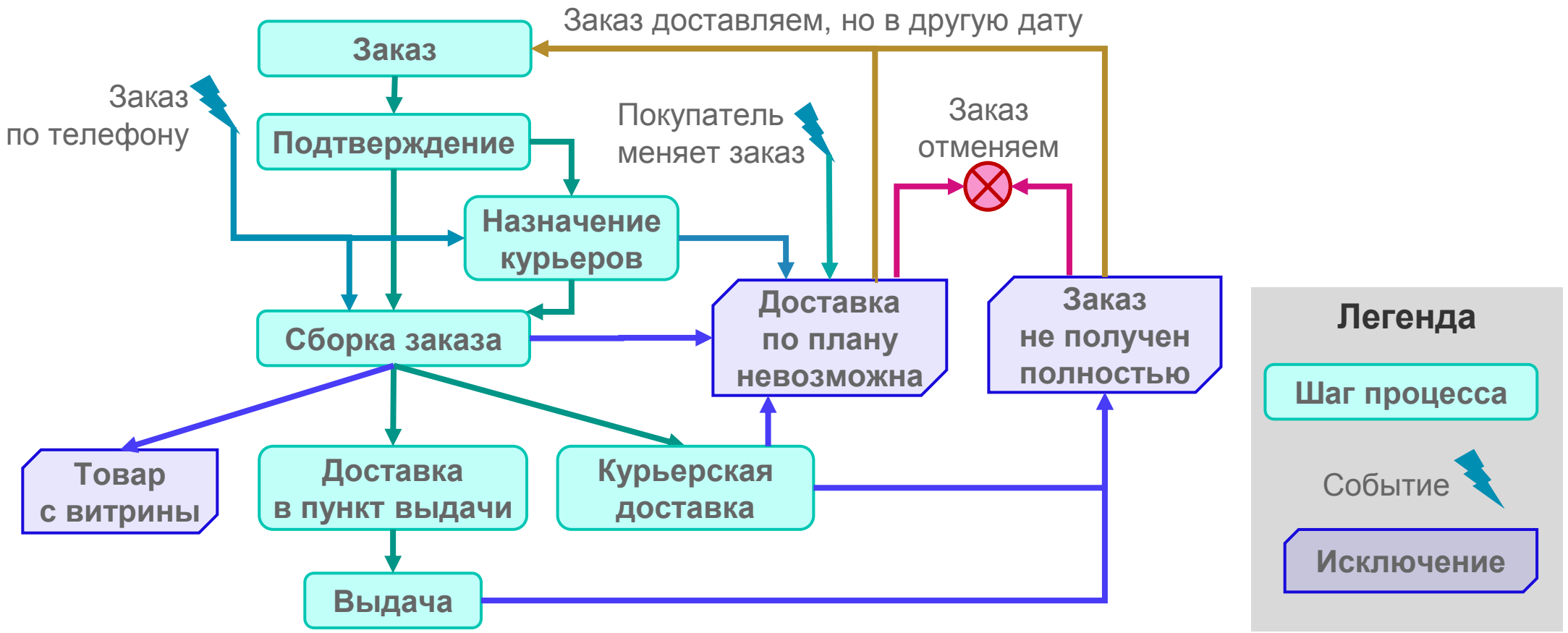
Слабо структурированная деятельность

- Есть довольно много примеров бизнес-функций, для которых сложно построить описание в виде процесса, при том что вовлечено много людей:
  - выверка бухгалтерской отчётности с решением обнаруженных проблем;
  - планирование расписания при недостатке ресурсов с переговорами;
  - поиск баланса между текущей работой, ремонтом (техдолг) и развитием;
  - а также обработка исключений и особых случаев.
- Не надо пытаться описать подобное языком бизнес-процессов!
- Альтернативой может быть координация по целям: критерии продвижения к цели и её достижения.

Это верно для любого метода: надо уметь видеть границы, и **не пытаться применять его не по назначению.** Апологеты методов часто выступают за повсеместное применение.

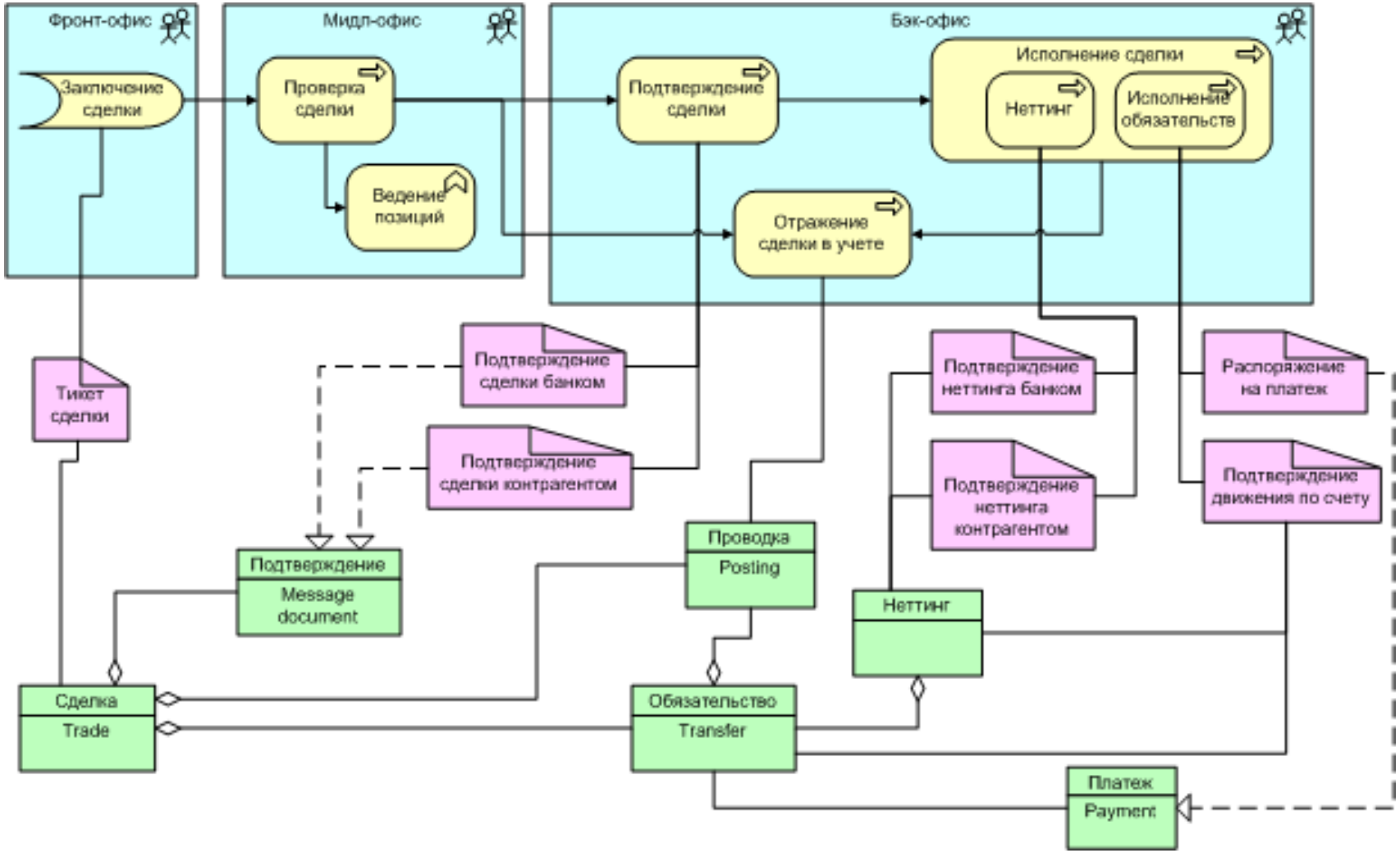
Исключения: Case Management

Процесс обработки заказа интернет-магазина с исключениями из моего доклада «Process и Case Management в информационной системе: от автоматизации As Is к поддержке развития бизнеса».



От бизнеса к софту

ArchiMate показывает связь уровней



Заложена гибкая связь бизнеса и софта

Гибко не получается

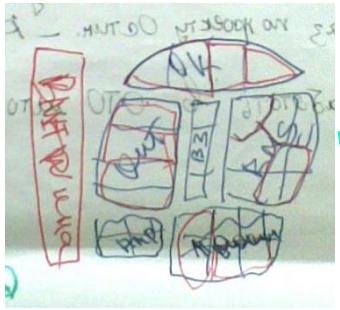
В основе — сервисная модель приложения

Метамодель ArchiMate 1.0

Оригинальный образ для отражения систем

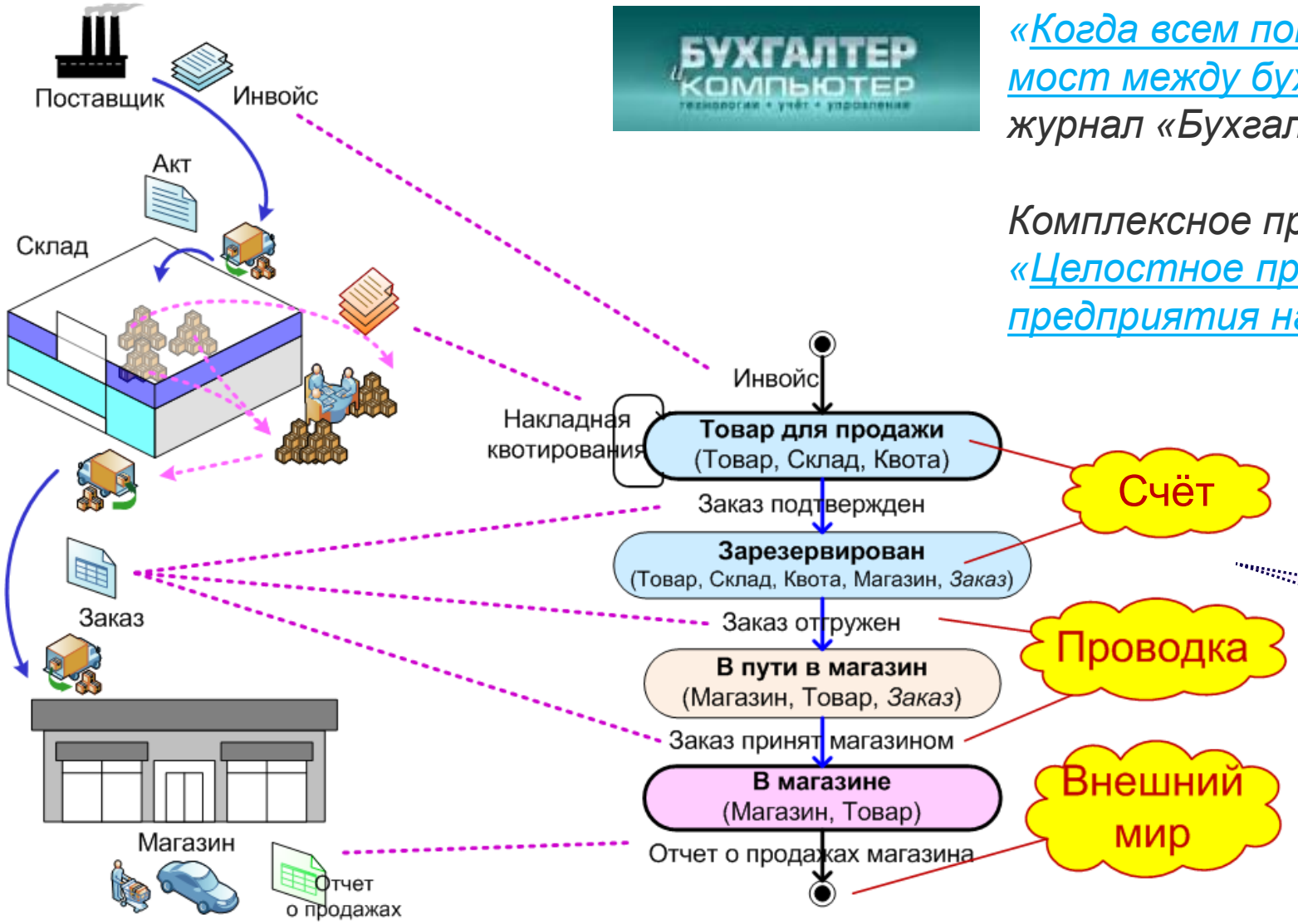
Пример: единая витрина для многих систем

Лупа: смотрим на сложную структуру



1 фаза

Учёт: бизнес как поток ресурсов

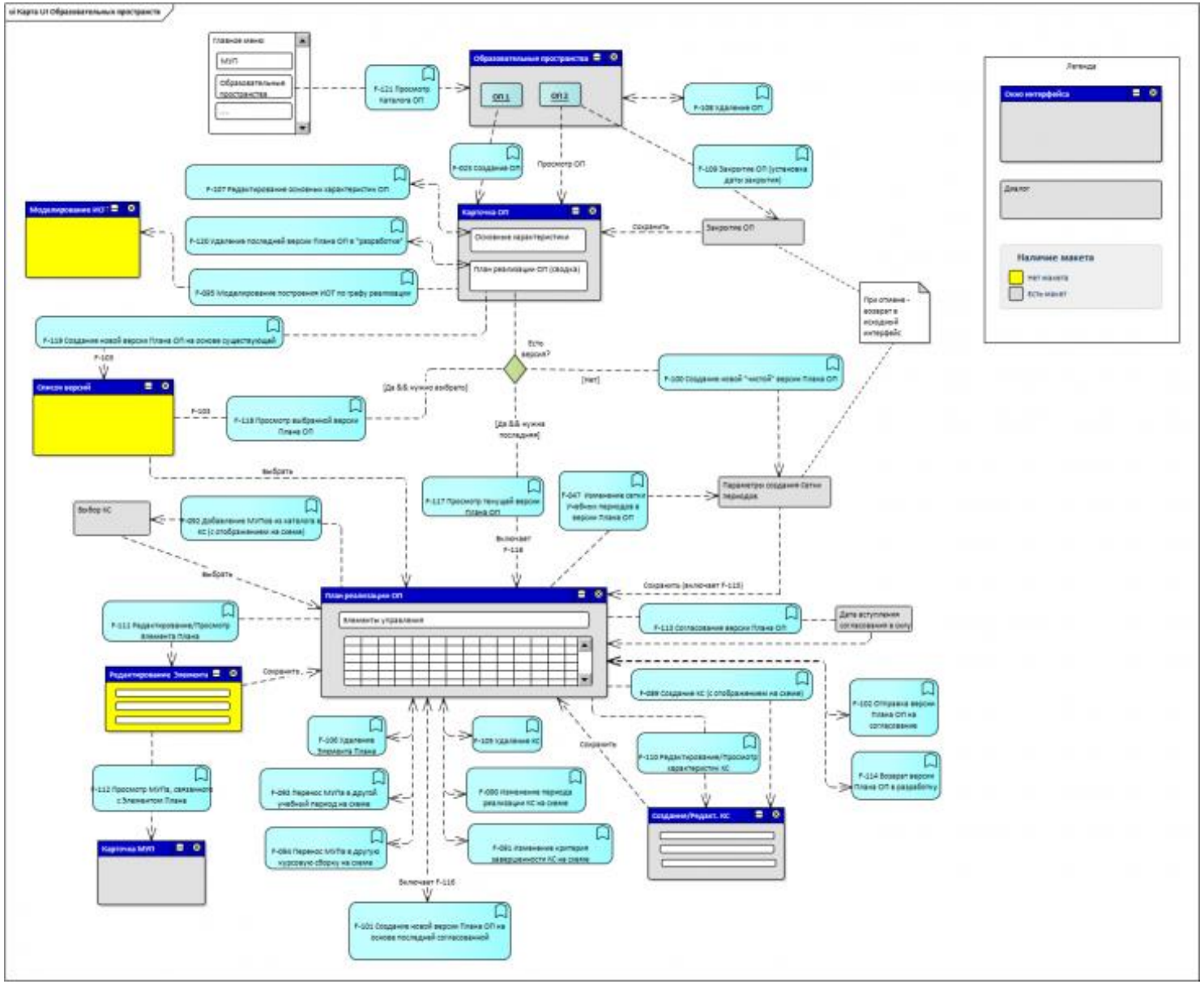


«Когда всем понятно. Диаграммы учёта: мост между бухгалтером и разработчиком» — *журнал «Бухгалтер и компьютер», №5-2011.*

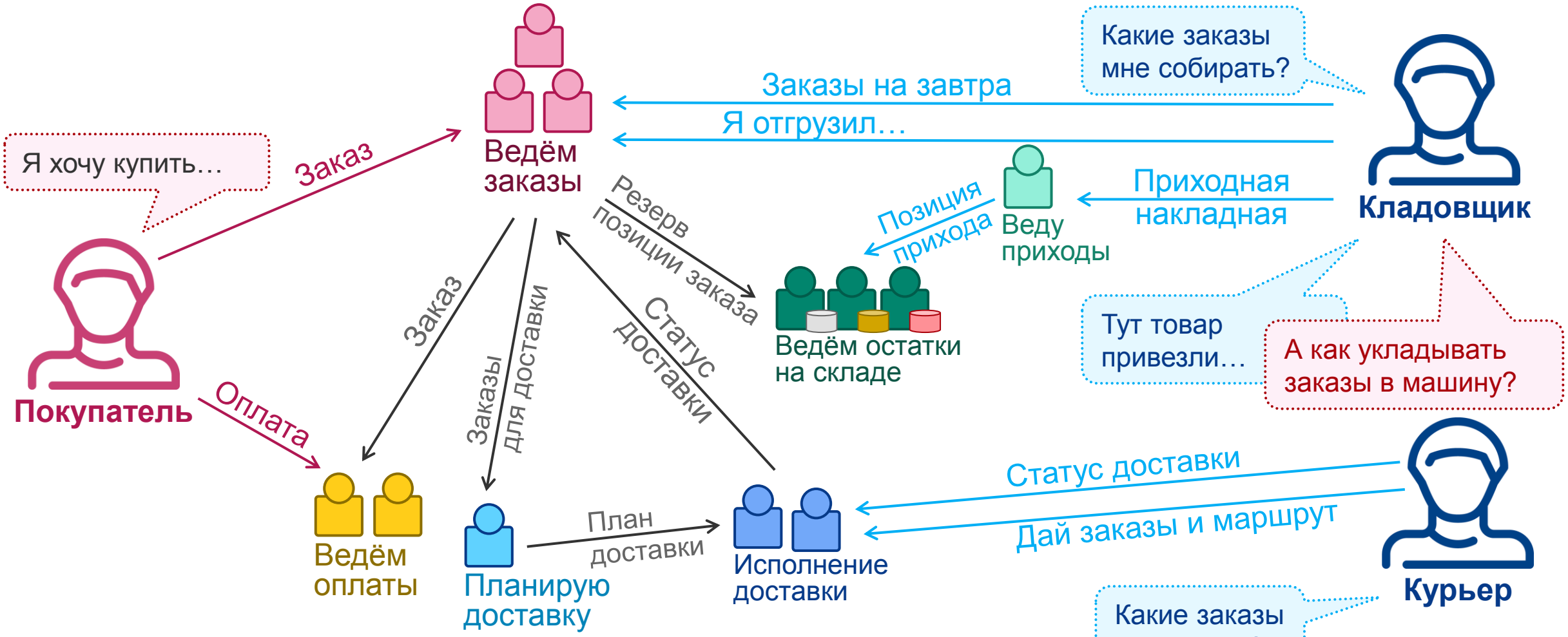
Комплексное представление — мой доклад «Целостное представление деятельности предприятия на диаграммах учёта».

Движение товаров
и других ресурсов отражается
как проводки между счетами

Поддержка бизнес-функций экранами приложения



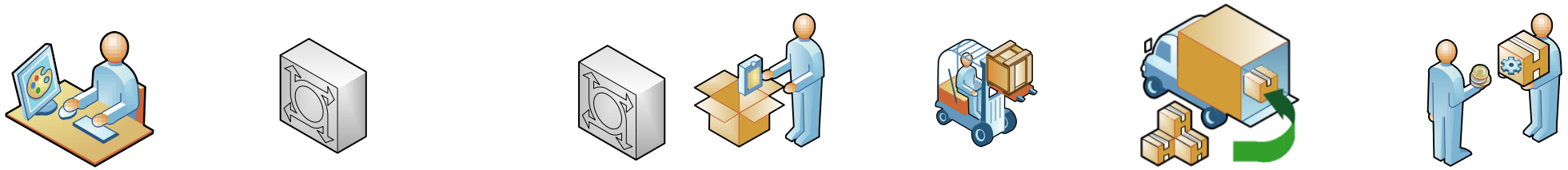
Модель для сервисной архитектуры



Из доклада «Визуальное проектирование масштабируемых приложений»

Переход между уровнями бизнеса и софта

Три уровня представления: обработка интернет-заказа



Деятельность

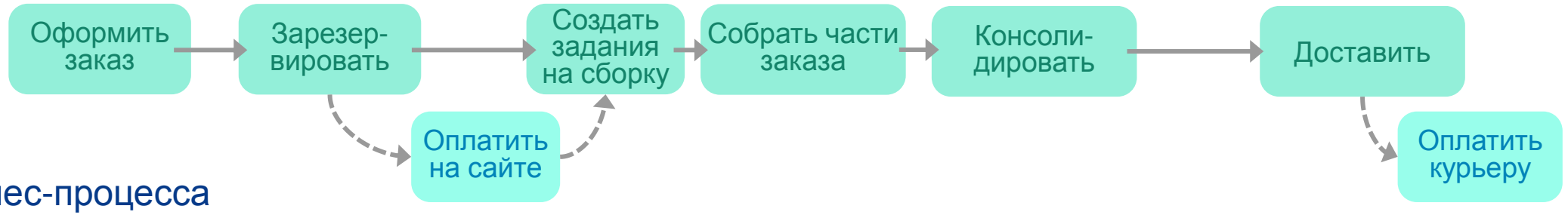
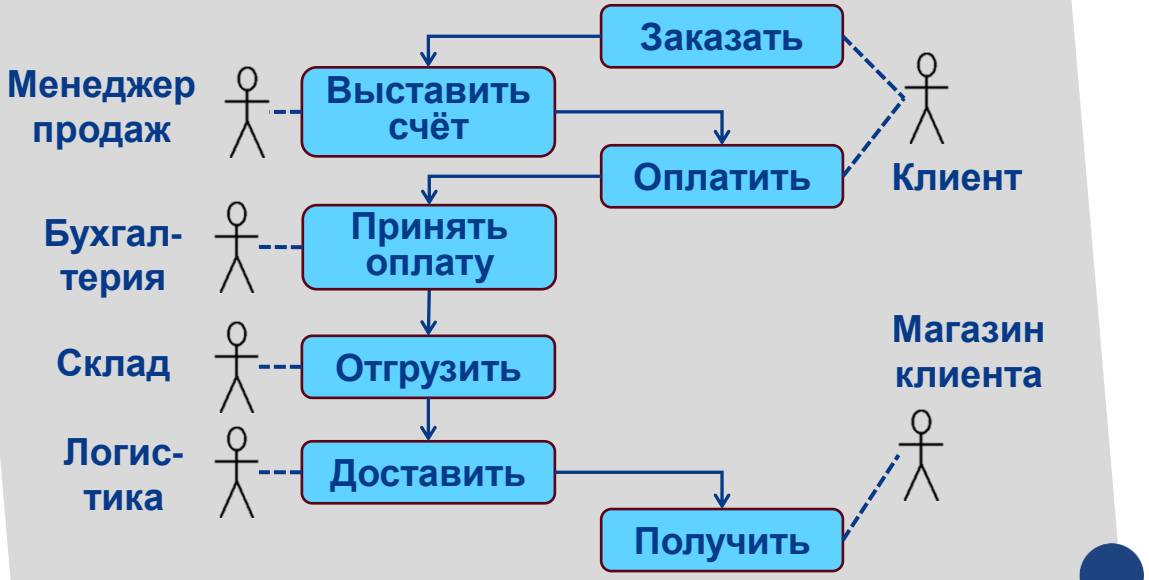


Схема бизнес-процесса

Объекты и их состояния

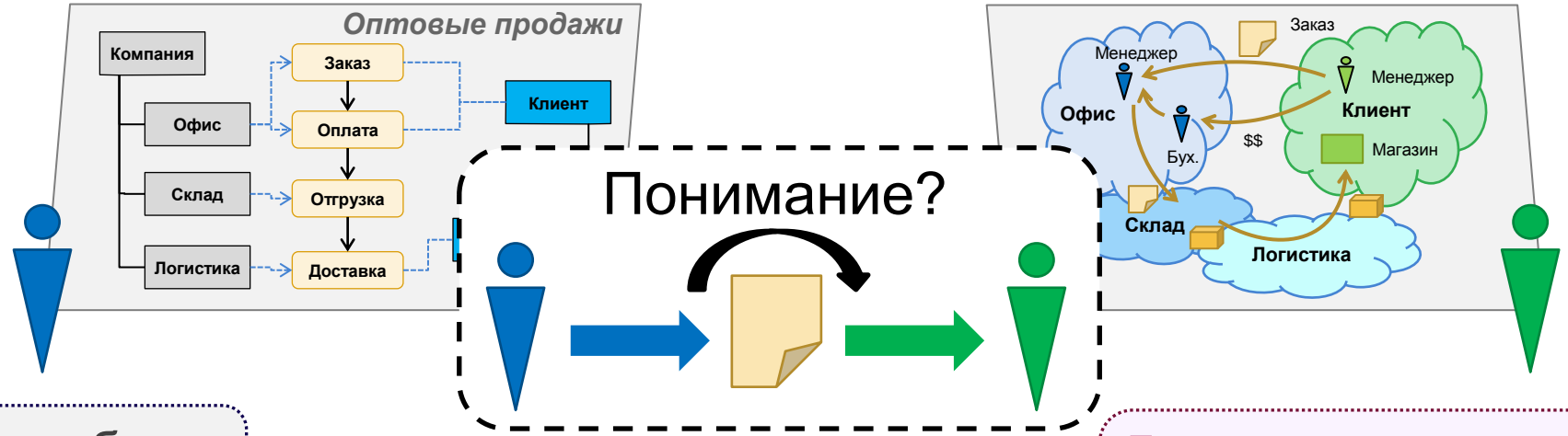
Формализация бизнес-модели

Оптовые продажи магазинам и торговым сетям

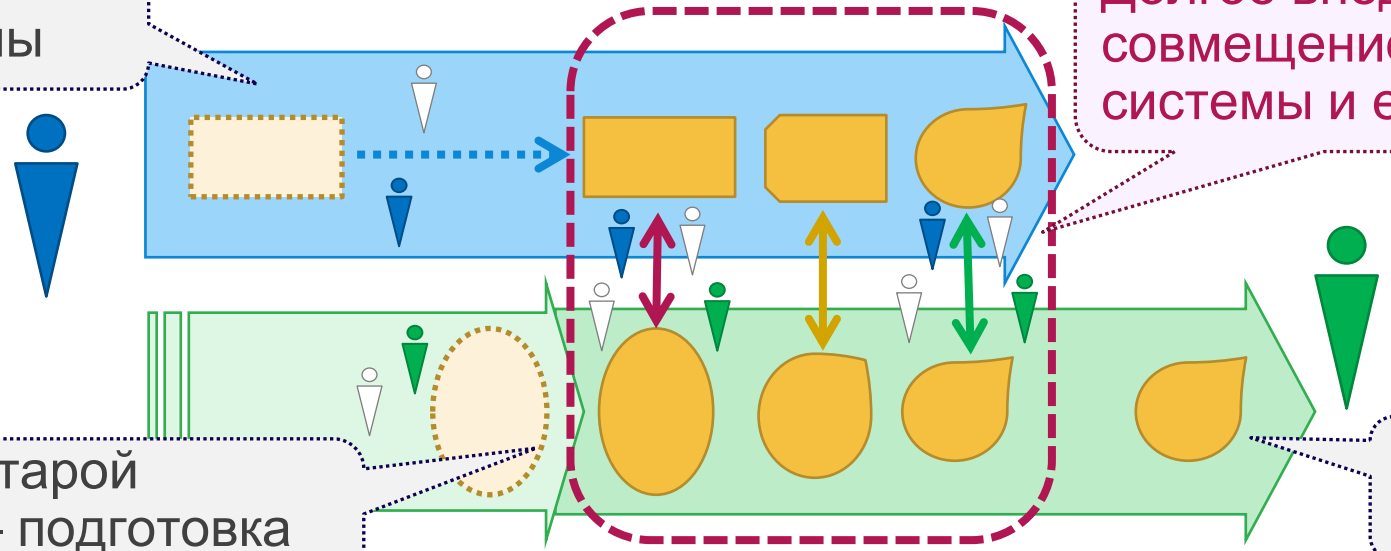


Как осуществляется переход к формальной бизнес-модели
от представлений о повседневной деятельности на интервью?

Непонимание влечёт долгое внедрение



Разработка системы



Долгое внедрение — совмещение контуров системы и её места

Работа в старой системе — подготовка места для новой

Работа по-новому

CUSTIS

Формализация снабжения магазинов

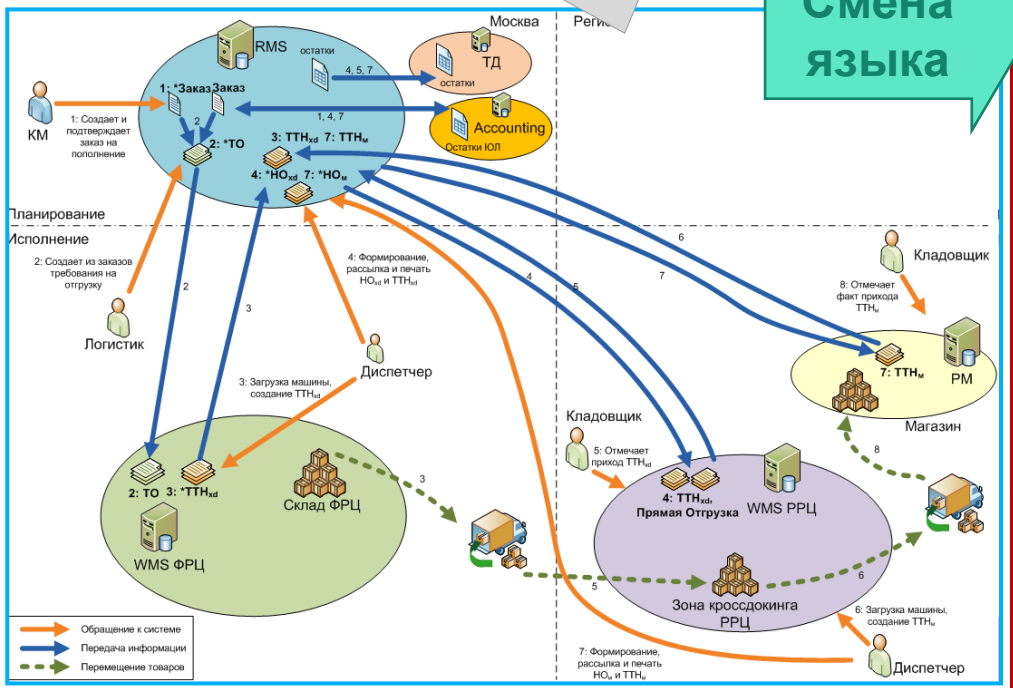
Объекты — Class Diagram

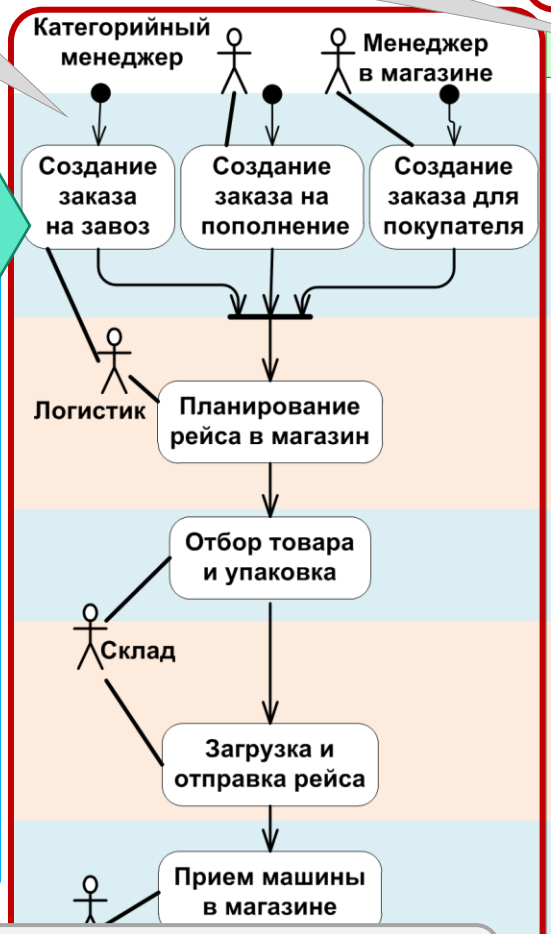
Бизнес-процесс — Activity Diagram

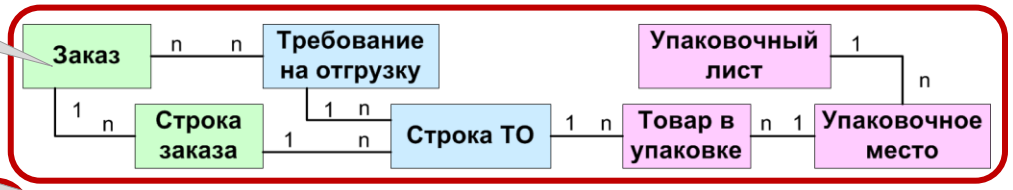
Состояния документов — State Diagram

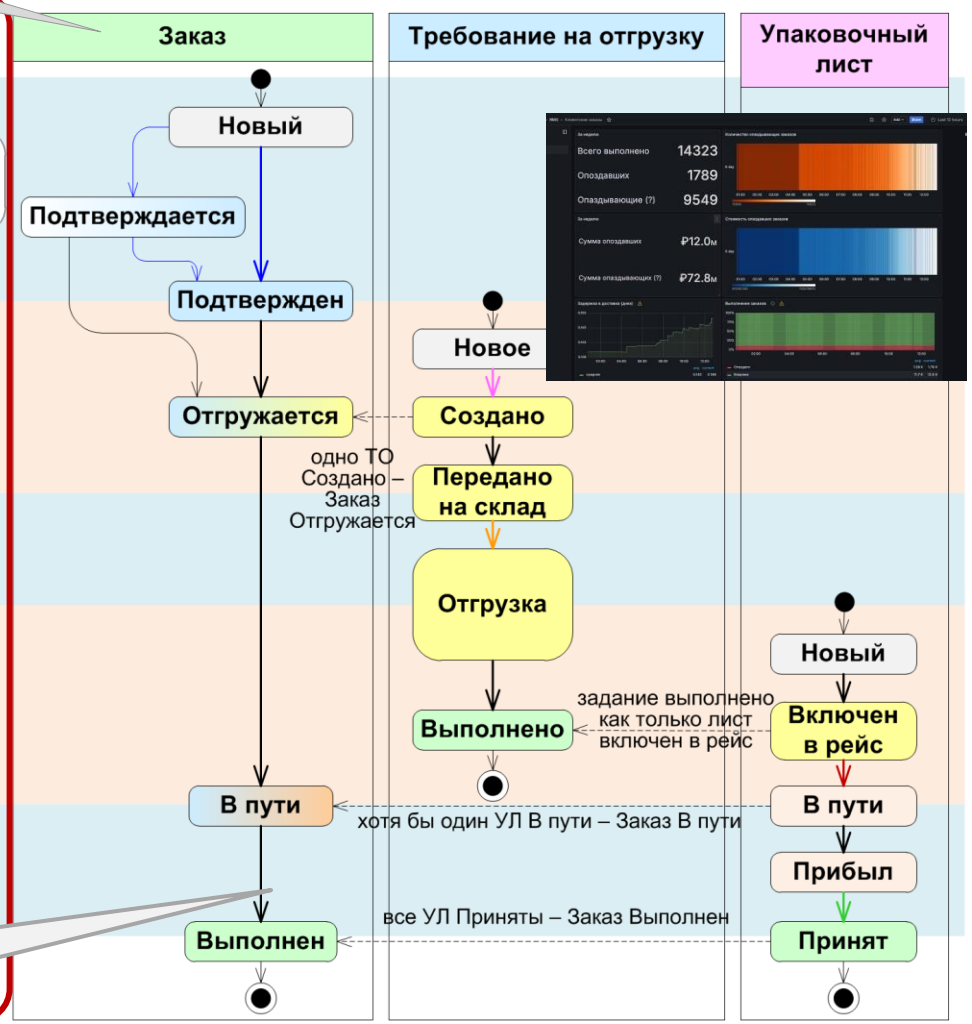
Неформальная схема деятельности и её отражение в существующих системах

Смена языка







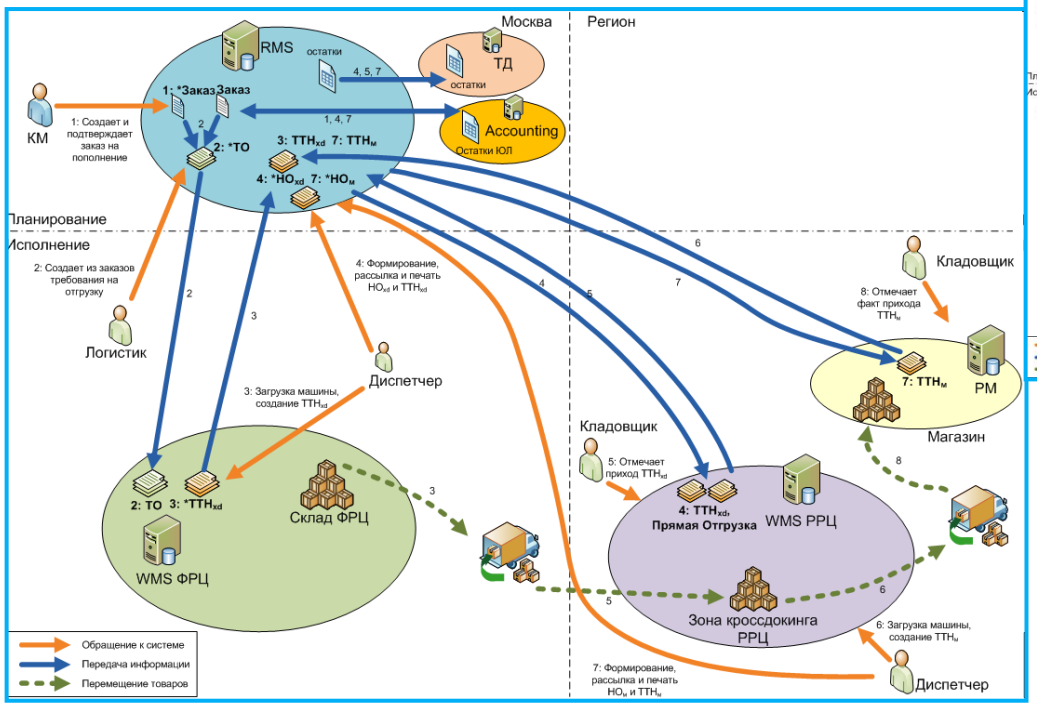


Если workflow документов прозрачно отражает бизнес-процессы, то можно обсуждать их сразу на такой схеме.

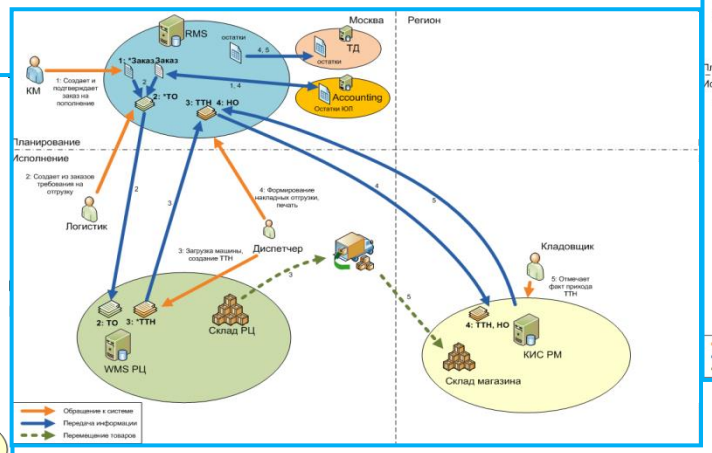
Проверка в неформальной модели

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.

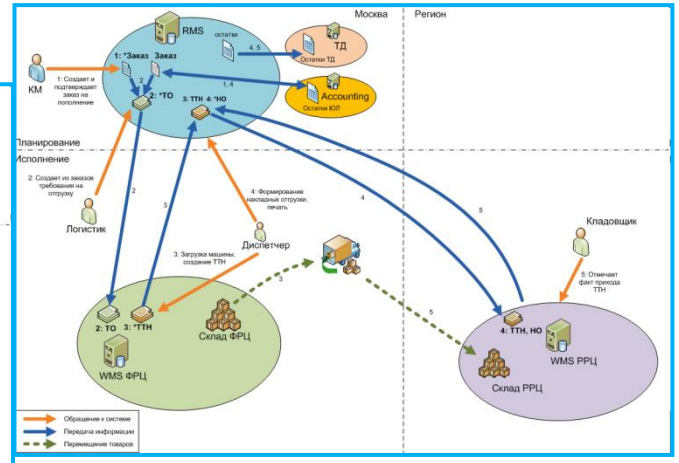
1



2



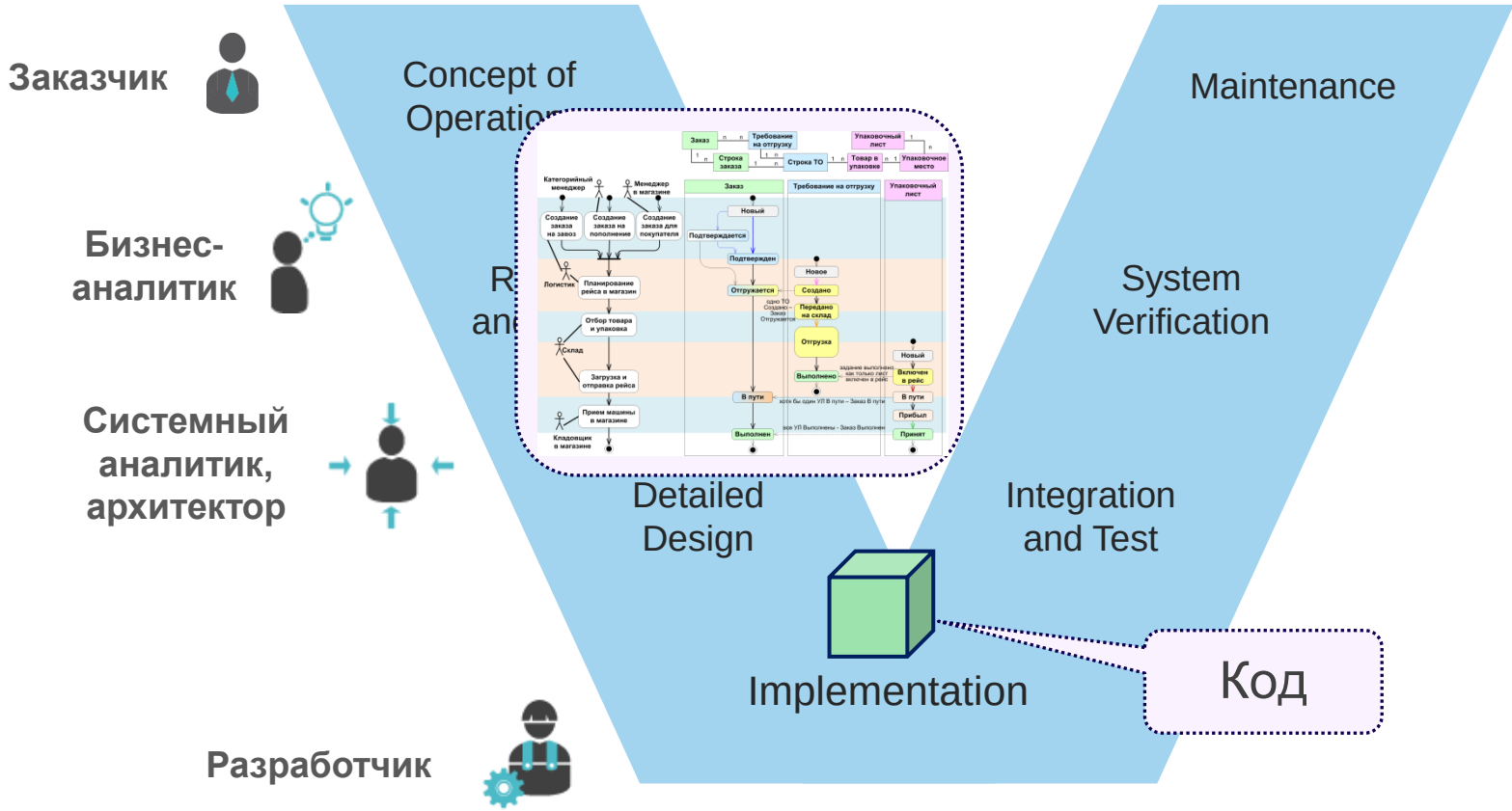
3



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

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.



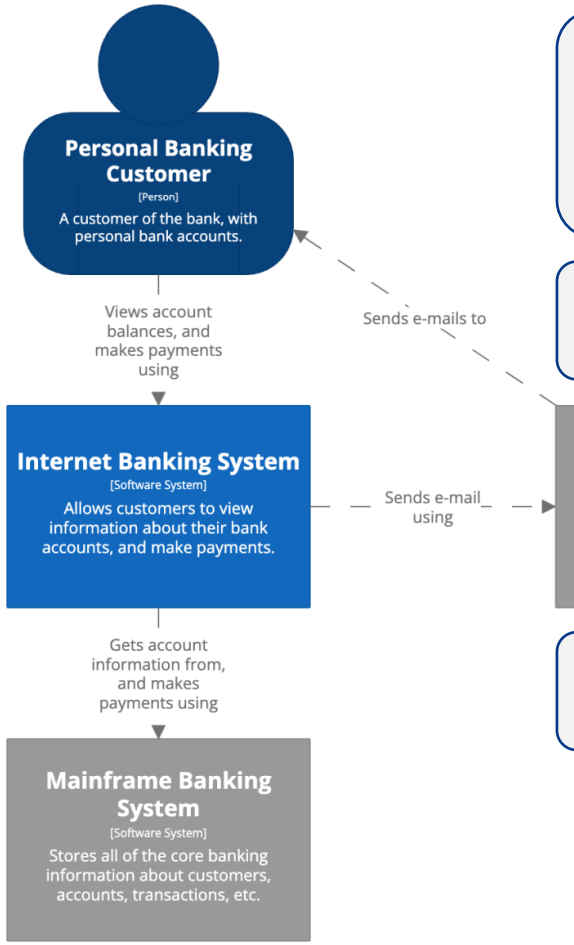
У меня есть много докладов о DDD, последний — «DDD: модели вместо требований 9 лет спустя (ЛАФ-2023)».

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.**

Контекстная диаграмма C4 Model — система в окружении

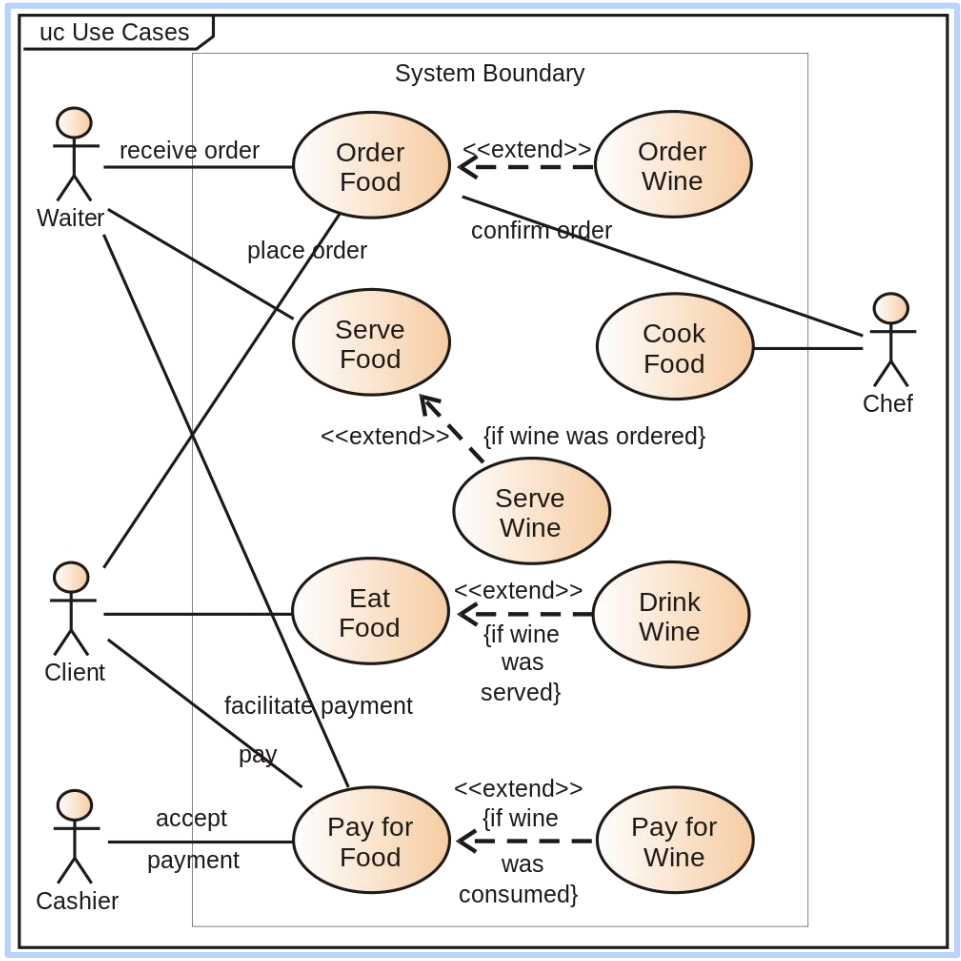


В центре — наша система, вокруг — пользователи и взаимодействующие системы

Стрелки — акты взаимодействия

Множество внешних систем — смена парадигмы на сервисы

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



Представляем пользователя!

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.

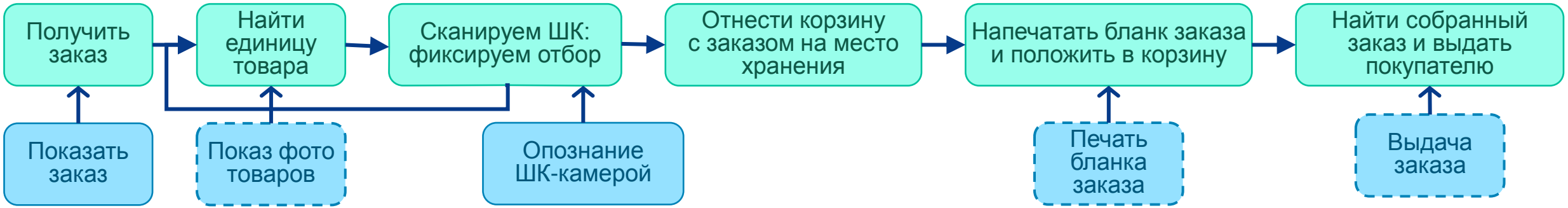
Запрос на доработку. Поддержать на мобильном рабочем месте отбор одежды по заказу в торговом зале магазина.

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.



Что надо знать о бизнесе?

Какие нужны знания о предметной области?

- Нужны **цели проекта** – с помощью impact mapping или в другой форме
- Нужны объекты, с которыми работают пользователи:
  - в виде словаря понятий, типов документов и сущностей в справочниках;
  - в виде полноценной объектной модели;
  - может быть ссылка на отраслевое описание или распространенное приложение.
- Нужно ли описание бизнес-процессов?
  - Обязательно, если при внедрении предполагается их изменение, *as is* и *to be;*
  - если автоматизируются имеющиеся процессы, можно без их описания, ограничиться ролями и функциями пользователей, которые описывать через use case или списком.
- Альтернативные способы описания бизнеса:
  - user story и story mapping,
  - event storming,
  - workflow документов с ролями пользователей.

Event Storming и бизнес-процессы

BPM — моделирование бизнес-процессов:

- представляем 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, через систему целей или через учёт.

Use case или user story?

- Оба нужны, если мы хотим описать работу пользователя для реализации интерфейсов без контакта с заказчиком до демо.
- Оба хорошо описываются на основе бизнес-процессов, на их основе можно делать тест-кейсы.
- User story:
  - ☺ фиксируют цели пользователей, а не только взаимодействие с системой;
  - ☺ хорошая единица работы: можно менять порядок для реализации по важности;
  - ☹ сложные случаи могут потребовать существенных переделок;
  - ☹ 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.

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.

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.

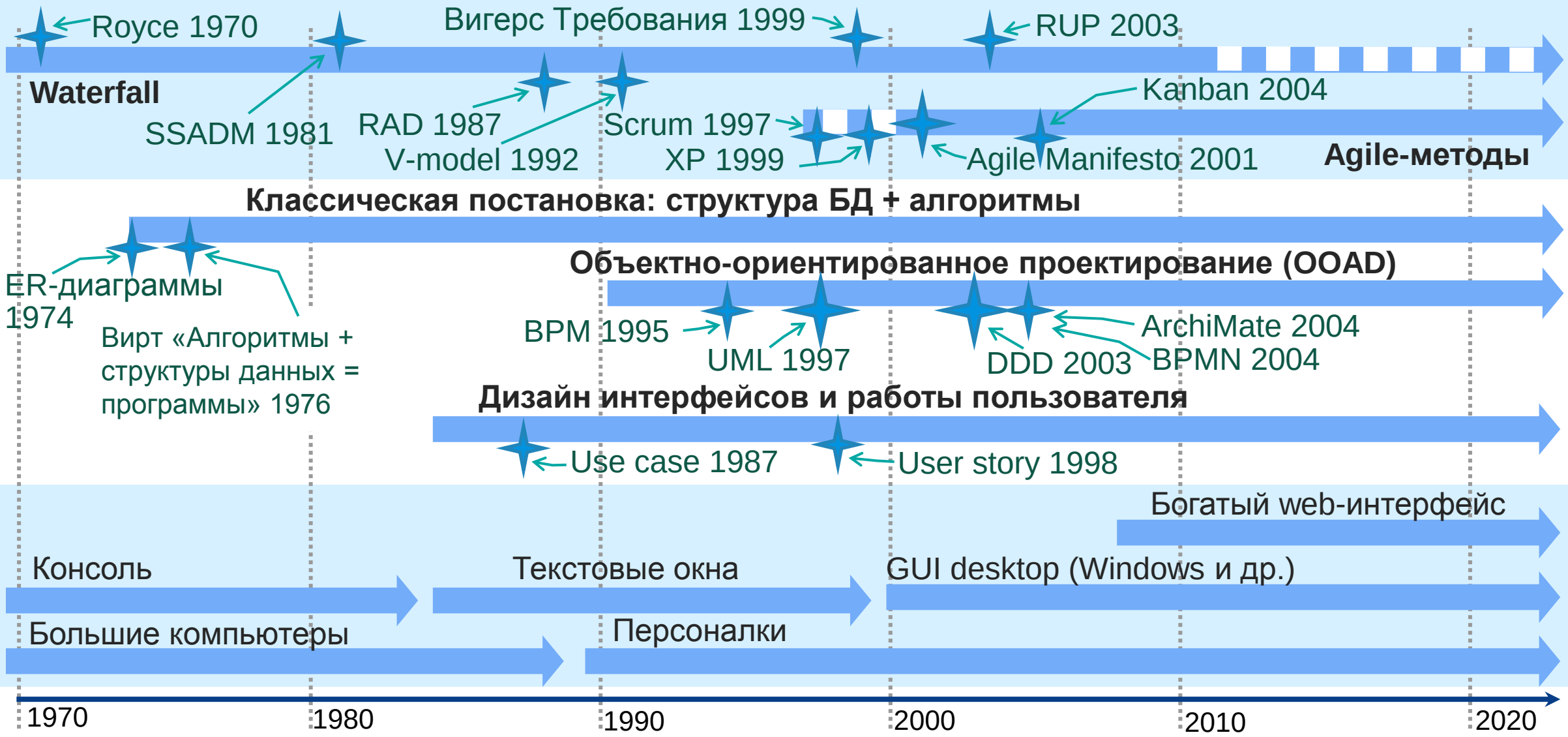
DDD: согласованная модель приложения.

- 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.

История развития проектирования



Итоги

ARCHDAYS | CUSTIS 2024

- Для успешного создания софта надо понимать архитектуру предприятия.
- 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, хотя 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.

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!