CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

DDD: модели вместо требований, 9 лет спустя

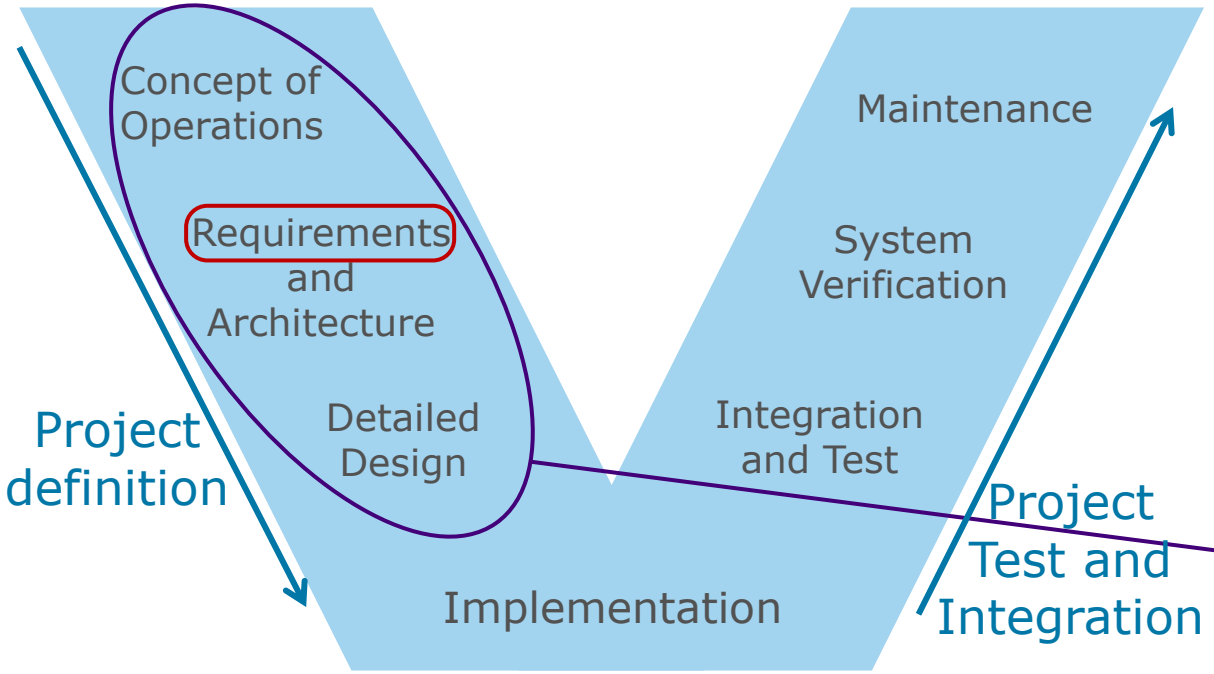
Максим Цепков

Главный архитектор решений CUSTIS
Навигатор в мире Agile, бирюзовых организаций
и спиральной динамики

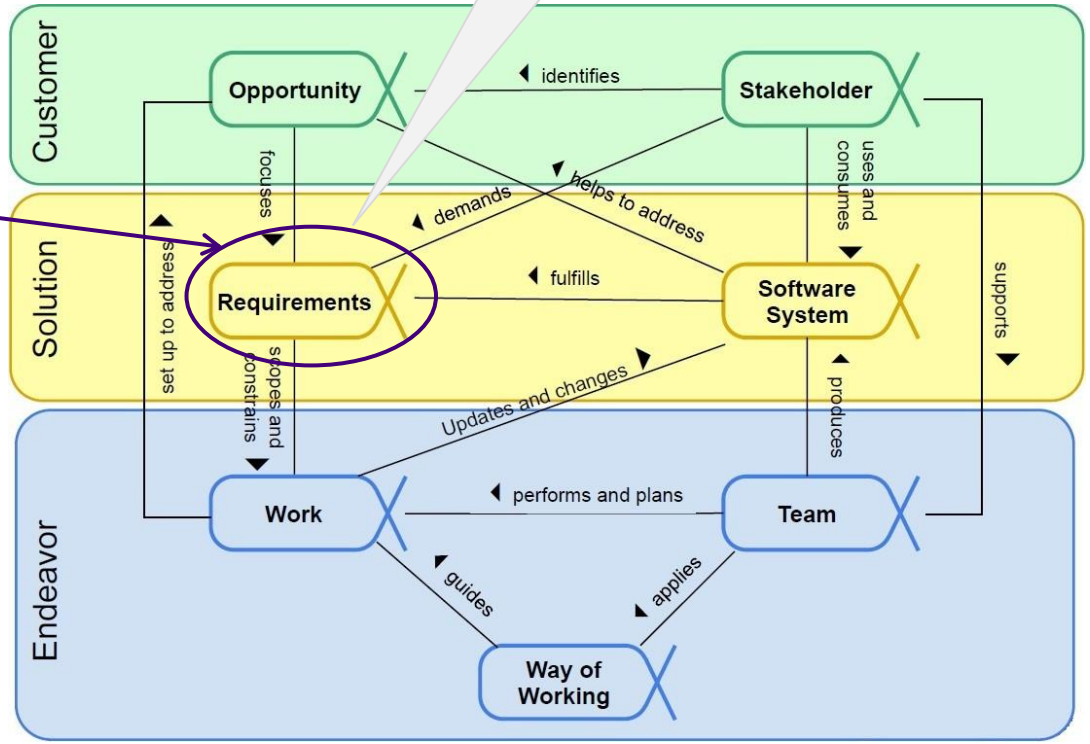
UML2.ru

Летний аналитический фестиваль
Кострома, 09–11 июня 2023

Что такое требования



В OMG Essence это назвали «**требования**», изменив смысл термина



V-Model (Wikipedia)

От требования к модели

CUSTIS

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

При резервировании товара по заказу надо контролировать доступный остаток

- Таблица текущих остатков хранит свободное и зарезервированное количество

Если покупатель — юридическое лицо, надо выписать счет-фактуру; они нумеруются последовательно с начала года

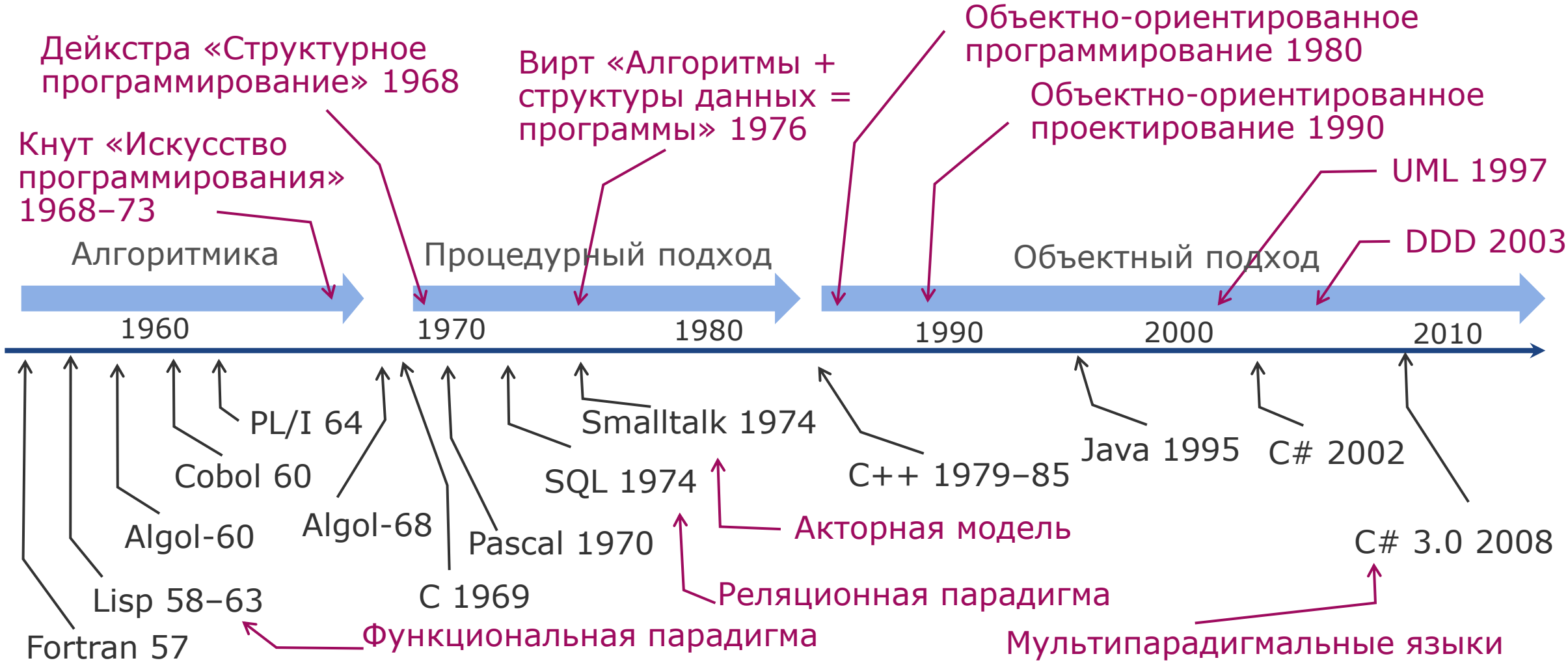
- В таблице покупателей храним признак юридического лица
- В таблице заказов храним номер счета-фактуры, отдельной сущности не создаем
- Номер присваиваем при переходе заказа в состояние «Отгружается»

Способ доставки зависит от адреса: в Москве — наши курьеры, по стране — почта

- Надо спроектировать, как по адресу определять возможность доставки курьером

Онтологическая разница: требования — тексты, нарратив, модель — описание устройства

Развитие программирования и проектирования



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

Процедурный и объектный подход: различие моделей

CUSTIS

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 и действия выполняем

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

Доставка самовывозом и курьером

- Особенности размазаны по всем алгоритмам через условия
- Делаем подтипы, инкапсулируя особенности поведения

Процедурный и объектный подход: проектирование

CUSTIS

Процедурный подход

- Структура БД
- Интерфейсы
- Алгоритмы обработки
- Процедуры API backend и API RPC (если необходимы)

Объектный подход

- Типы и статусы объектов
- Методы бизнес-логики
- Интерфейсы в стиле **naked object** — витрины объектов и методы
- REST API

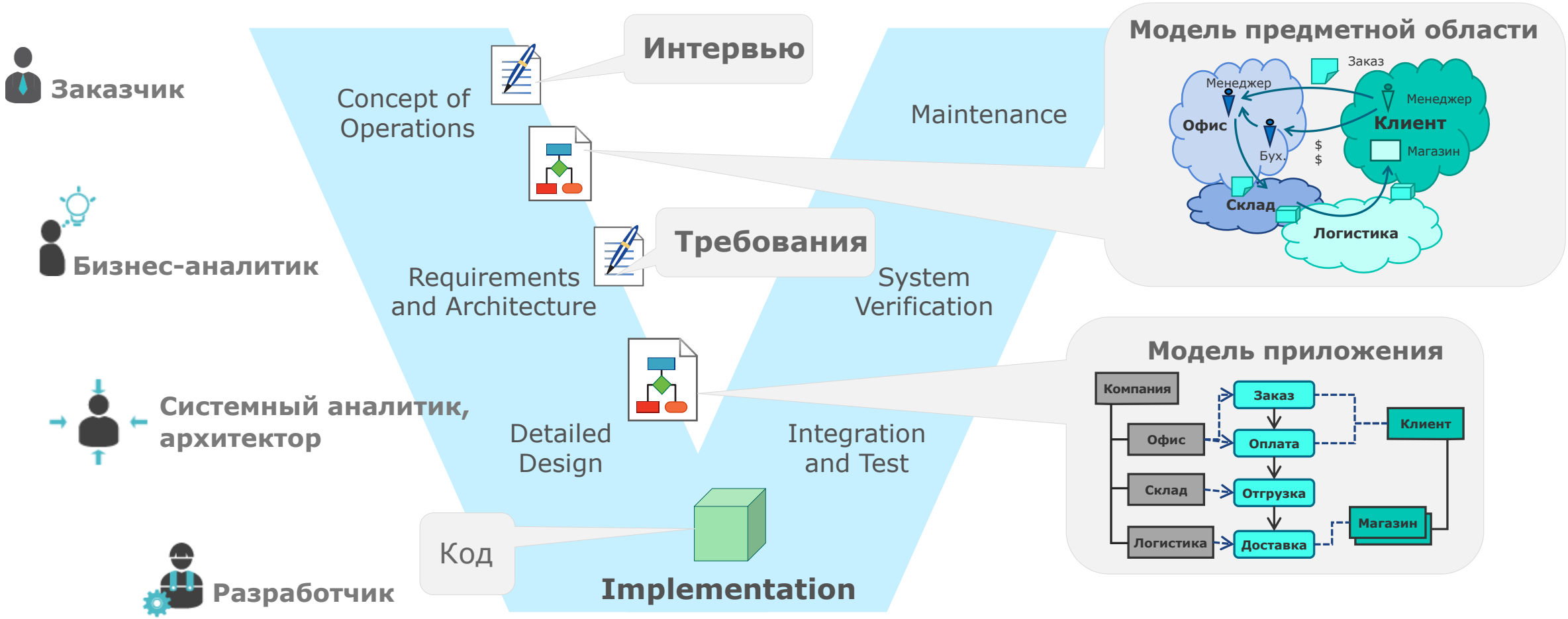
Бизнес-логика

- Распределена по разным процедурам

- Инкапуслирована в объектах

Модели в классических постановках

CUSTIS



Проблема: каждый уровень отдельно — сложно изменять и поддерживать соответствие

Пример — снабжение магазинов

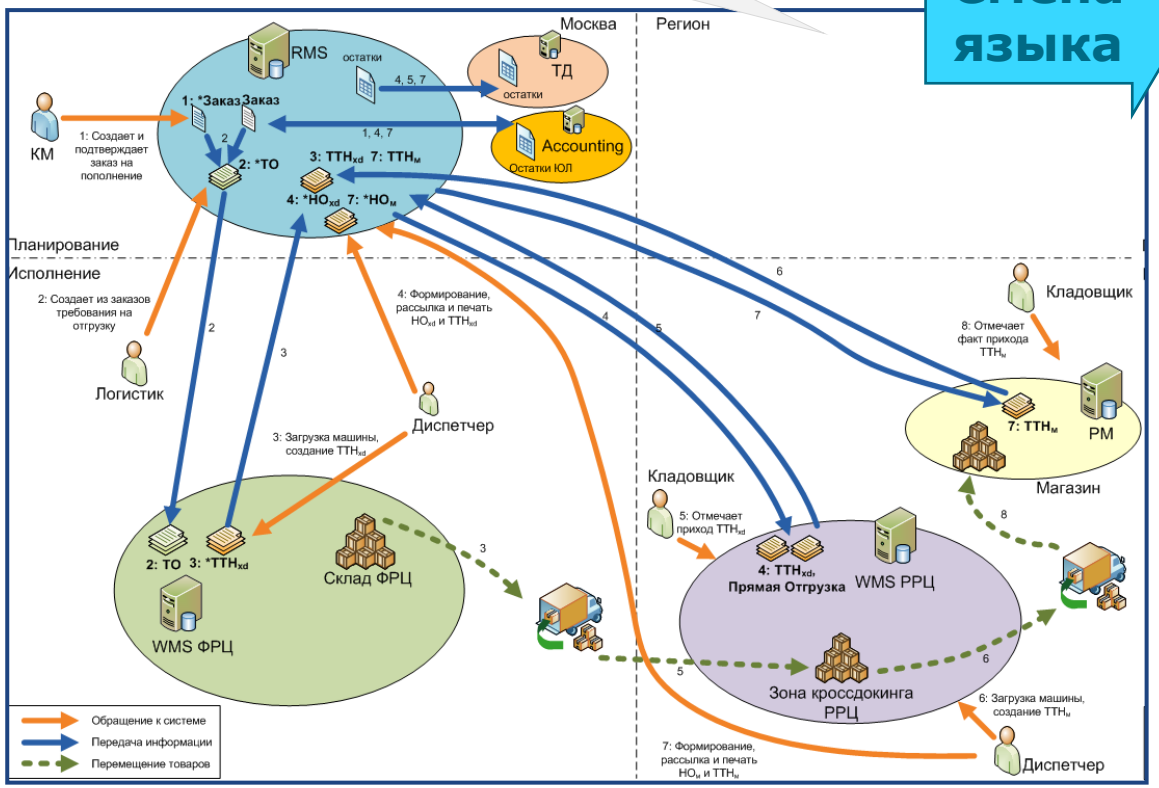
CUSTIS

Неформальная схема деятельности и ее отражение в существующих системах

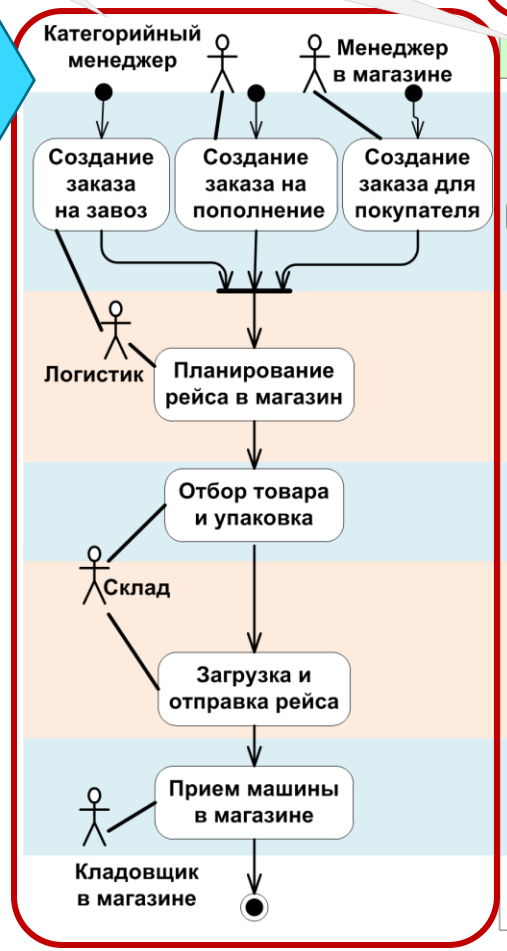
Бизнес-процесс — activity diagram

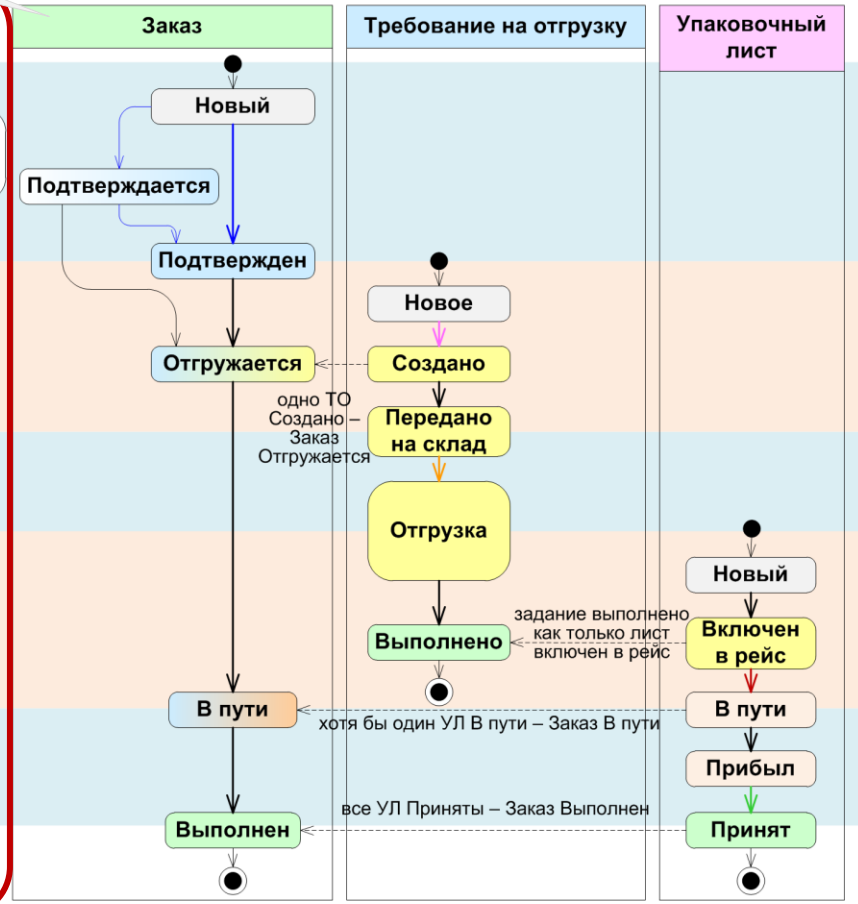
Объекты — class diagram

Состояния документов — state diagram

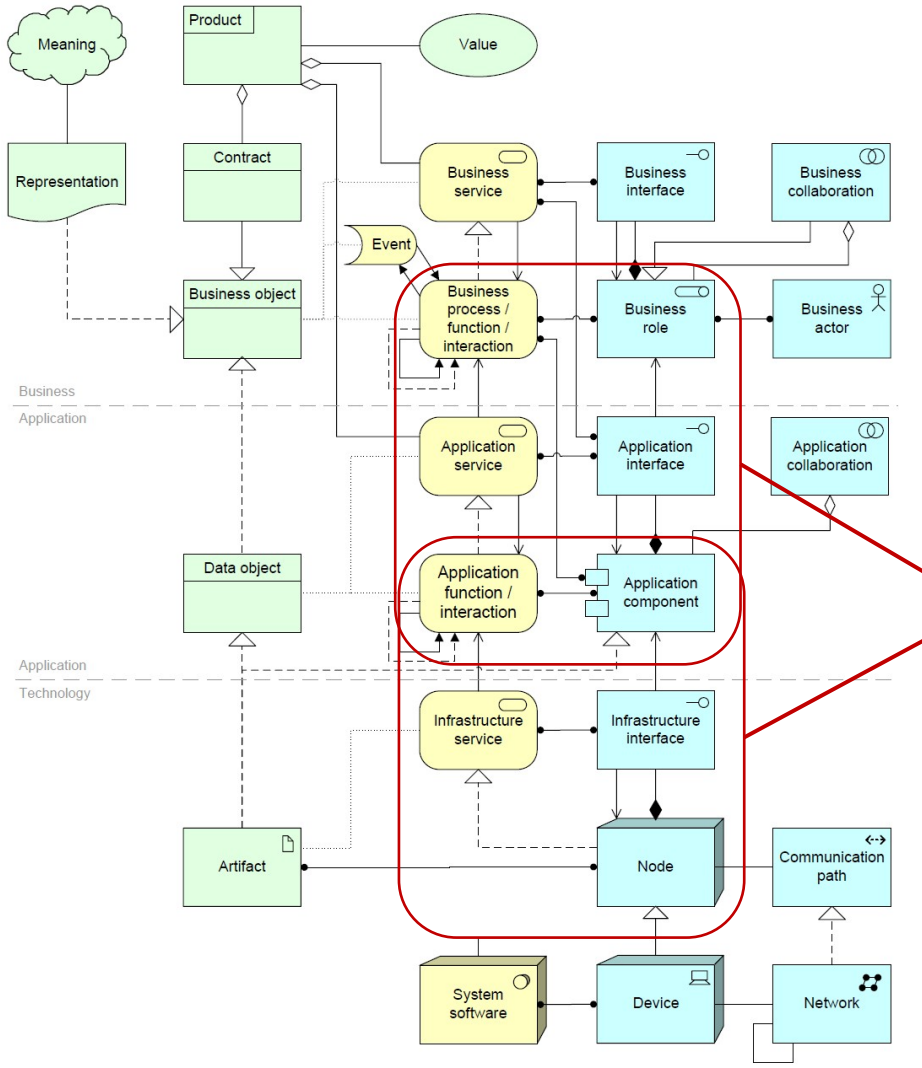


Смена языка

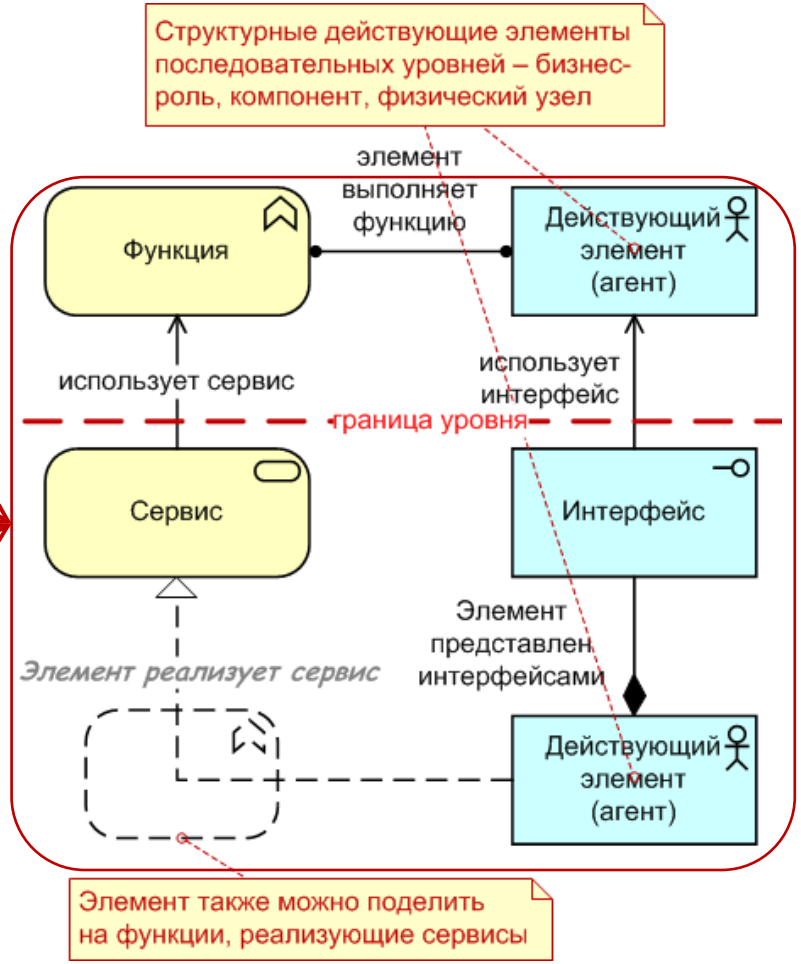




Изоляция уровней в Archimate



Типовое сопряжение уровней



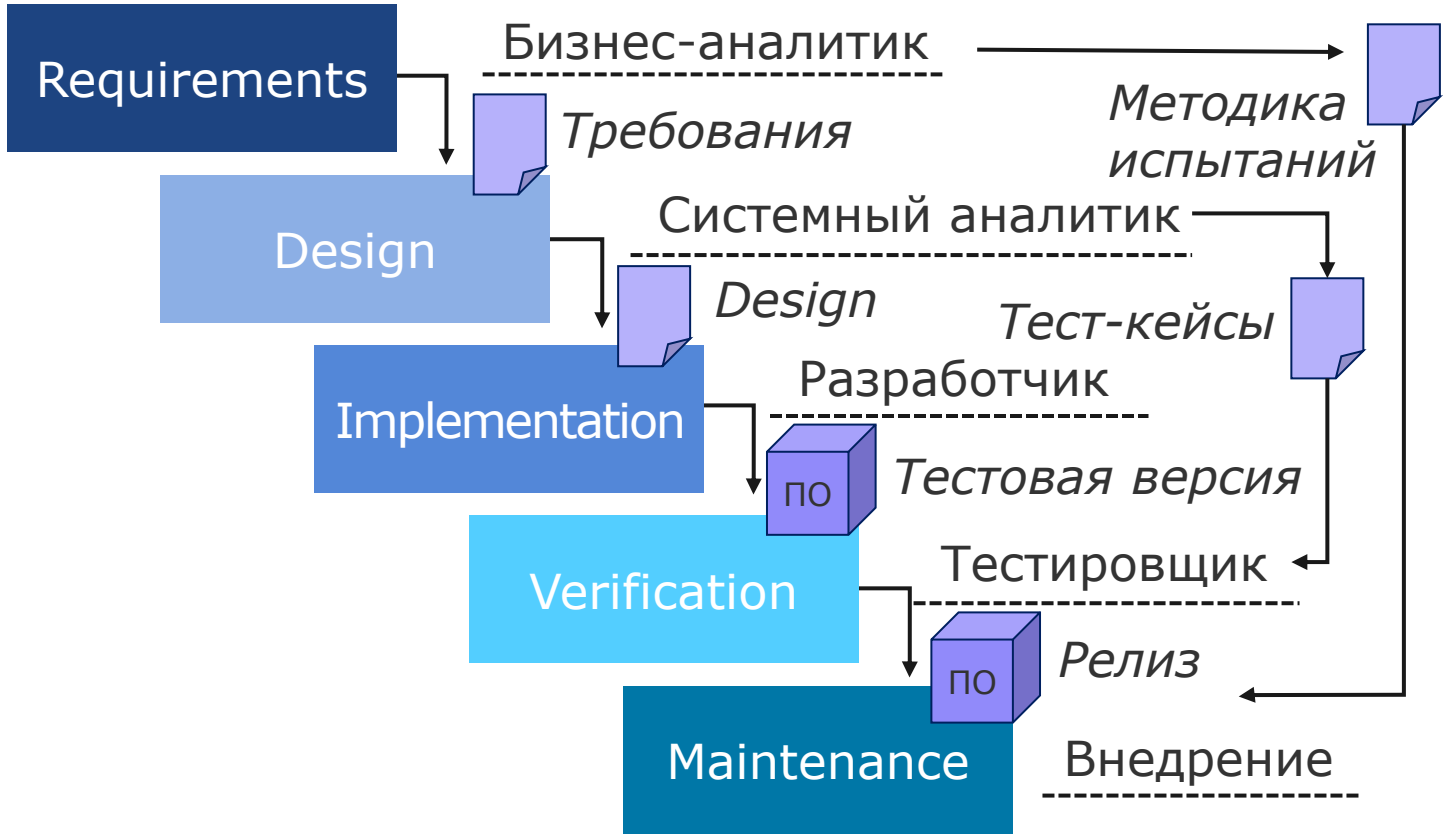
Метамодель Archimate 1.0

Решение Archimate — изоляция уровней.

В схеме предполагается, что ИТ — это инфраструктура, развивающаяся опережающими темпами.

В современных условиях это недостижимо, т. к. именно ИТ ставит ограничения бизнесу.

Водопад — специализация по фазам проекта



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

Каждой фазе — своя роль.

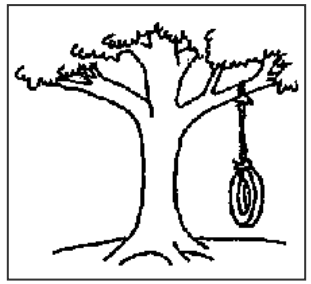
Роли выполняются разными людьми или командами.

Передача работы формализована **через артефакты**.

Каждая передача искажает смысл

CUSTIS

Что хотел заказчик



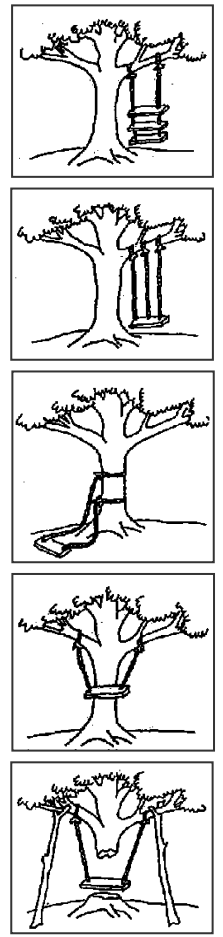
Requirements — Бизнес-аналитик

Design — Системный аналитик

Implementation — Разработчик

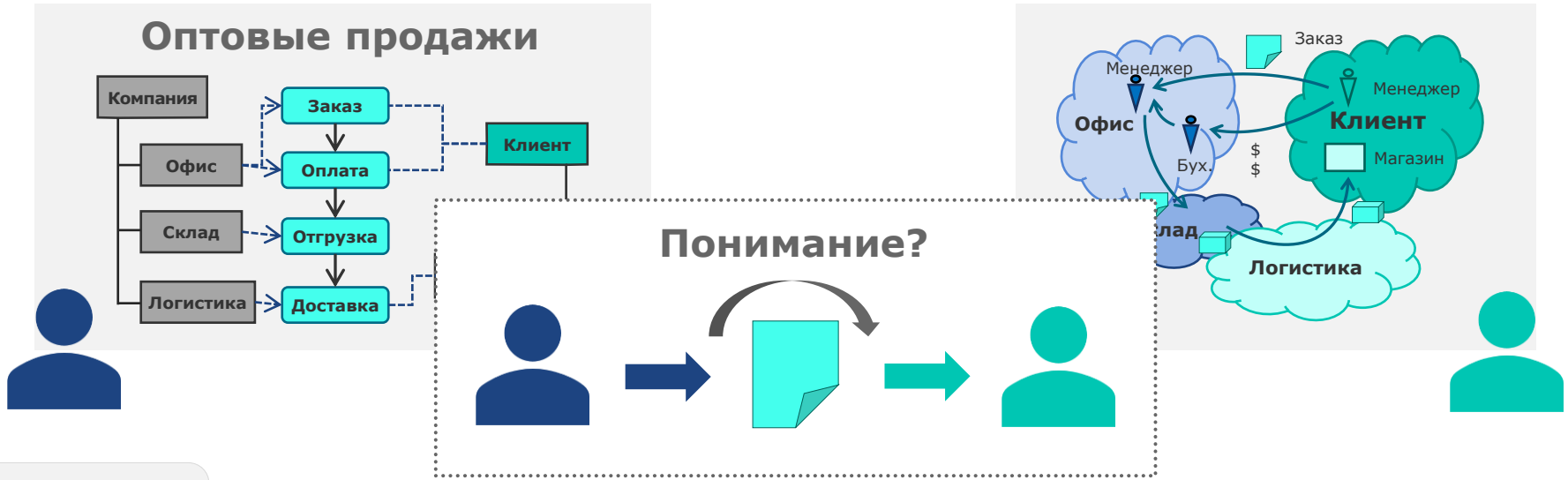
Verification — Тестировщик

Maintenance — Внедрение

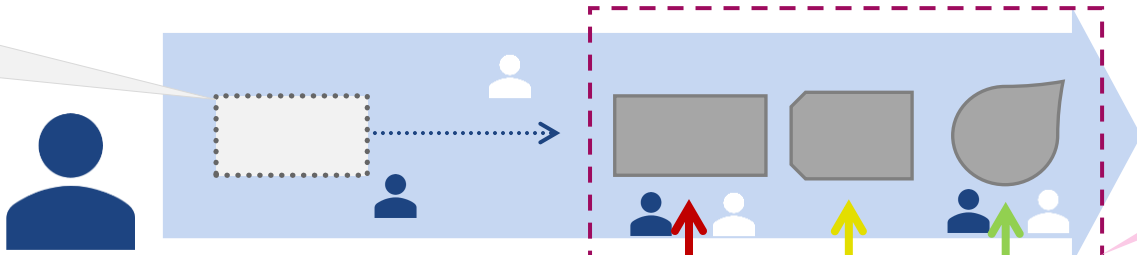


Старый известный образ

Проблема двух моделей

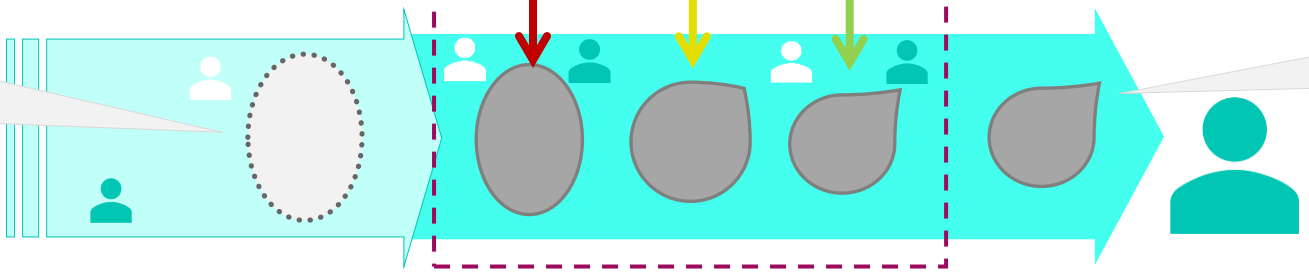


Разработка системы



Долгое внедрение — совмещение контуров системы и ее места

Работа в старой системе — подготовка места для новой



Работа по-новому

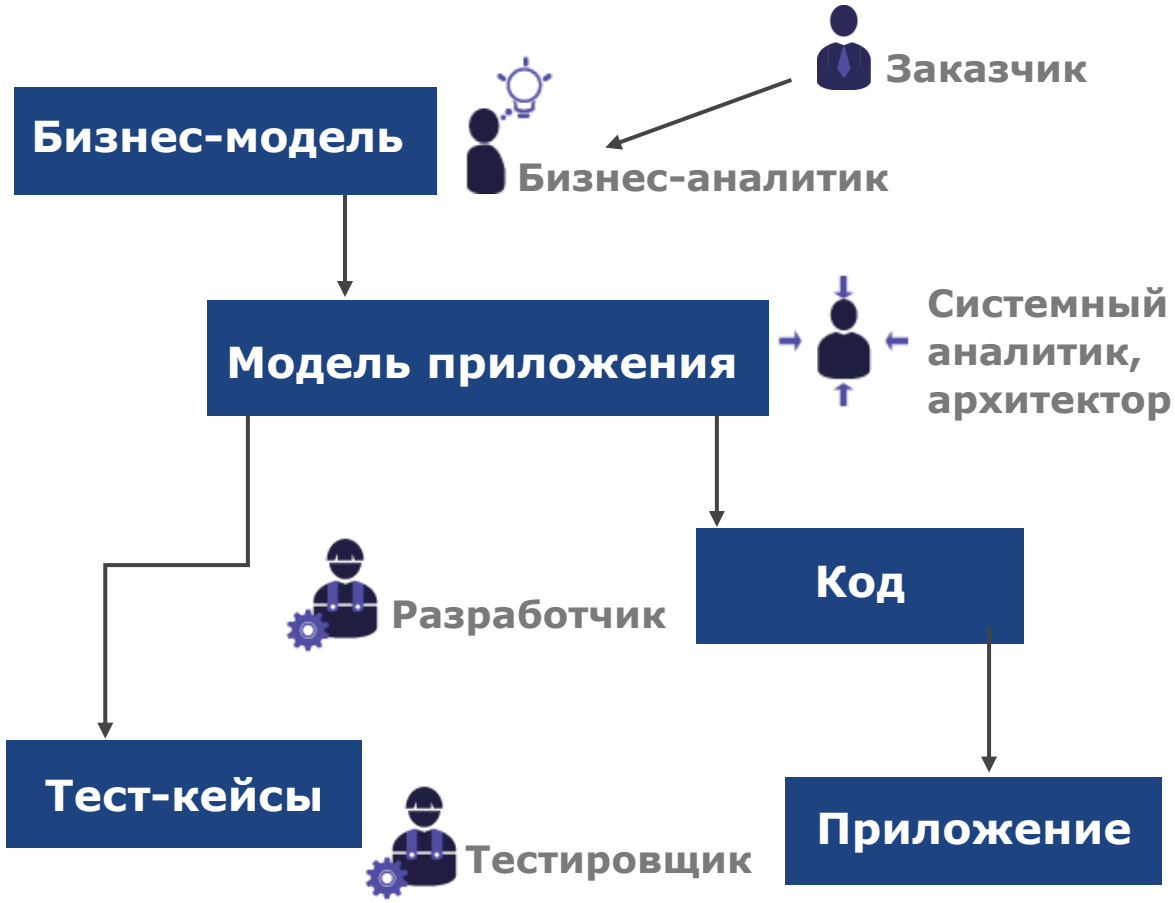
CUSTIS

DDD — единая модель и единый язык

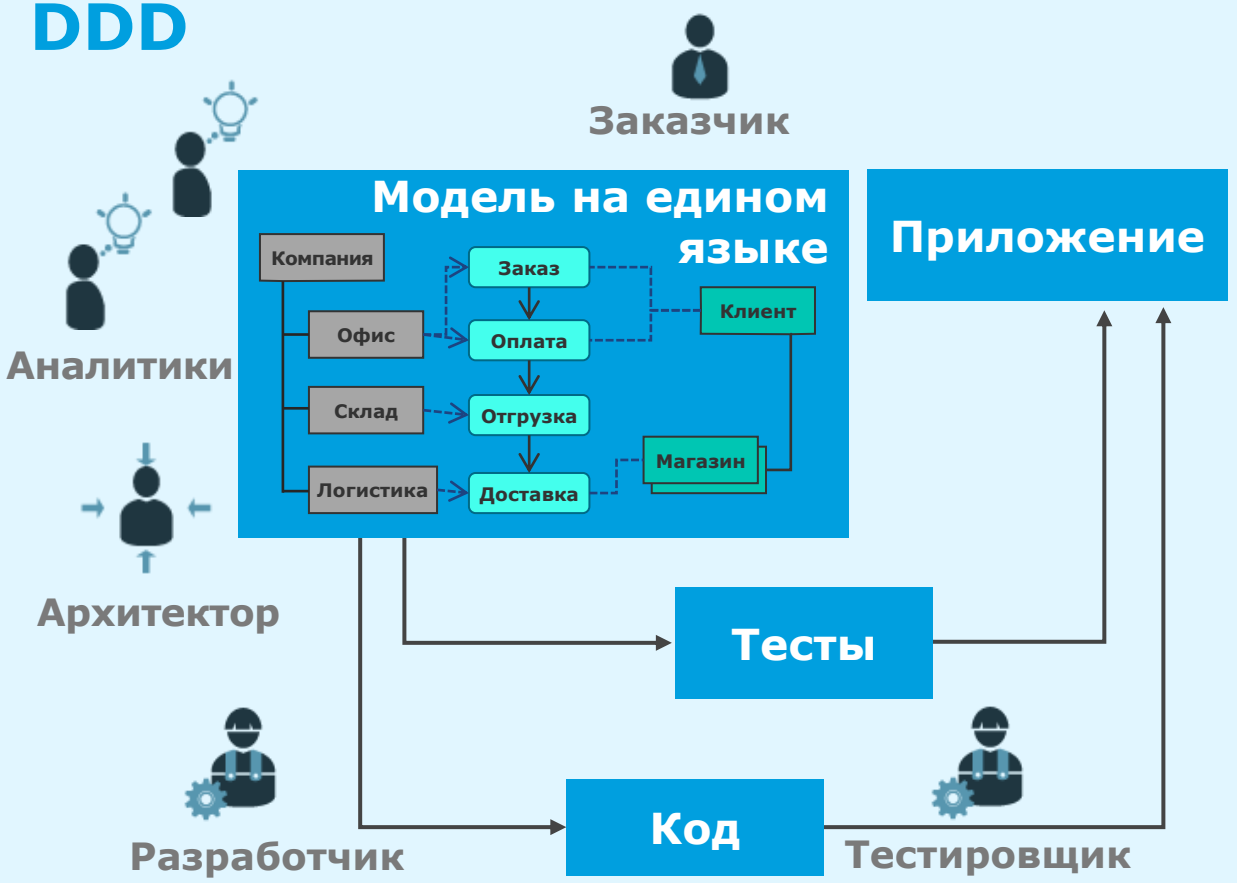
DDD: единый язык и единая модель приложения

CUSTIS

Классика



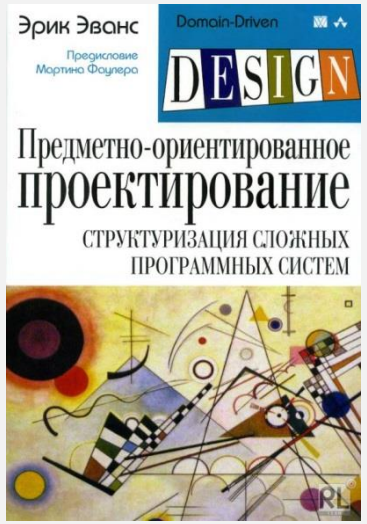
DDD



Единый язык создает общее пространство общения

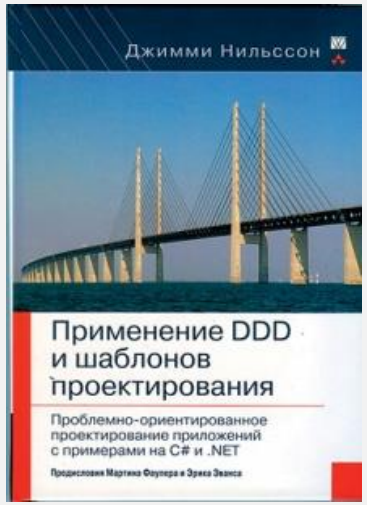
DDD: источники

CUSTIS



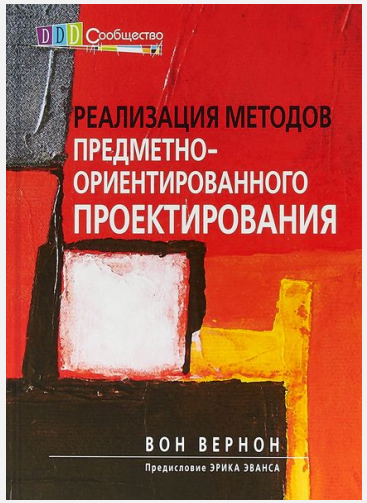
Концептуальная книга Эрика Эванса

- на английском — в 2003 г.
- на русском — только в 2010 г.



Практическая книга Джимми Нильссона

- на английском — в 2006 г.
- на русском — в 2007 г. (почти сразу!)



Обновление от Вона Вернона

- на английском — в 2013 г.
- на русском — в 2016 г.

Единый язык (ubiquitous language)

CUSTIS

- Построен на основе терминов предметной области
- Его понимают и ИТ-специалисты, и эксперты бизнеса
- На нем описана модель ИТ-системы и ее место в бизнес-процессах

Понятия единого языка: **клиент, накладная, платеж, долг** — из предметной области

Единая модель

CUSTIS

- Аналитик собирает требования и строит модель — сначала предметной области, затем системы
- Артефакты модели описывают систему и ее использование в бизнес-процессах предприятия
- Разработчик реализует модель
- Артефакты модели можно проследить в коде

Модель предметной области становится **моделью системы**

Плюсы единой модели

CUSTIS

- Верификация постановок бизнес-специалистами
- Общее понимание требований на стороне бизнеса
- Обсуждение модели бизнесом и ИТ, поиск баланса в сложных вопросах
- Перенос моделей из других предметных областей
- Бизнес представляет потенциальные возможности системы и сложность различных доработок
- На этапе эксплуатации — эффективное общение бизнес-пользователей и ИТ без квалифицированных переводчиков-аналитиков

Не забывайте про бизнес-уровень

CUSTIS

✕ Работая с моделью системы, аналитики часто забывают о реальных процессах, и модель получается неработоспособной

Примеры

Сборка интернет-заказа из торгового зала магазина:

- Надо визуально найти нужный товар среди развешанных на витрине; на складе такой проблемы нет, там товары легко идентифицируются по кодам
- Собранный товар надо отложить и пометить, а для этого напечатать заказ

CUSTIS

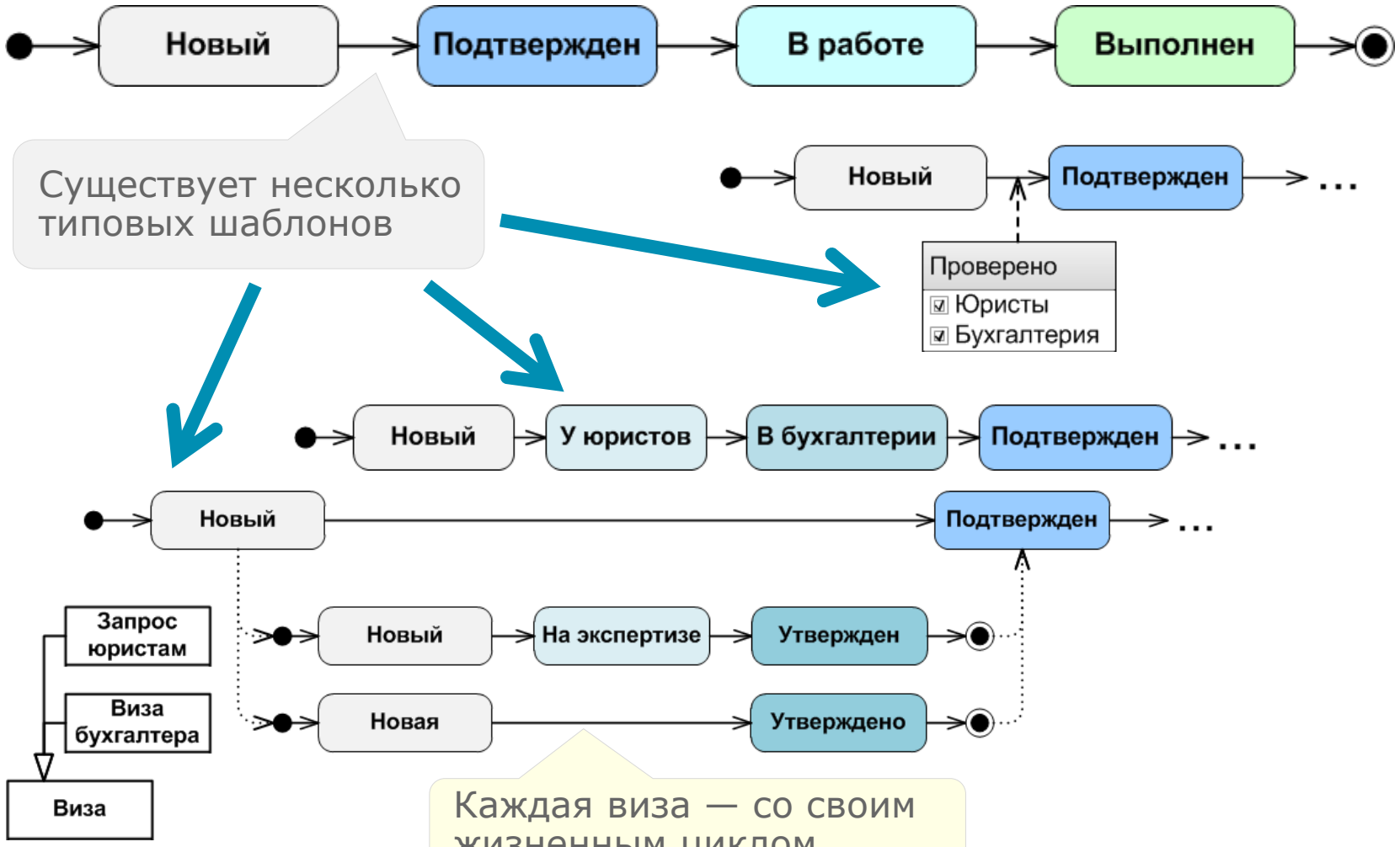
Пример проектирования: виза на документы

Задача

Задача касается конкретного документа

В процессе обработки документа (сделки, договора, контракта) обеспечить проверку и одобрение определенными сотрудниками или отделами

Выбор проектного решения



В чем проблема?

CUSTIS

- Шаблоны обладают разной гибкостью и сложностью
- Для выбора нужно понимать требуемый баланс гибкости и сложности решения

Традиционный подход

- На этапе сбора требований аналитики формулируют задачу для конкретного документа
- Исходя из этого в системе проектируется решение

✕ Выбранное решение отражает текущую ситуацию

Решение надо принимать с учетом потенциального изменения бизнес-процессов

Примеры взаимосвязей виз

CUSTIS

- Проверка сделки казначейством и отделом рисков: не получается выполнять параллельно
- Юристы отозвали одобрение кредита, а служба безопасности на него опиралась: связи между визами не контролируются системой
- Настройку виз для одобрения договоров с недвижимостью передали в ИТ из-за сложности

Решение в рамках DDD

CUSTIS

- Представляем шаблоны, описанные применительно к конкретному документу, показываем разницу
- Бизнес-специалисты оценивают потенциальную потребность в реализации бизнес-процессов

✓ Решение принимается с учетом перспективы

В чем единый язык?

CUSTIS

Для решения модель надо понятно описать бизнесу

- Можно описать обобщенные решения для документов, «состояния» и «визы», и на них ссылаться
- Можно описывать каждый случай отдельно

Термины должны быть понятны заказчику

- Например, «визированием» могут считать одобрение документа, требующее только просмотра, а если требуется дополнительная работа, то это называется «обработкой» или «проверкой»

Общий шаблон надо «перевести» на язык проекта

Результат

CUSTIS

- Мы используем известные шаблоны решений
- Заказчик оценивает вариант решения не только с точки зрения текущих потребностей, но и из предположений о развитии бизнес-процессов
- Проектируя изменения бизнес-процессов, заказчик представляет потенциал гибкости системы и принимает решения с учетом этого

Точки расширения

CUSTIS

Вопрос: Как обеспечить легкое развитие системы при изменениях правил проверки и одобрения документа?

Ответ: Для этого нужны точки расширения

Стратегии — именованные алгоритмы обработки, включаемые в определенных точках

- Проверки или обработка при переходе в состояние
- Алгоритм определения состояния документа по визам

Стратегии дополняем спецификациями — декларативными описаниями условий, применяемых для выбора стратегий по параметрам документа

- Декларативное описание графа перехода документа и набора виз, связанных с переходами

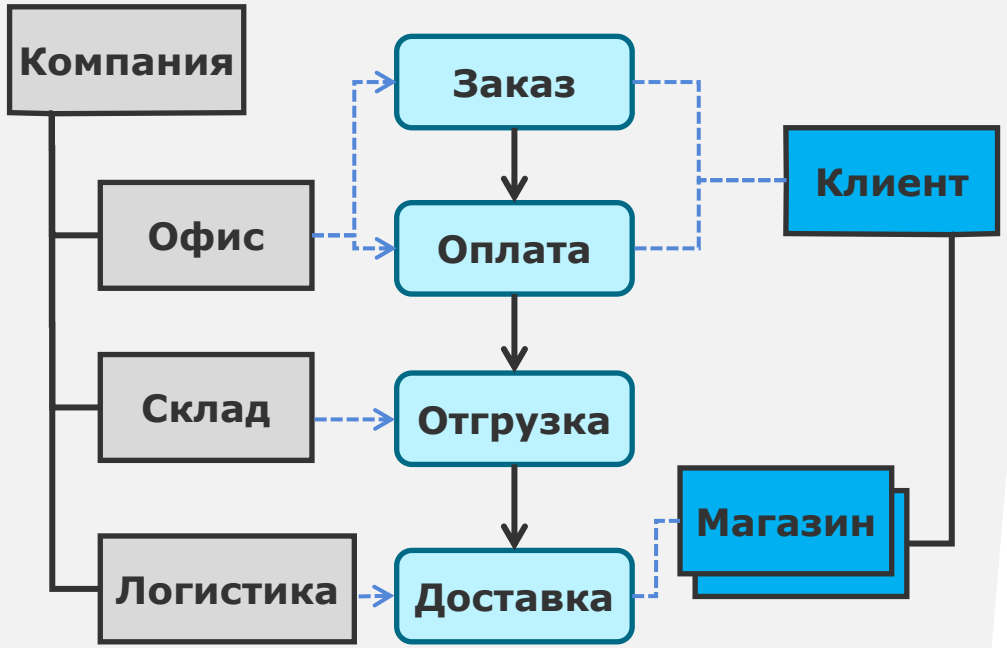
CUSTIS

Влияние модели на деятельность

Два представления о реальности

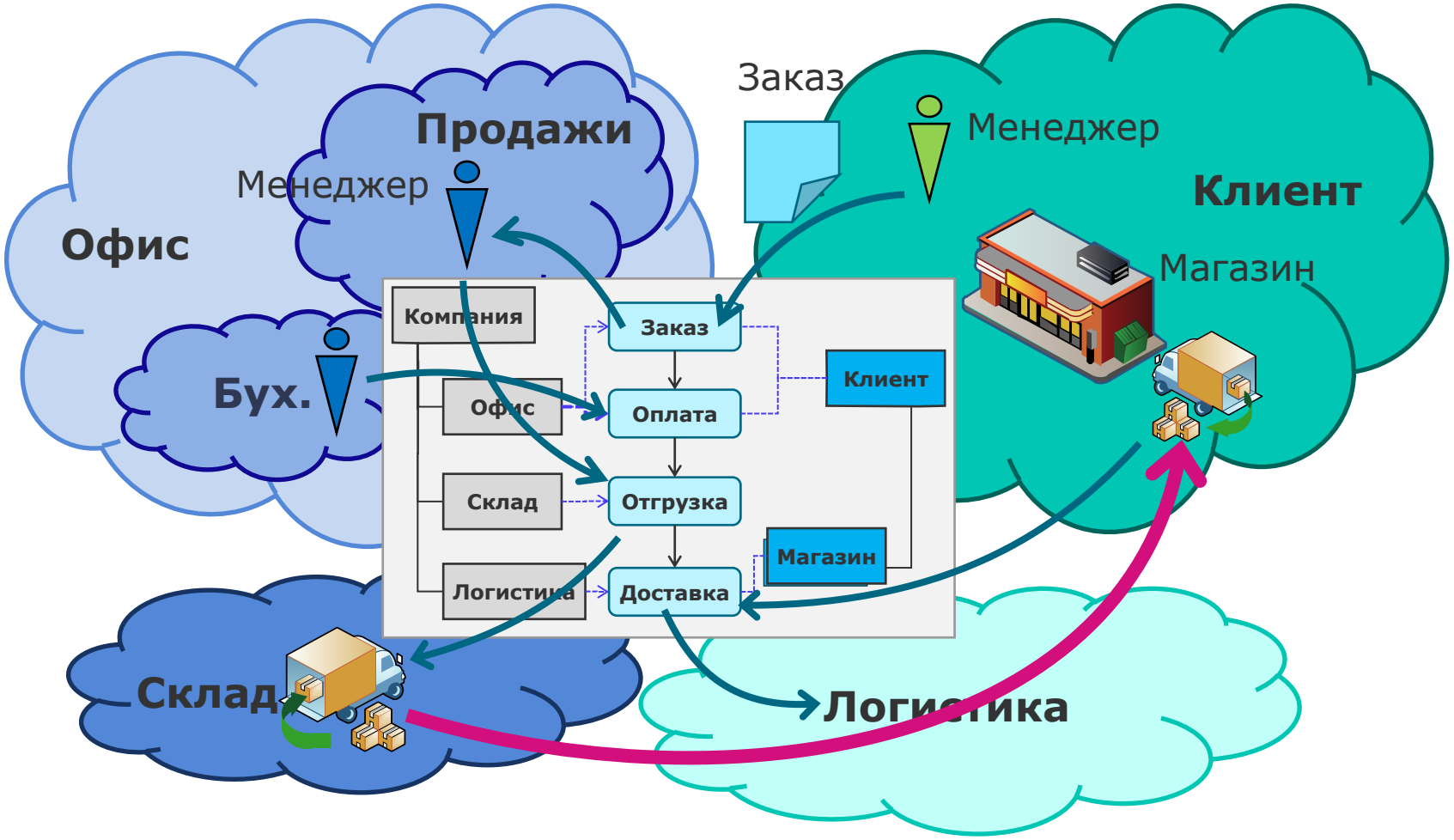
CUSTIS

Оптовые продажи магазинам и торговым сетям



Система дополняет ИТ-ландшафт

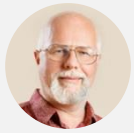
Система становится основой



Три категории постановок

CUSTIS

1. Разработка новой системы поверх фрагментарной и малой автоматизации:
   - выясняем модель бизнес-процессов
   - создаем модель системы
   - работаем над ее встройкой в процессы, изменяя их
2. Доработка существующей системы: проектируем изменения модели существующей системы и ее 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 расчете **на первую ситуацию.** Сейчас мы имеем дело **со второй и третьей.**

Пример: старая модель проросла в требованиях

CUSTIS

При разработке системы оптовых продаж было **требование**:
обеспечить сбор товара под заказ клиента с разных складов

- Весьма трудоемко в реализации и усложняет систему
- Требует организации бизнес-процесса перевозки — его не было

Выяснилось, что собирать с разных физических складов не планировали, требование возникло из разделения товара по логическим складам в старой системе

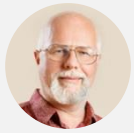
В новой системе для логического деления придуман другой механизм — квоты

Пример: додуманная по решению цель оказалась неверной

CUSTIS

Запрос от заказчика: сделайте механизм заявок, которые бы собирали товар с разных квот и складов, в том числе — сразу после его появления в доступе

- Интерпретация цели: появился заказ от клиента под будущий товар, его пообещали исполнить — надо, чтобы товар при поступлении не ушел другим
- Реальная задача: на складе обнаружили проблемы с конкретным товаром, его отгрузить не могут — надо заблокировать для заказов, а это не успевают сделать

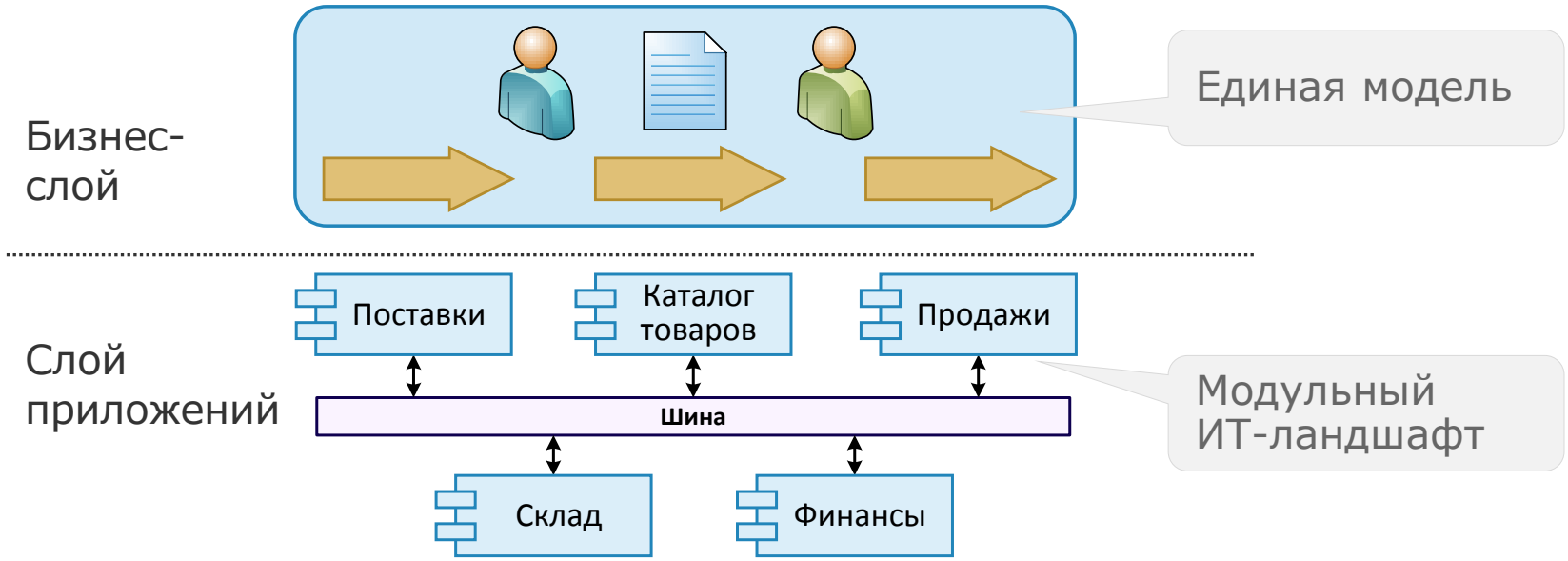


Эти цели предполагают разную логику работы с квотами и взаимодействие заявок между собой; это выяснили, обсуждая решение. А можно было спросить

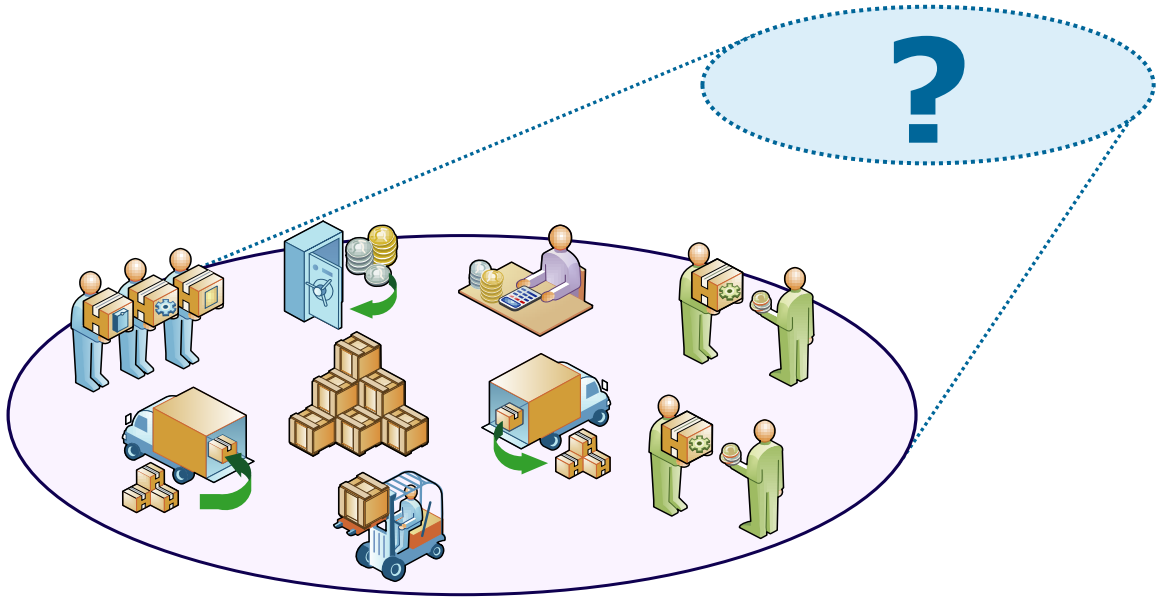
CUSTIS

Модульная модель предметной области: ограниченные контексты

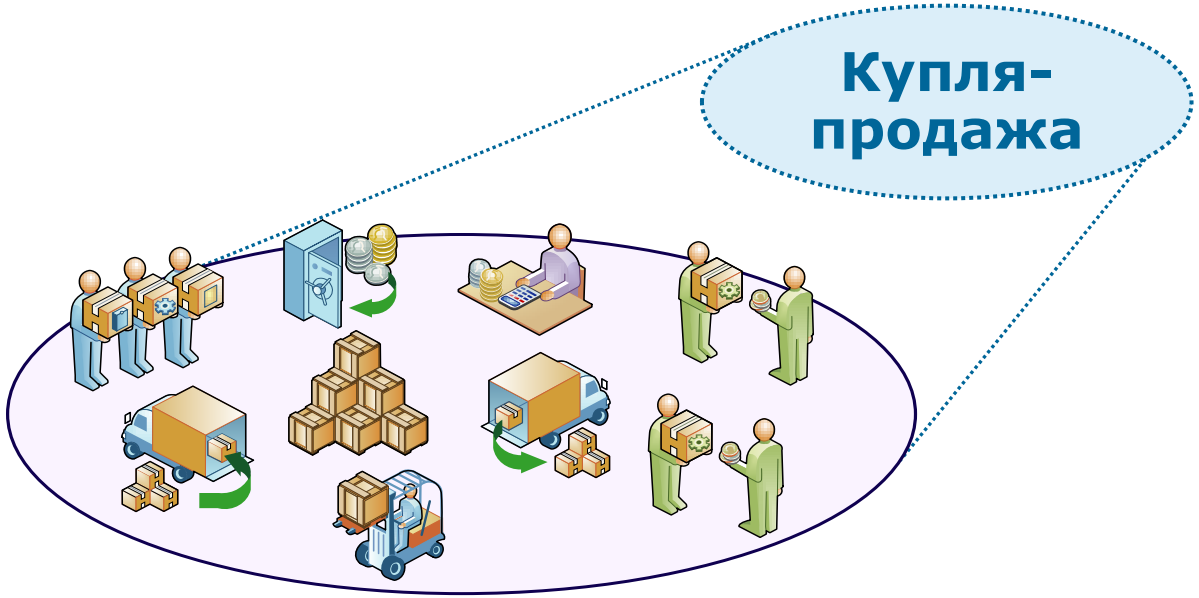
Приложения давно модульные, а **модель предметной области** часто хотят построить на общей непротиворечивой системе понятий



Это основано на убеждении, что существует **непротиворечивая и целостная картина мира,** которую мы можем не знать, но выясняем

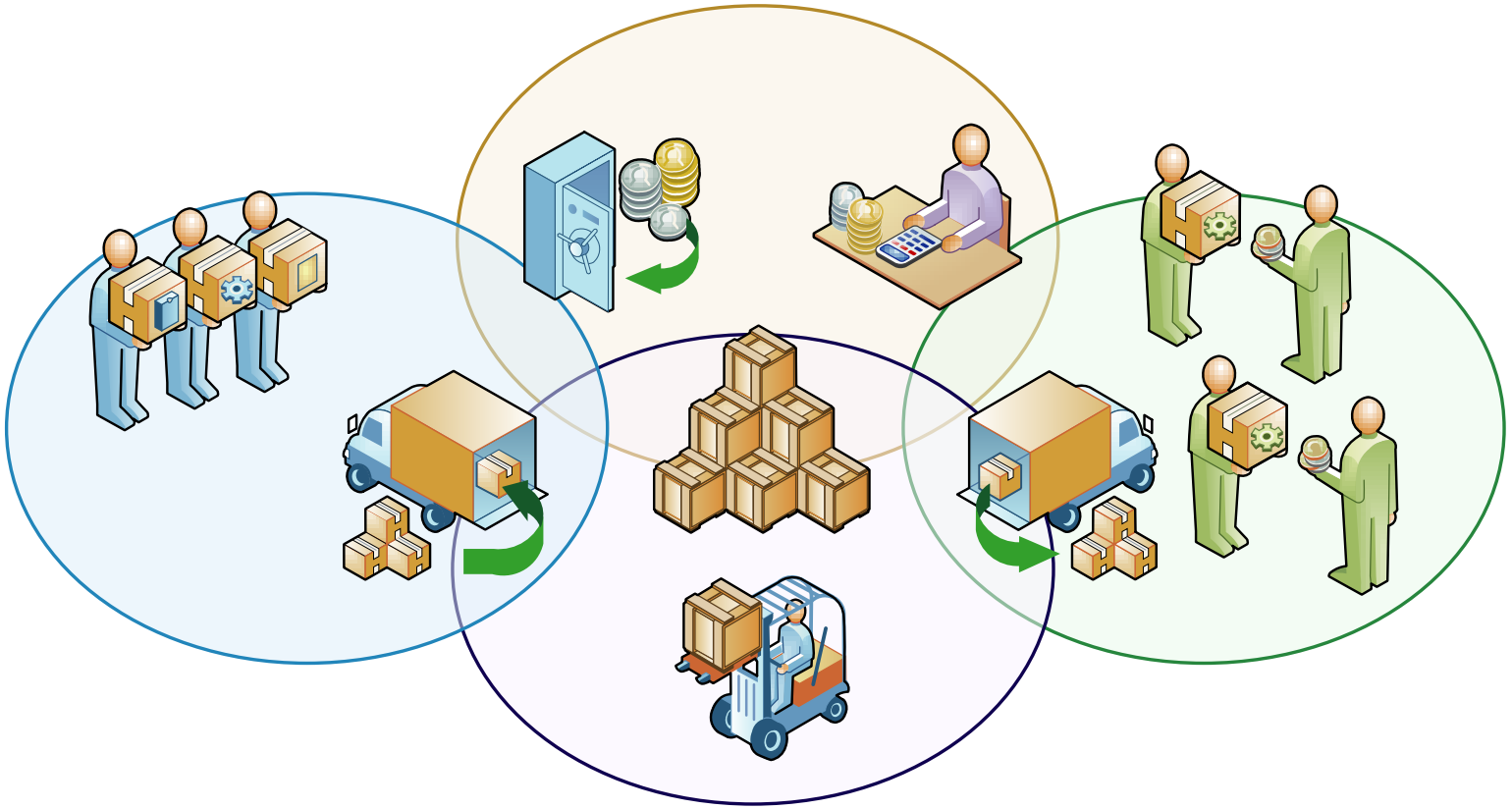


Может быть, в мире идеальных объектов Платона и Декарта так и есть,
но на практике целостная картина оказывается **слишком жесткой**



Решение — ограниченный контекст

CUSTIS



Модель предметной области — модульная

CUSTIS

- Предметную область делим на ограниченные контексты
- Для каждого из них — свой единый язык и своя модель
- Целостность удерживает карта контекстов
- Для изоляции и выделения общего в моделях работают методы ООП
- Модели учитывают позицию фирмы: поставки и продажи — разные контексты и модели, хотя сделка одна — оптовая купля-продажа

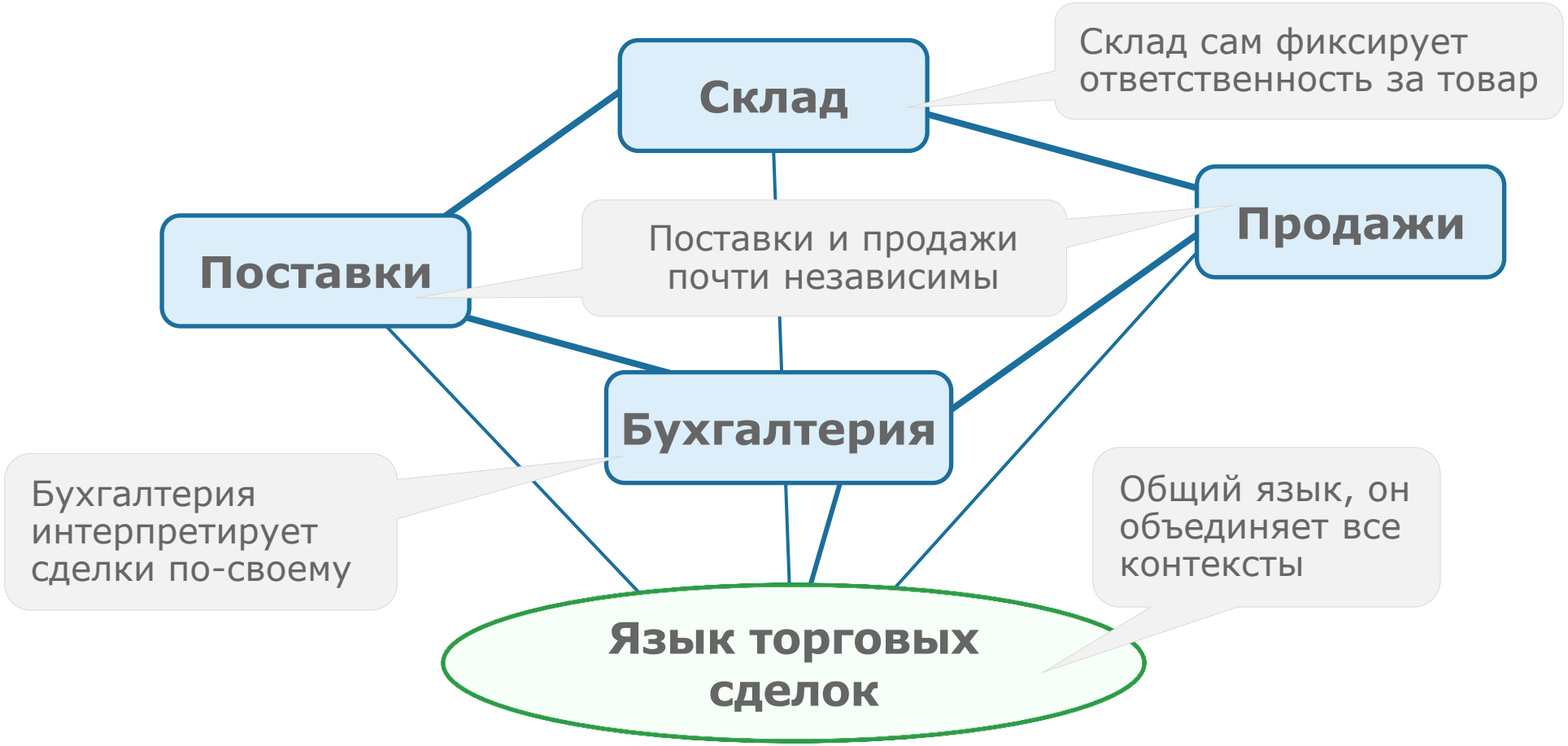
Подробнее — мои доклады «От монолитных моделей предметной области — к модульным» и «Domain-driven design: от справочников и документов до отчетов»

Шаблоны работы с контекстами

CUSTIS

Контексты товарных документов

CUSTIS



Деление не всегда по модулям ИС: каталог товаров должен отражать многообразие взглядов

CUSTIS

Деление на контексты не всегда уместно. Пример: долг клиента

Пример: долг клиента

CUSTIS

Ведение взаиморасчетов с клиентами по продажам

- Обеспечивающее ведение управленческих лимитов
- И отражение расчетов в бухгалтерию

Проблема: смешение языков на бизнес-уровне

- Менеджеры по продажам: долг — основа для управленческих решений. Увеличивается, как только приняли решение об отгрузке, уменьшается, когда признали претензию
- Бухгалтеры: долг — в соответствии с ПБУ, с учетом оформления документов и прохождения процедур
- Следствие: управленческий и бухгалтерский долг имеют систематические различия

Процесс отгрузки товара

Проблемы двух пониманий долга

CUSTIS

- ✕ Контроль правильности учета запаздывает
- ✕ Менеджеров не получается контролировать через бухгалтерию
- ✕ Имеются две разные суммы долга, что затрудняет принятие управленческих решений
- ✕ Для сверки с клиентом и решения проблем менеджеры должны вникать в ПБУ

Проблемы объясняют, почему требуется решение **в одном** ограниченном контексте, а не разделение на два контекста

Решение от DDD

CUSTIS

- Вырабатываем единый язык, описывающий долг в терминах, понятных менеджерам и бухгалтерам
- Строим модель, которая показывает управленческий и бухгалтерский долг и их различие
- Ситуации, в которых долг различается, описываем на едином языке, согласуем со специалистами
- Вырабатываем требования по контролю различий, а также по устранению несущественных различий

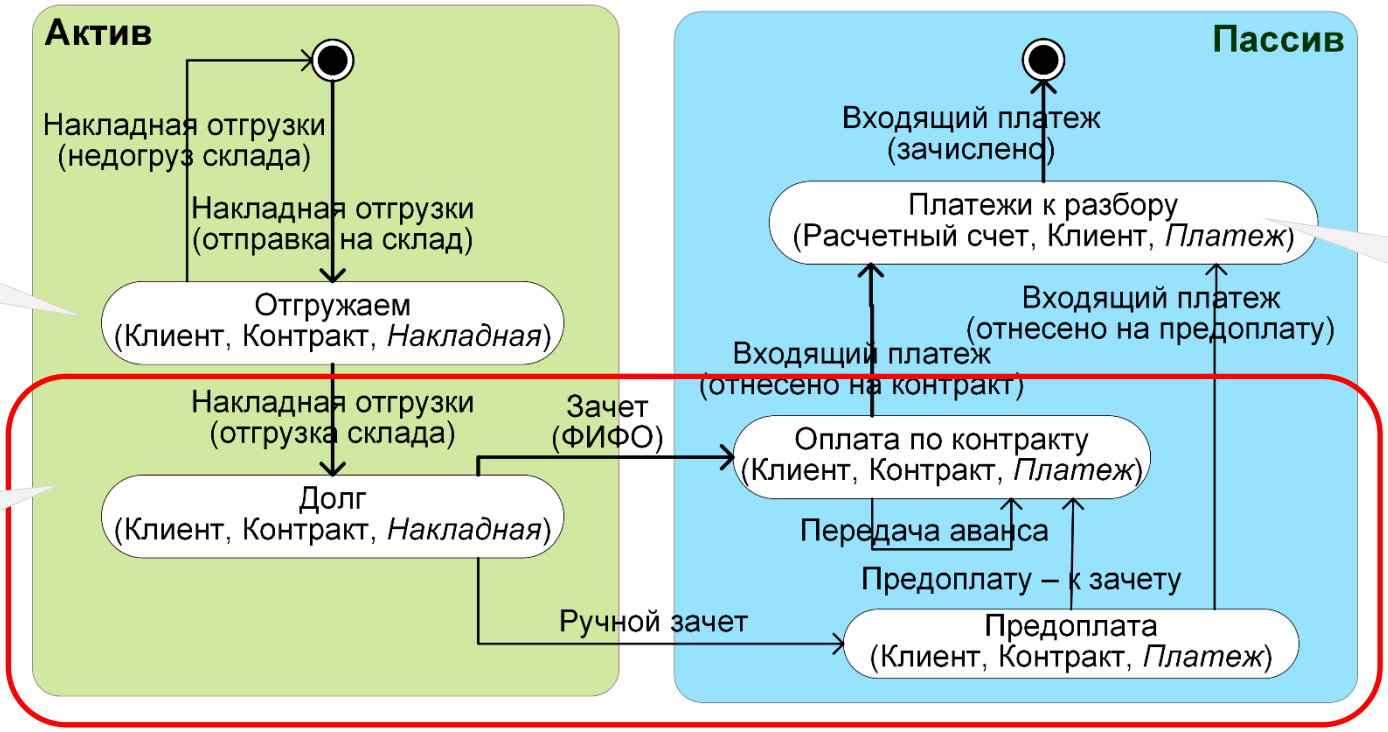
Результат: общий взгляд у бизнес-специалистов на предметную область, описанный в модели, которая реализуется разработчиками

Модель долга клиента

CUSTIS

Этого долга нет у бухгалтеров

Общее понимание долга



Этих платежей нет у менеджеров

Овалы — счета
Стрелки — проводки

И бухгалтеры, и менеджеры понимают данные системы, сверку с клиентом успешно выполняют менеджеры

Дополнения по примеру

CUSTIS

Надо ли привязывать платежи к конкретным отгрузкам?

- Для управления обычно не нужно, а для бухгалтерии достаточно автомата FIFO
- Но если за просрочку есть пени, а частичная предоплата разрешает отгрузку, то это становится необходимо, хотя FIFO хочется сохранить как автомат

Кто может привязать платежи? Бухгалтер при вводе обычно не может, т. к. контекст взаимодействия с клиентом — у менеджера

CUSTIS

Корпоративное приложение

Типичное корпоративное приложение

CUSTIS

Пользователи создают документы

По необходимости заполняют справочники

Потом документы исполняют

При этом меняются учетные данные

Которые влияют на исполнение документов

И отражаются в отчетах

Объектная модель

Жизненный цикл документа

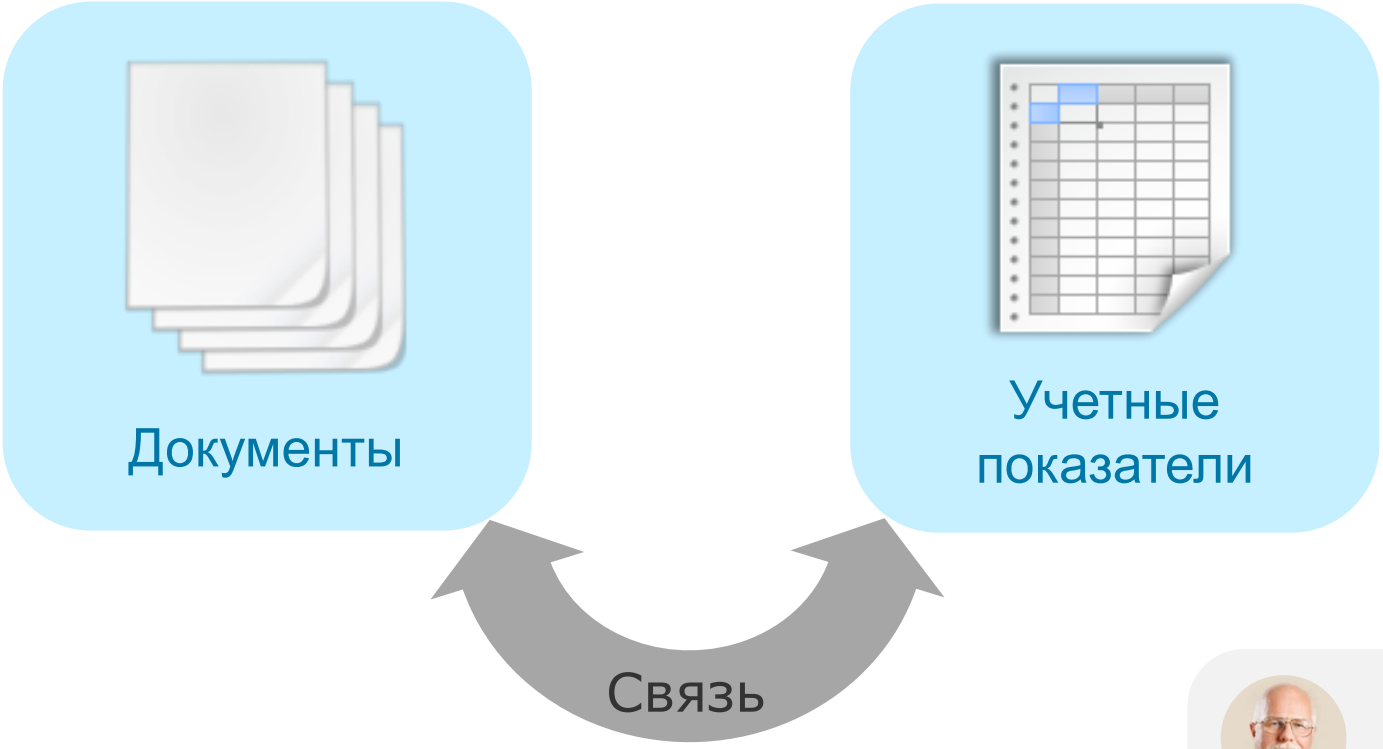
Учет не объектная модель

Классические приложения

CUSTIS

- Клиент-сервер, на сервере — СУБД, она обеспечивает конкурентную работу, транзакционность и консистентность данных в рамках обработки запроса пользователя
- Трехзвенная архитектура не принесла принципиальных изменений, она позволила писать бизнес-логику на сервере на объектном языке
- Масштабирование — за счет железа и системного софта
- Классическое проектирование: **процедурный** и **объектный подход**
- Для классических постановок есть хорошие средства визуального проектирования: ER-модель и диаграммы классов, потоков данных, модулей, последовательности и др.

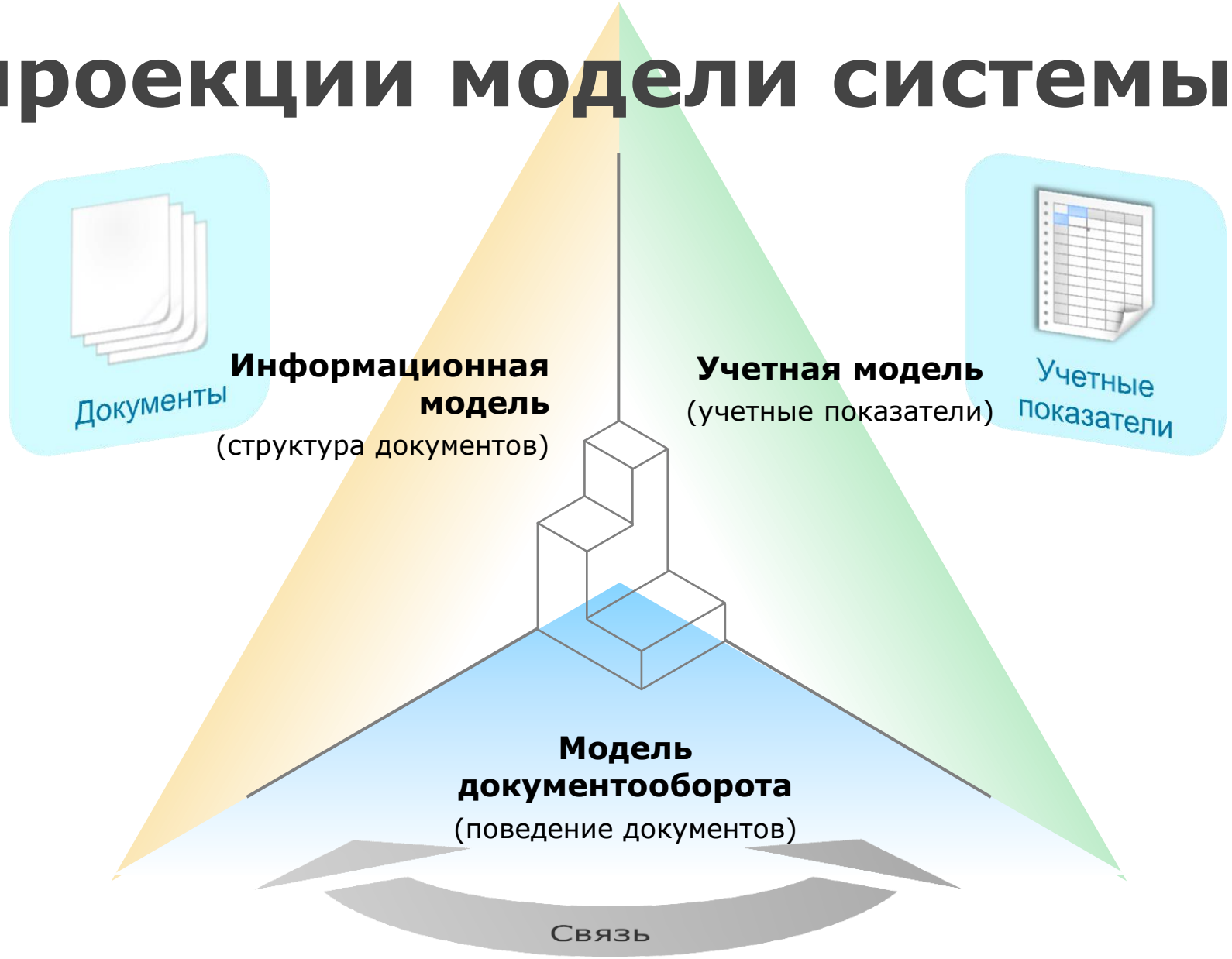
Одна модель или несколько?



А то при документах появляется свой «маленький учет»

Если учет существенно влияет на исполнение документов, то необходима единая модель

Три проекции модели системы



Диаграммы для проекций

CUSTIS

Диаграмма классов

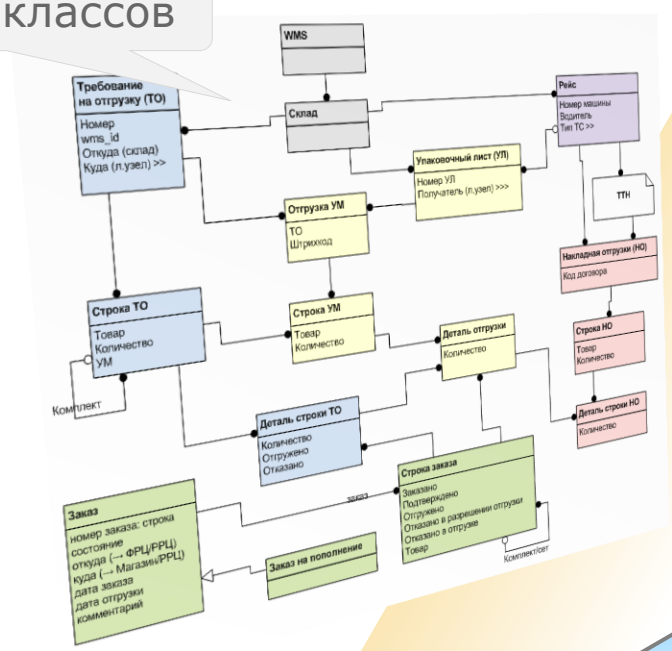


Диаграмма учета

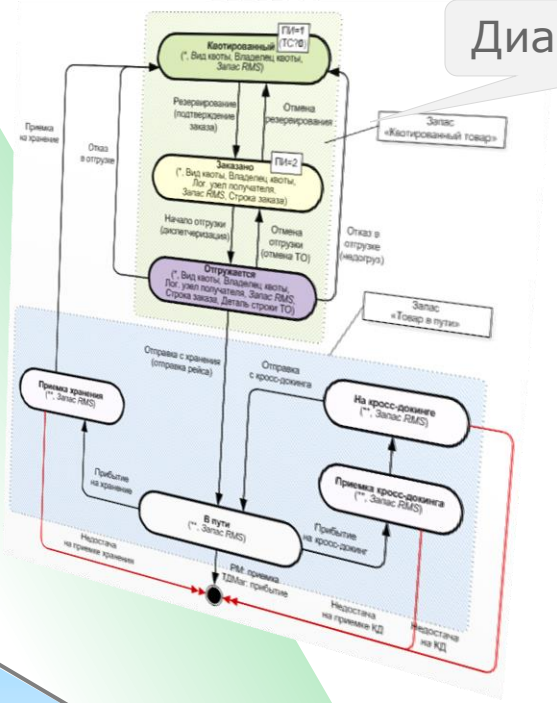
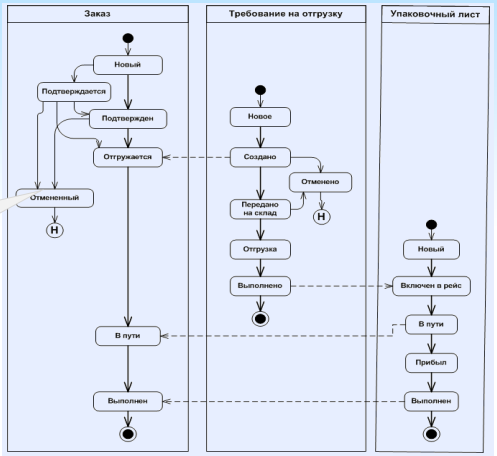
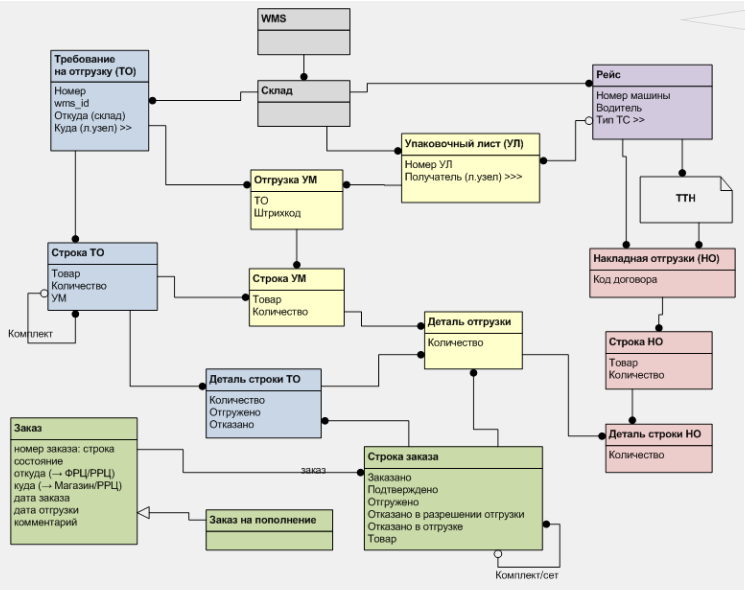


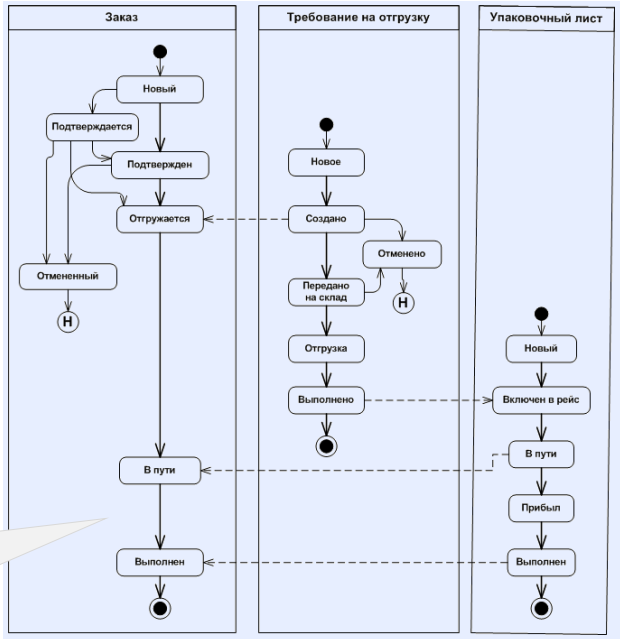
Диаграмма состояний



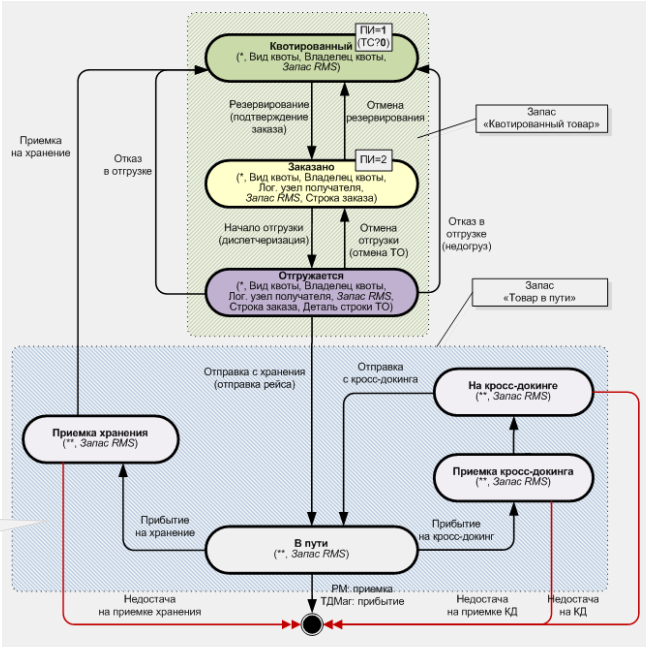
Пример: шаблон для корпоративного приложения



Пользователи создают документы, по необходимости заполняют справочники: используем **объектную модель**



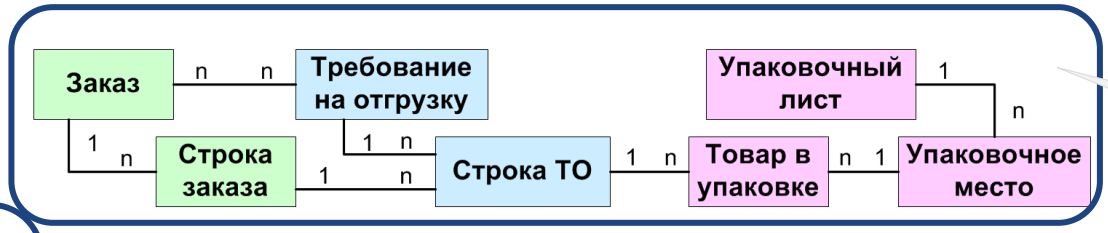
Потом документы исполняют: используем переходы документов между состояниями (State Entity)



При этом меняются учетные данные, которые влияют на исполнение документов и отражаются в отчетах: используем **учетную модель**

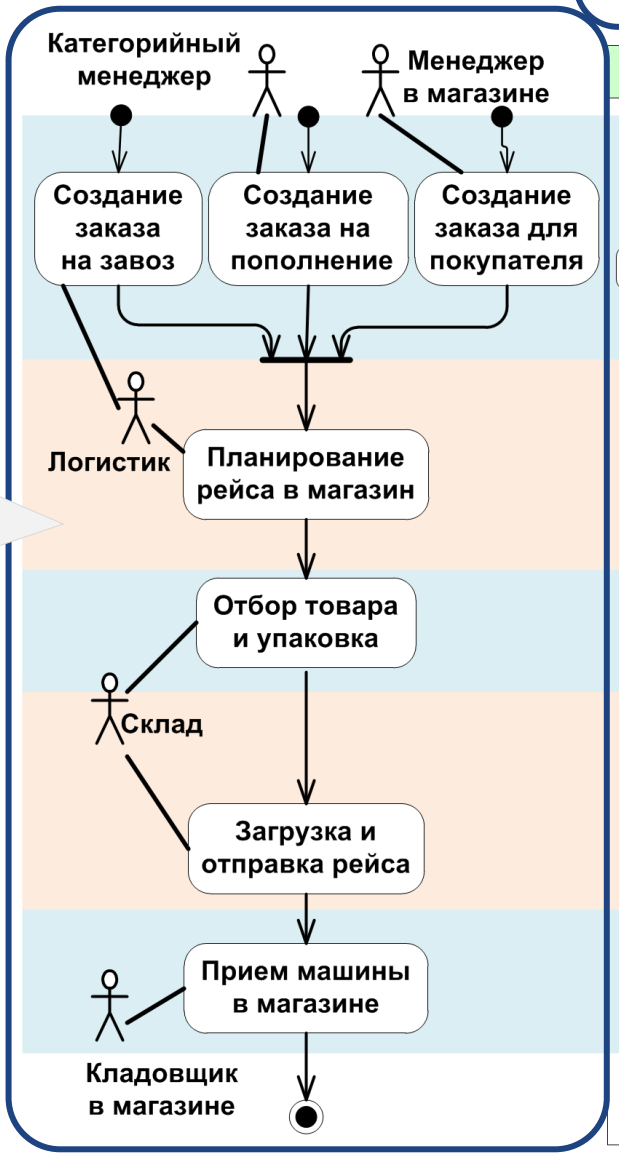
От бизнес-процесса к документам

Снабжение магазинов



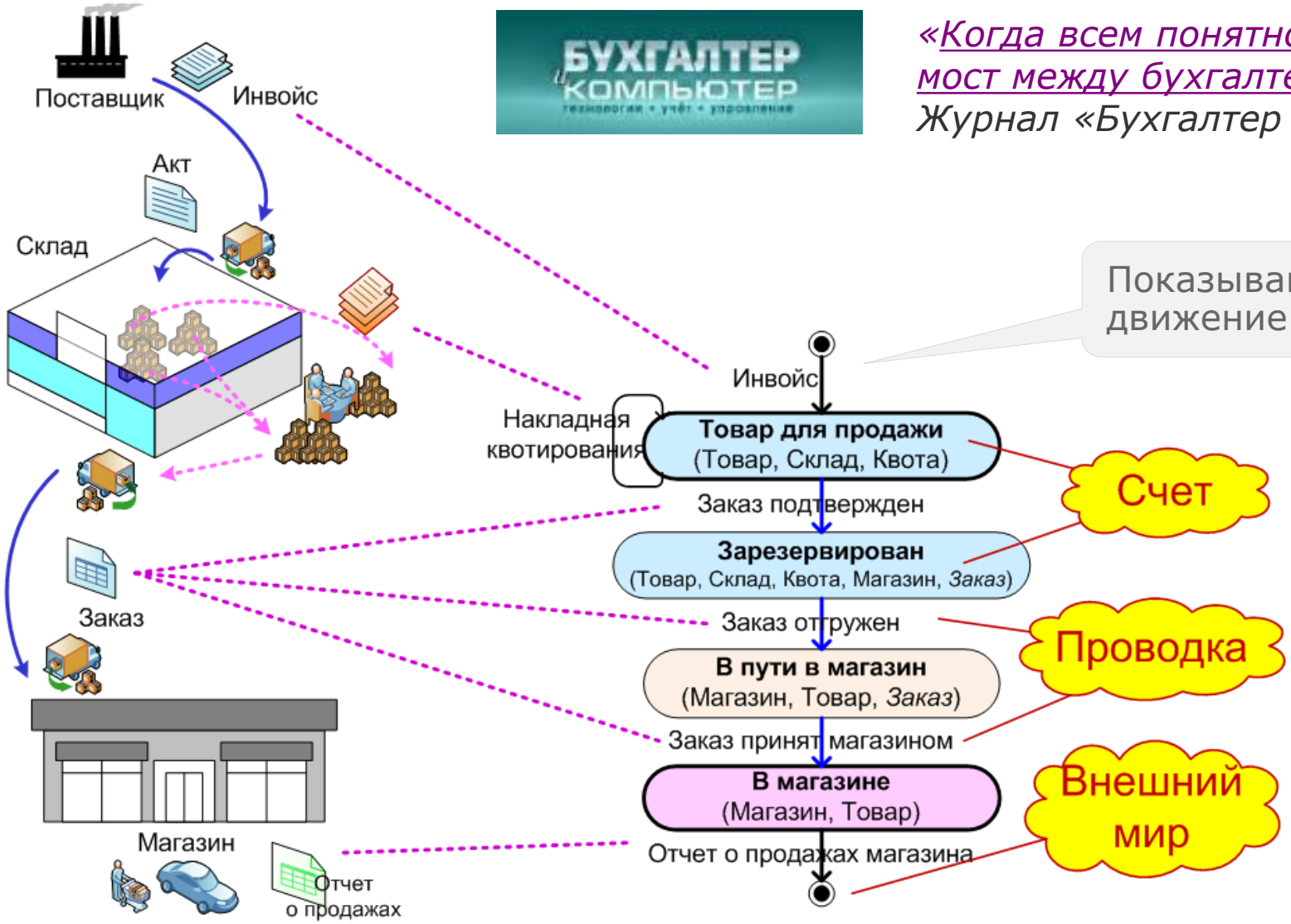
Выделяем объекты-документы

Бизнес-процесс: диаграмма активности



Переходы фиксируют **выполнение** бизнес-функции, а **состояния** соответствуют **передаче по этапам** обработки

От документов к учетным показателям



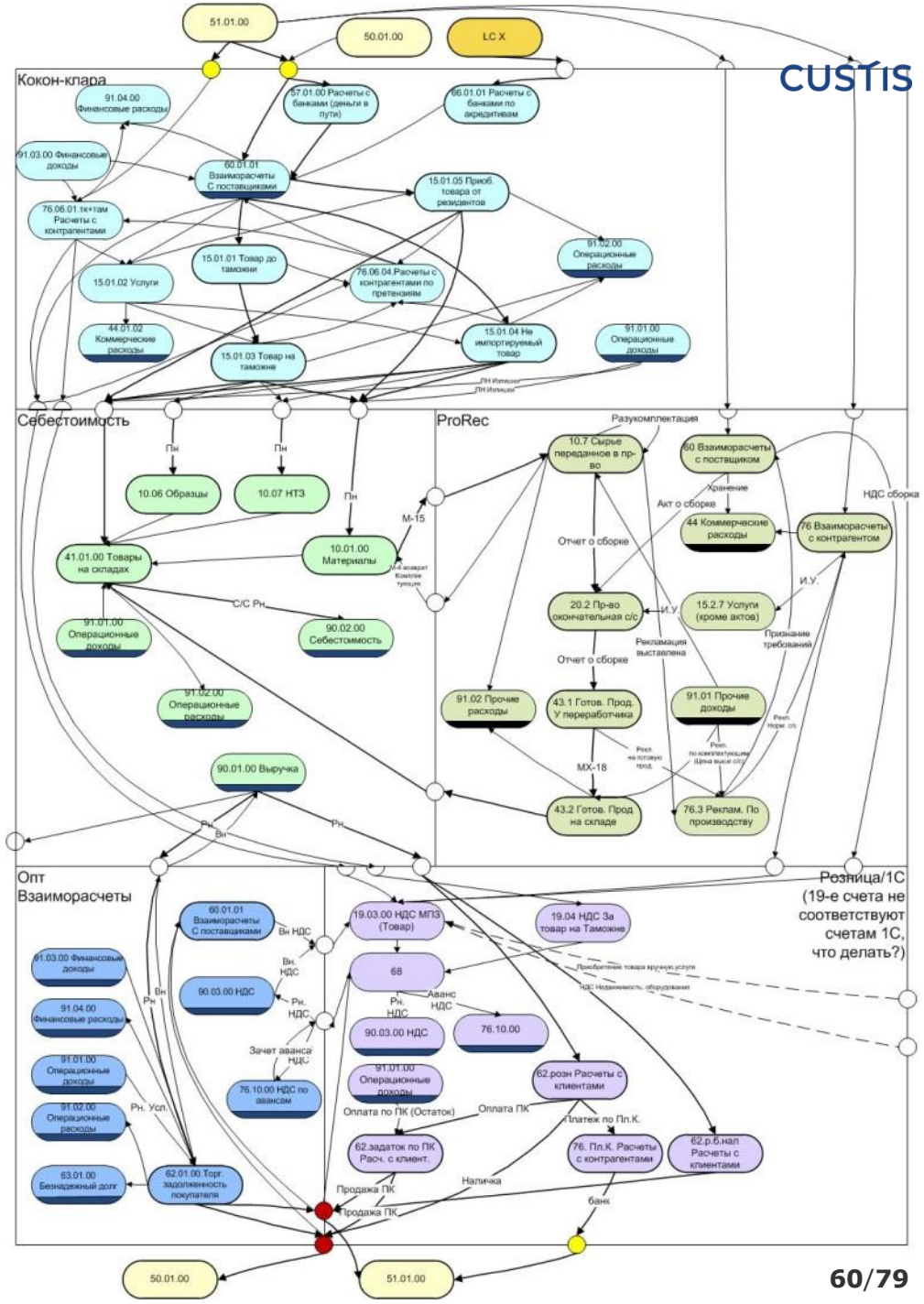
«Когда всем понятно. Диаграммы учета: мост между бухгалтером и разработчиком»
Журнал «Бухгалтер и компьютер», №5-2011

Показывают, как отражается движение ресурсов в учете

Учет — взгляд через поток ресурсов

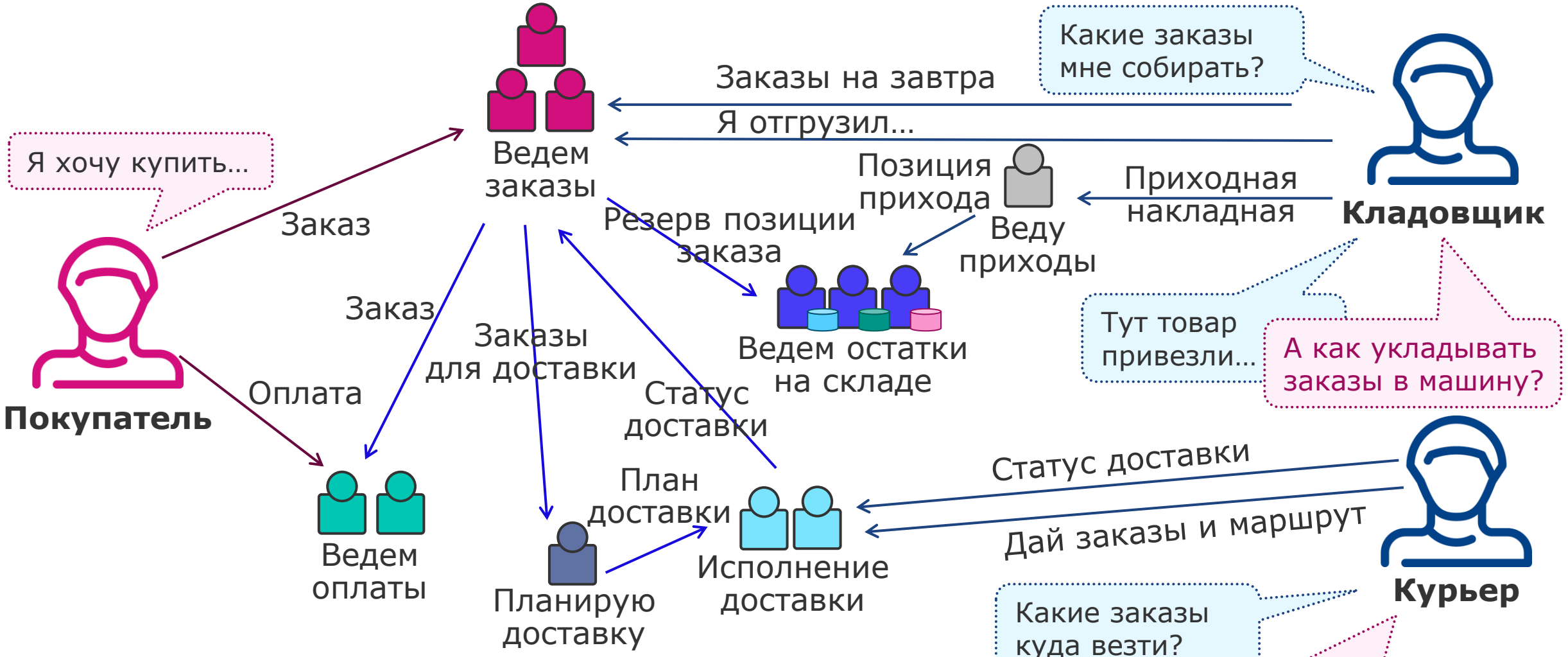
Полная схема учета по МСФО торговой компании

Учет дает целостную картину,
а диаграмма позволяет ее увидеть



Модель для сервисной архитектуры

CUSTIS



Из доклада «Визуальное проектирование масштабируемых приложений»

CUSTIS

Отражение модели в код

Проблемы использования одной модели

CUSTIS

- Необходимость понимания модели заказчиком существенно ограничивает моделирование
- Технические подробности и сложные формальные методы становятся недопустимы
- А без них модель не может служить проектом для реализации, нужно дополнительное проектирование

Единственная модель на едином языке —
и преимущество DDD, и источник проблем

Большие и сложные объекты

CUSTIS

Бизнес-объект включает все аспекты деятельности

- Клиент — субъект делового мира, партнер по сделкам; взаимоотношения — юридические и управленческие
- Заказ — полный жизненный цикл от начальных переговоров до исполнения, включая юридическое и другое оформление

Бизнес-объекты тесно связаны между собой

- ✗ При отражении в код объекты получается очень похожими на антипаттерн Big Objects
- ✗ Сложно понимать, развивать, тестировать

Техническое усложнение модели

CUSTIS

- Принципы ООП побуждают к декомпозиции, что увеличивает число объектов
- Применение шаблонов (стратегии и др.)
- Технические и инфраструктурные атрибуты и классы
- Ограничения на объекты со стороны фреймворков и обходные пути для их преодоления

✕ Сложная модель непонятна заказчику, простая — не отражает реализацию

Решение: технологические рельсы

CUSTIS

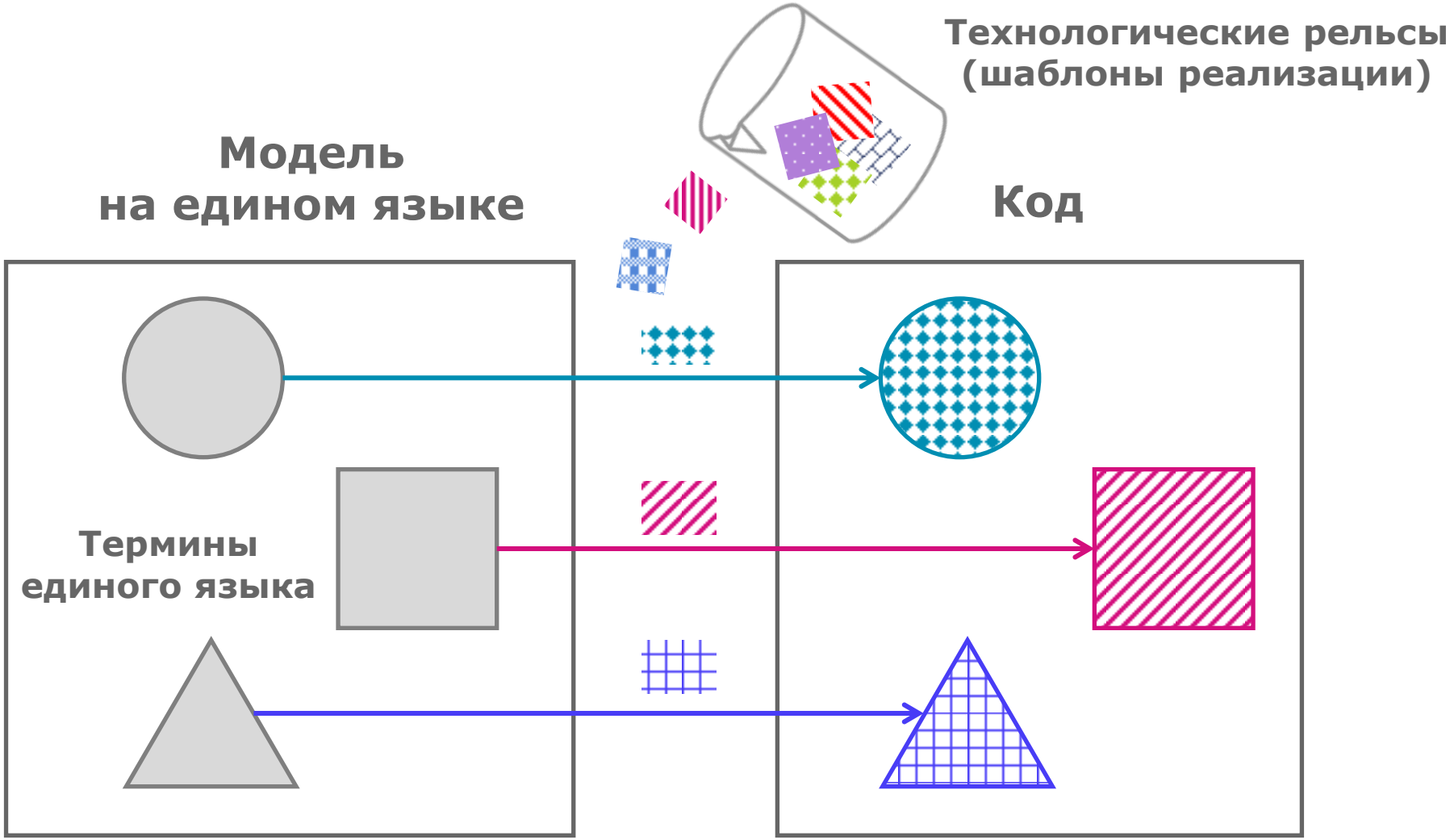
Технологические рельсы — это способ перевода модели на едином языке в код

- Платформы, фреймворки и библиотеки
- Способы их организации в единое приложение
- Способы отражения модели приложения в код
- Типовые задачи и design patterns для их реализации

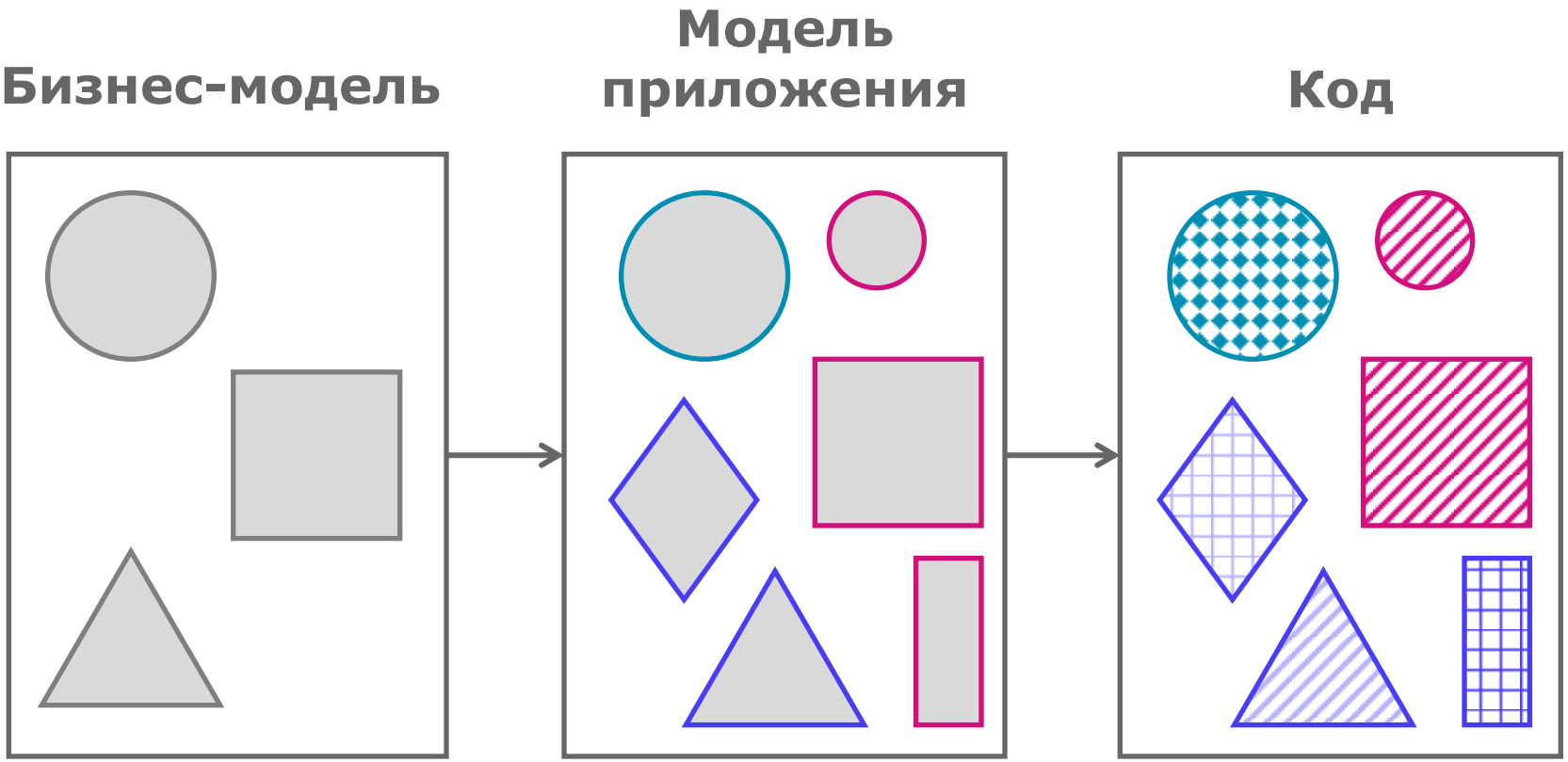
Формулируются на всех уровнях — от архитектуры до частных задач посылки сообщения

«Применяем Domain Model и Rich Objects» — частный случай технологических рельсов

По рельсам


Технологические рельсы
(шаблоны реализации)
Модель
на едином языке
Код
Термины
единого языка

А как без рельсов



CUSTIS

Технологические рельсы для современных приложений

Отражение бизнес-объектов

CUSTIS

- Анемичный транспортный объект или несколько идентичных
- Прозрачное отражение транспортного объекта в БД (с ORM или без)
- Класс-контроллер бизнес-логики на бэкенд или несколько — для одного объекта, фабрика для создания, для обработки коллекций
- Классы-контроллеры логики для фронтенда и для клиентов

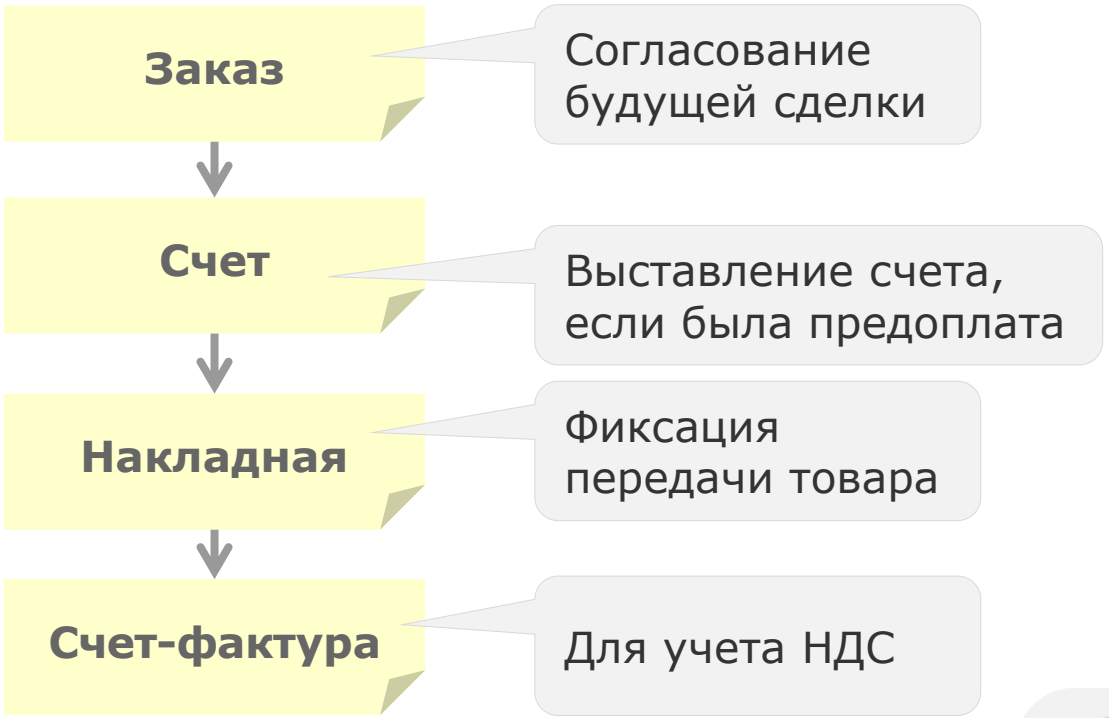
Технологические рельсы

- Генерация всех объектов данных и БД по одному описанию (JSON, XML)
- Решение для workflow на основе state entity
- Библиотека или базовый класс для naked-objects интерфейсов
- API, уведомления, печать и другая инфраструктура — на описаниях

Пример: оптовая торговля

CUSTIS

Сделка купли-продажи — нормативный workflow

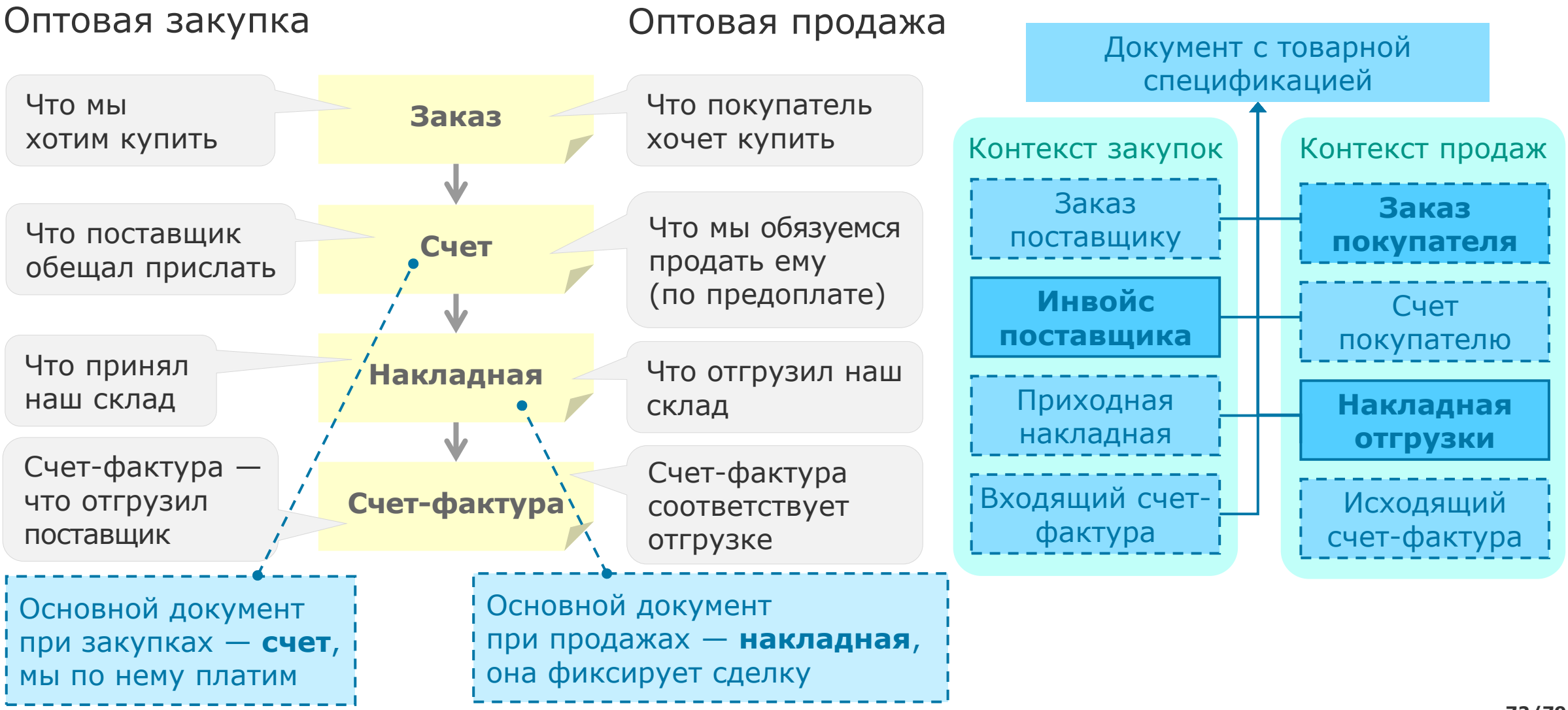


Вариант объектной структуры

Вариант — плохой, не учитывает различие контекстов и позицию компании в сделках

Разница контекстов сделки

CUSTIS



Именование объектов

CUSTIS

- Бизнес-объекты, их атрибуты и методы называются по-русски
- Для них в модели должны быть зафиксированы английские названия
- Политика: перечисление и (или) простые справочники
- Все названия в коде — на основе английских названий в модели
- Надписи в интерфейсе — на основе русских названий в модели
- Технологичная поддержка для переименования объектов и атрибутов:
  - Смена русского названия (полу)автоматически отражается в интерфейсе
  - Смена английского названия — через (полу)автоматический рефакторинг
  - Регламент для изменения — преобразование перечисления в справочник
- Однородное решение для получения пользовательских названий экземпляров объектов, используемых в сообщениях и логах

Workflow на основе шаблона state entity

CUSTIS

- Принцип: единственный атрибут документа отражает его состояние в workflow, возможные действия над ним и права на их совершение
- Граф переходов отражает возможные изменения состояния
- Граф переходов отражает реализацию документом бизнес-процесса, согласован и понятен бизнесу
- Реализация собственным движком или библиотекой типа Activity
- Интерфейсы единообразно опираются на состояния документов, включая проверку на интерфейсах и логику доступных действий

Джимми Нильссон разбирает четыре способа реализации workflow. Я считаю state entity наиболее технологичным, см. мой доклад «Domain Driven Design — от требований до кода»

Выделение базовых абстракций

CUSTIS

Для разработчика логично выделение базовых абстракций

- Справочники — объекты без workflow, но с выделением черновиков и устаревших
- Документы — объекты с workflow, реализующие бизнес-процесс
- Классы документов: «договор», «документ с товарной спецификацией» и т. п.
- Обобщенная печать для любых документов
- Обобщенная интеграция для любых объектов и т. п.

Для бизнеса эти абстракции не существуют, он мыслит реализацию объектов как не зависящих друг от друга

✓ Базовые абстракции можно выделять как инфраструктурные объекты

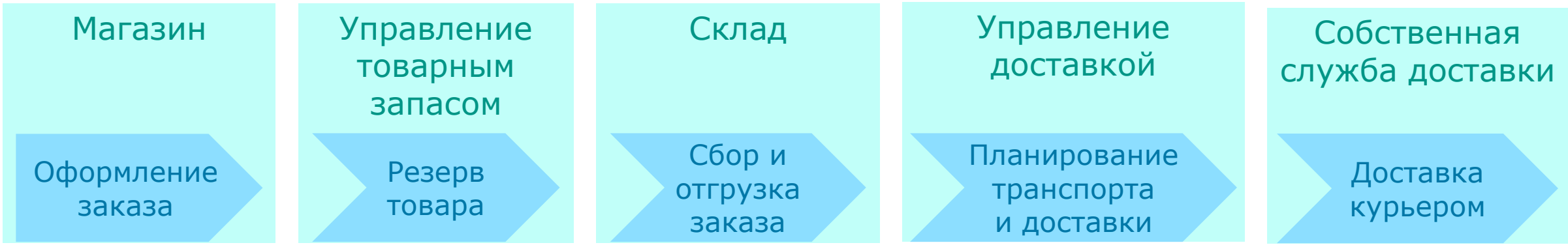
✗ Проблема: вынесенное в базовый объект ограничивает изменения

Пример: если печать ТОРГ-12 или счета-фактуры реализовать в классе «Документ с товарной спецификацией», то контексты закупок и продаж оказываются связаны в инфраструктурном объекте

Гибкость инфраструктуры

CUSTIS

Пример: продажа с доставкой в розничной торговой сети



Инфраструктура интеграции поддерживает движение объекта по бизнес-процессу

- В одних системах может быть общий анемичный объект «Заказ покупателя», в других он преобразуется на входе и выходе из собственного представления
- Универсальное представление в шине дает узкое горло
- Надо уметь добавить к заказу «план проезда» — скан рисунка покупателя, который нужен только курьеру

Единство кода для back, front и интеграции

CUSTIS

Логика, которая присутствует в нескольких слоях приложения:

- Бизнес-логика доступности действий на формах и проверка прав
- Бизнес-логика проверки значений для атрибутов и удобного ввода

Проблемы, требующие технологичного решения:

- Как обеспечивается соответствие логики в разных слоях: вручную, одинаковым языком реализации, генерацией на основе метаописаний или конфигураций, как-то иначе
- Как поддерживается работа бэка с многими версиями фронта и API

DDD и сервисная архитектура приложения

CUSTIS

Предметная область разбивается на ограниченные контексты

Каждый контекст реализуется одним или несколькими сервисами, возможен композит — несколько сервисов и общая БД

- Ведение покупателей может быть включено в ведение заказов или отдельно

Не все сервисы основаны на объектной модели

- Примеры: учетные сервисы ведения остатков, сервисы работы с расписаниями

Что же достигает DDD?

CUSTIS

Единый язык + единая модель:

- Надежная основа коммуникации всех участников проекта при принятии решений
- Успешно заменяют мелкую россыпь требований
- Позволяют эффективно развивать сложную систему

DDD в современной архитектуре требует более сложного отражения в код, чем для монолитов, но технологические рельсы решают задачу

http://mtsepkov.org

@MaximTsepkov

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

Вакансии

Пишите на hr@custis.ru, подходите с вопросами