Domain-driven design:
от справочников и документов до отчетов

Максим Цепков

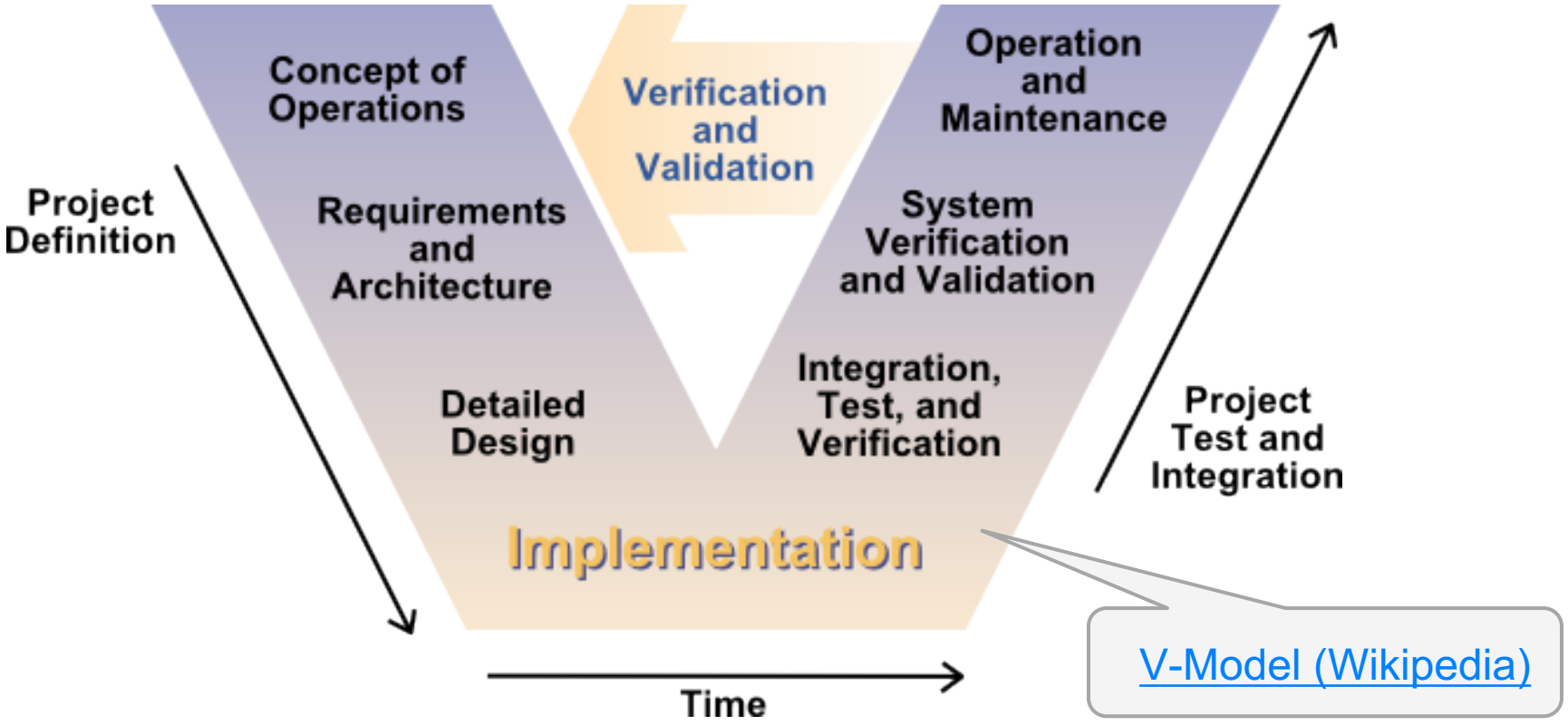
IT-архитектор и бизнес-аналитик,
Навигатор и эксперт по миру Agile,
бирюзовых организаций и Спиральной динамике

http://mtsepkov.org

World Information Architecture Day (WIAD)
Площадка в Москве 21.02.2020

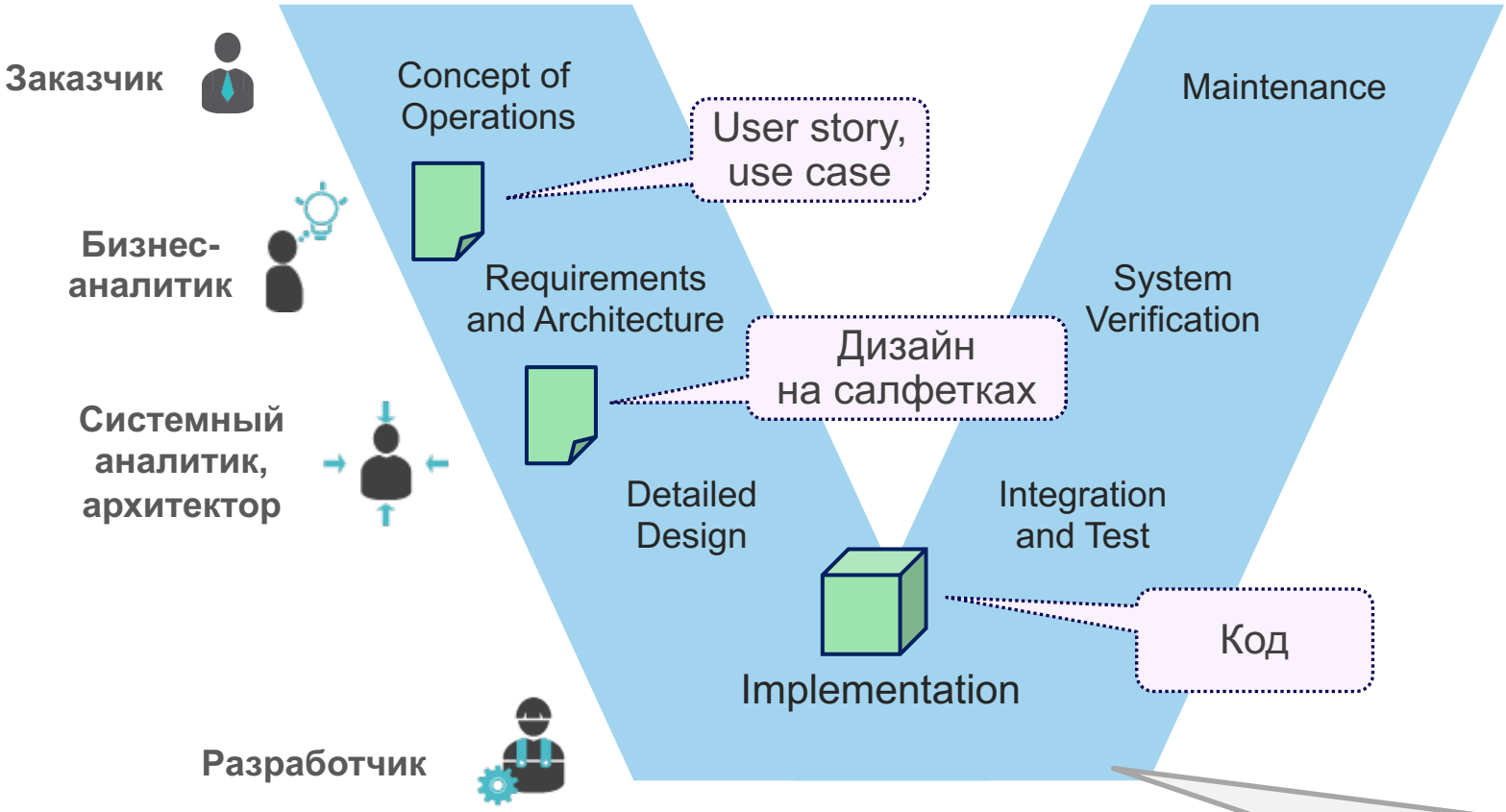
Domain driven design: что это и зачем?

Путь проекта – V-модель



[V-Model (Wikipedia)]

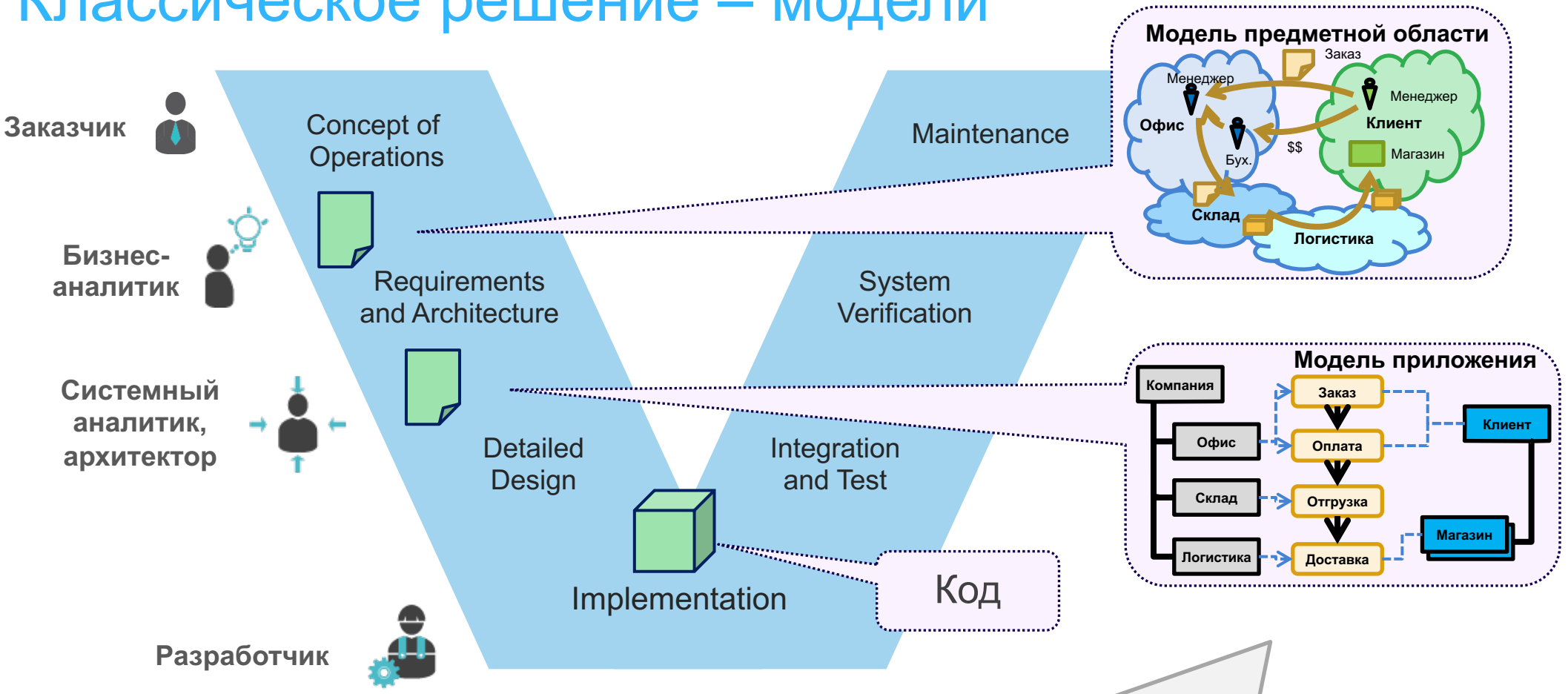
Проект можно сделать без моделей



Проблемы:

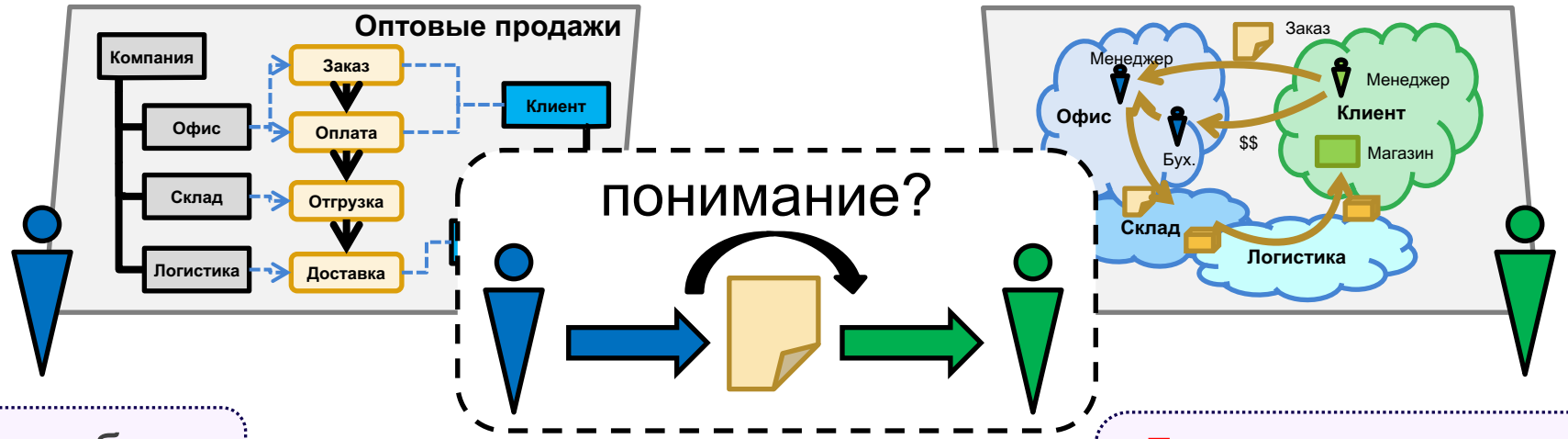
- нет целостного представления
- реализуемость не понятна

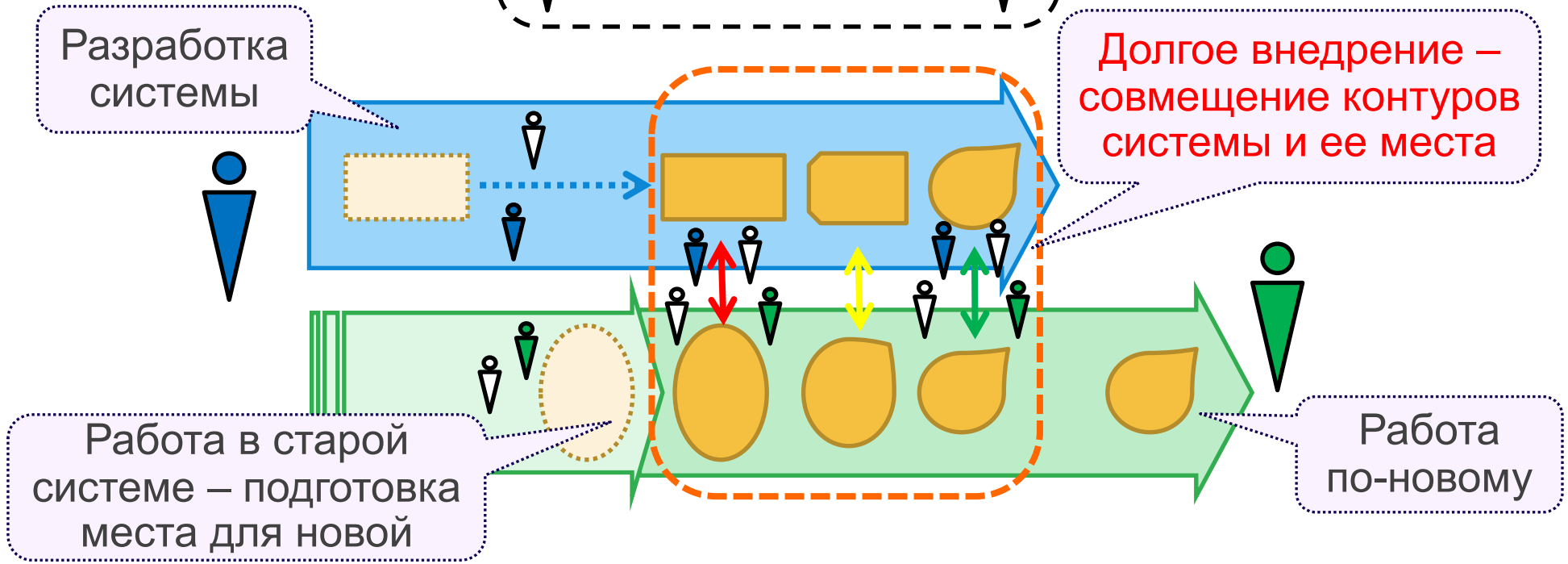
Классическое решение – модели



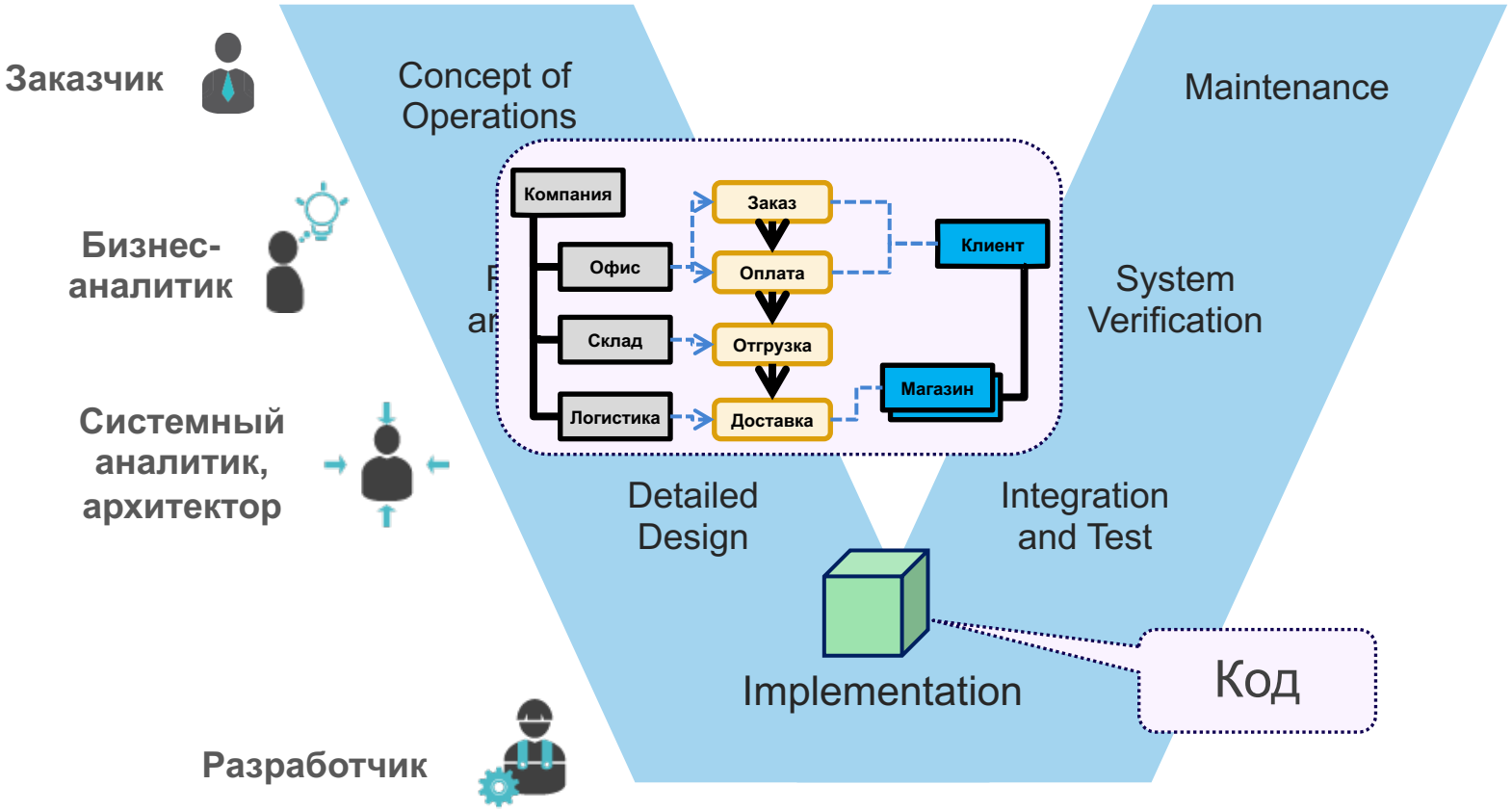
Проблема: каждый уровень отдельно – сложно изменять и поддерживать соответствие

Проблема двух моделей





DDD: единая модель, единый язык, прозрачное отражение в код



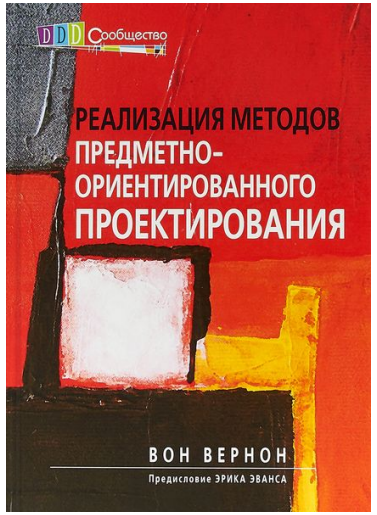
DDD – источники

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)

- Построен на основе терминов предметной области
- Его понимают 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

- Аналитик собирает требования и строит модель: сначала – предметной области, затем – системы
- Артефакты модели описывают и 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

- ☺ Верификация постановок бизнес-специалистами
- ☺ Общее понимание требований на стороне бизнеса
- ☺ Обсуждение модели бизнесом и ИТ, поиск баланса в сложных решениях
- ☺ 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

Отражение модели в код

Проблемы использования одной модели

- 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

- Бизнес-объект включает все аспекты деятельности
  - Клиент – как 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 IT-объекты получается очень похоже на антипаттерн Big Object

☹ Сложно понимать, развивать, тестировать

Техническое усложнение модели

- 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 не понимается заказчиком, простая – не отражает реализацию

Решение – технологические рельсы

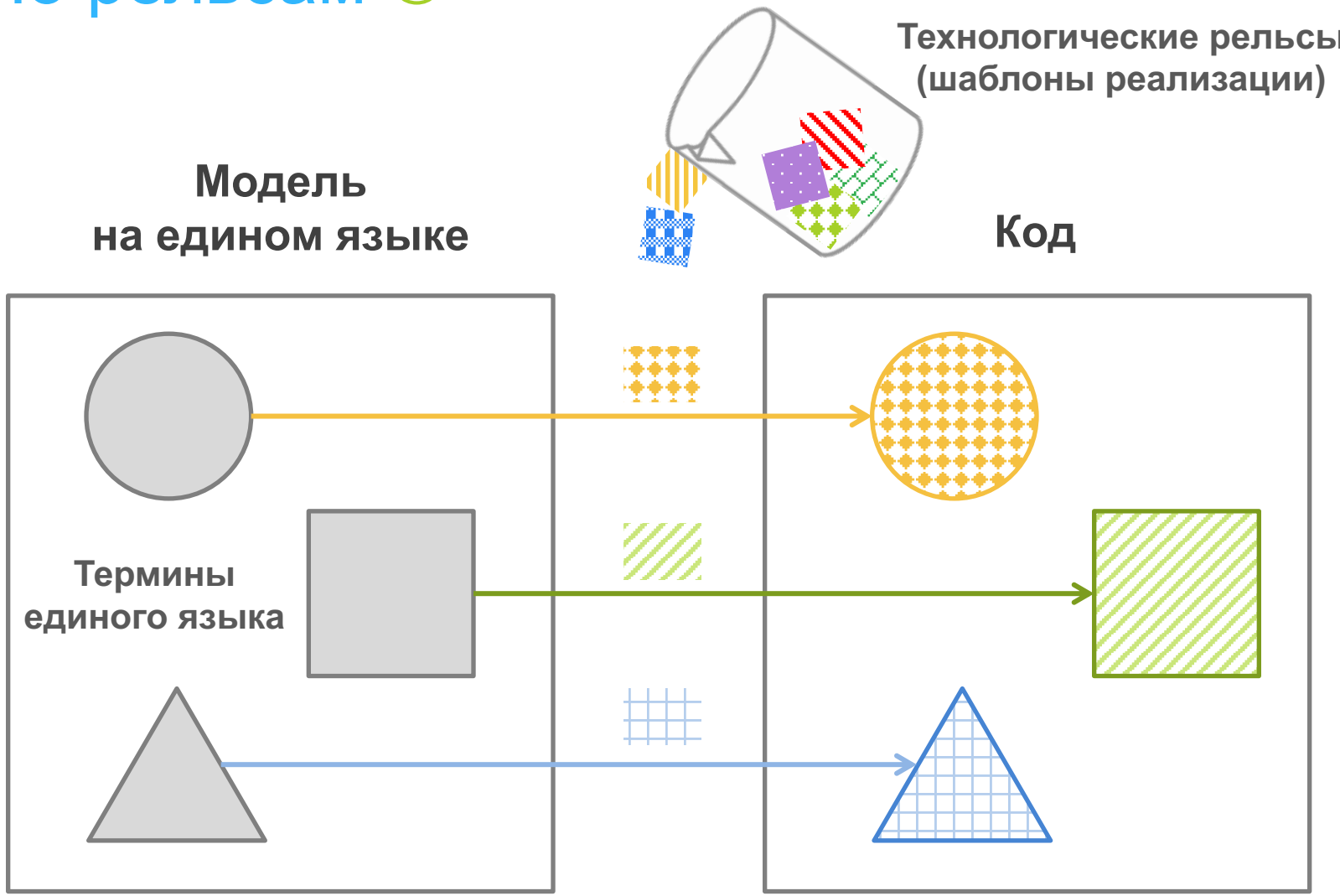
Технологические рельсы – это способ перевода модели на едином языке в код

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

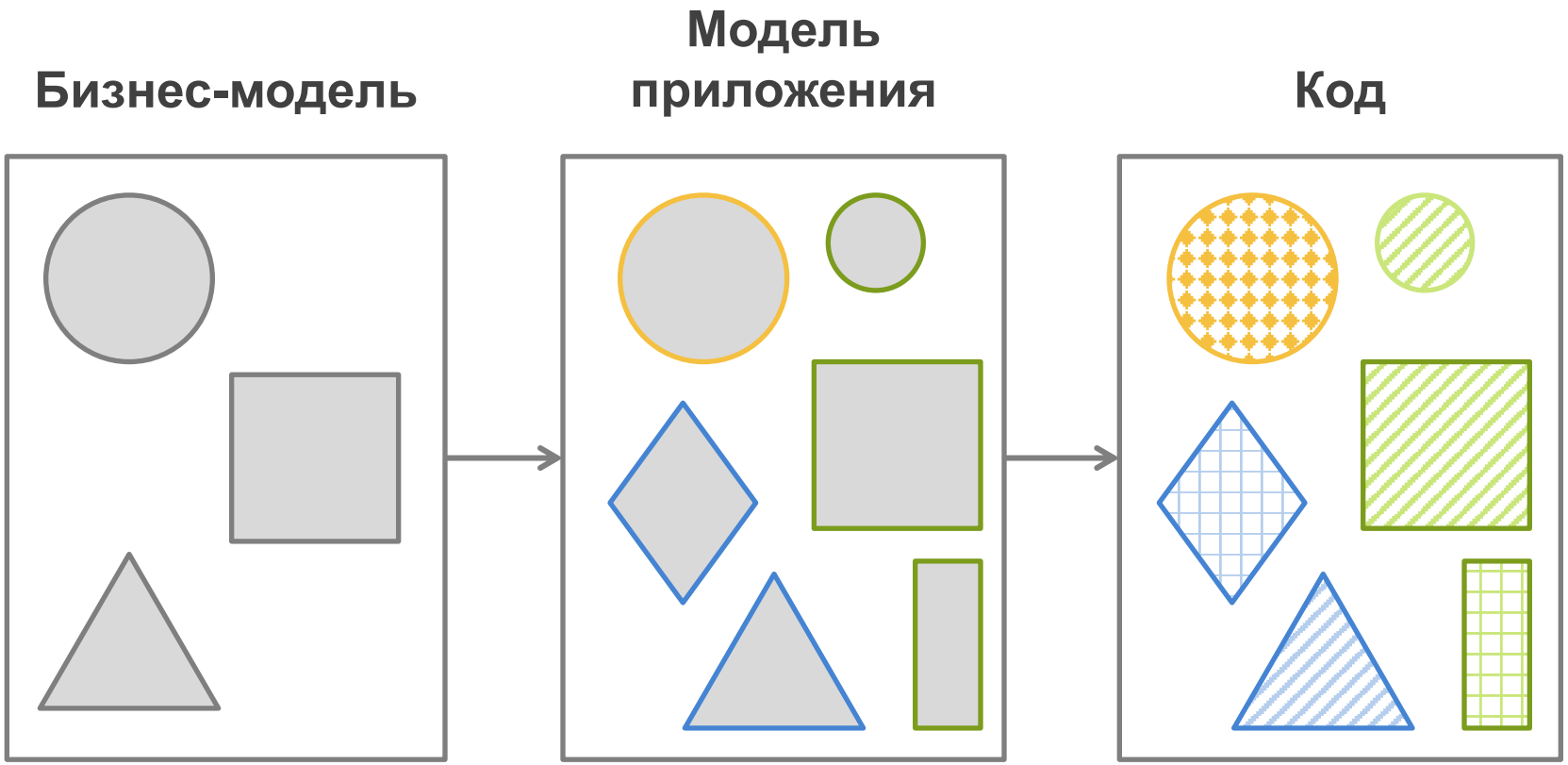
Формулируются на всех уровнях – от архитектуры
до частных задач посылки сообщения

«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



Корпоративное приложение

Типичное корпоративное приложение

- 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

Одна модель или несколько?

Документы

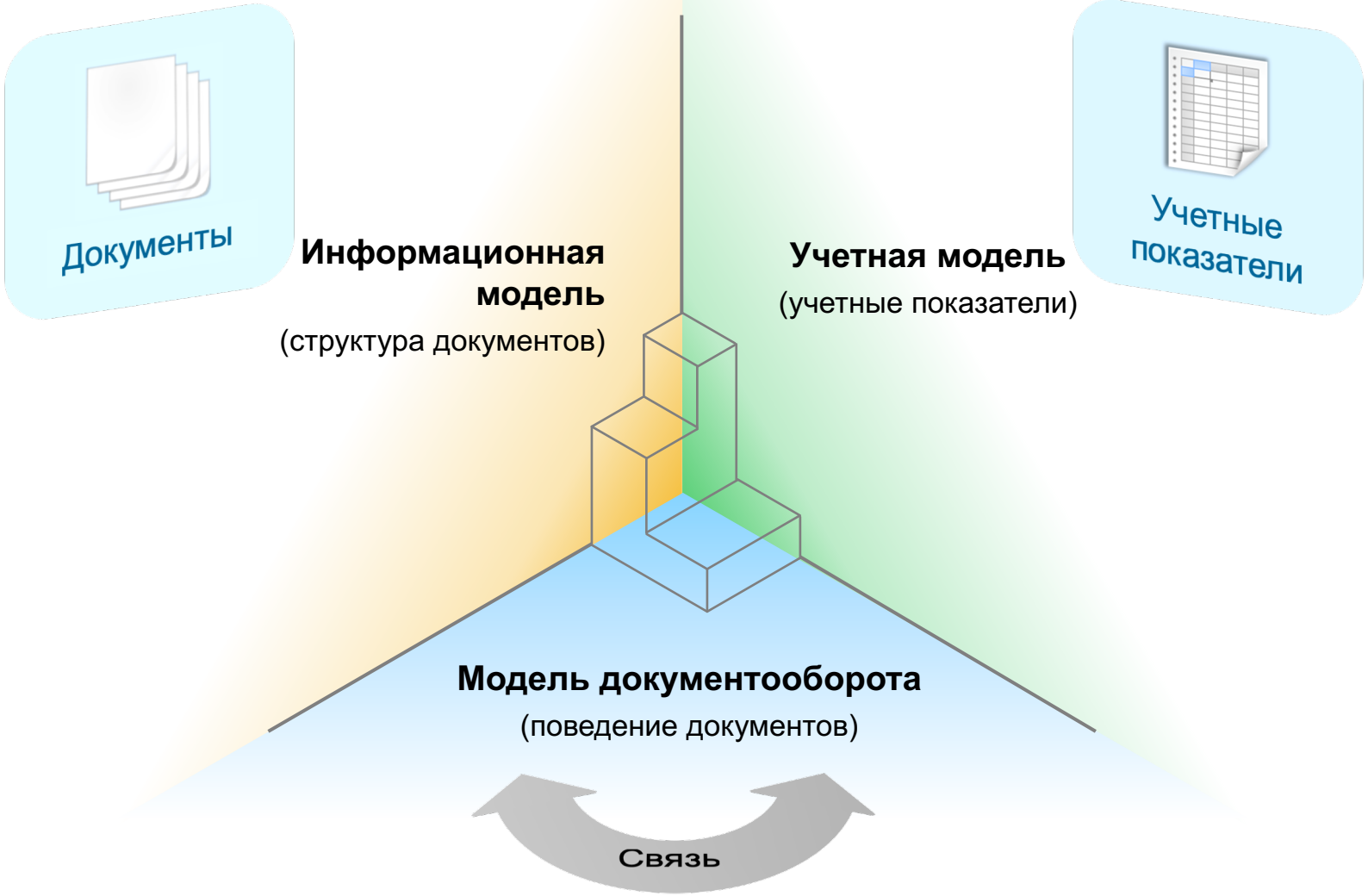
Учетные показатели

Связь

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

Диаграмма классов

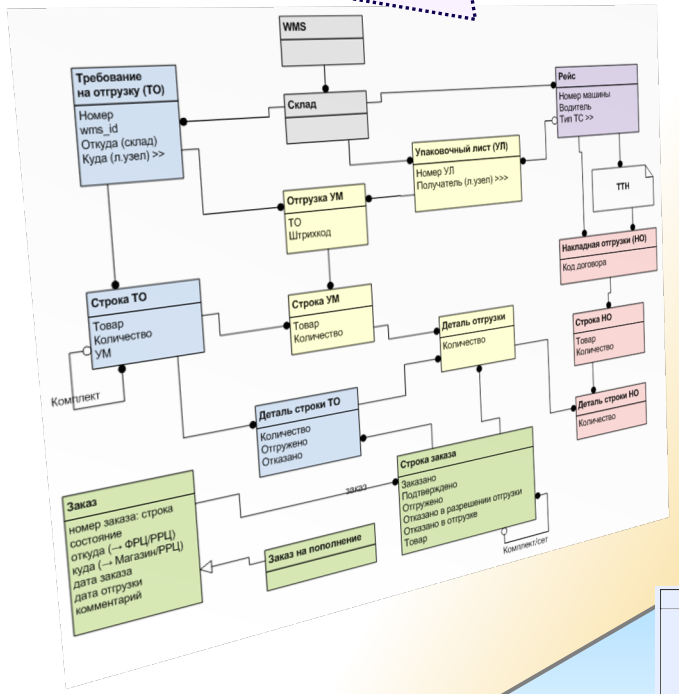


Диаграмма учета

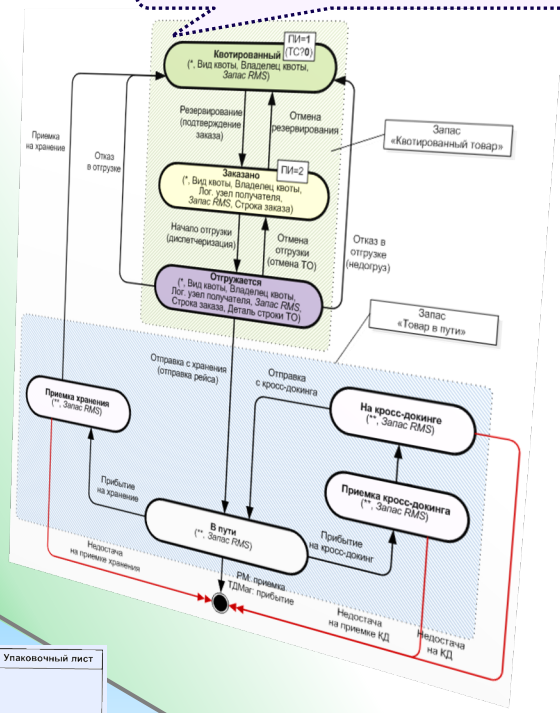
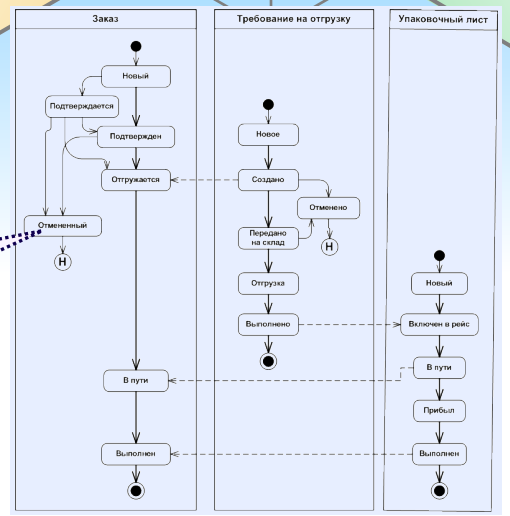
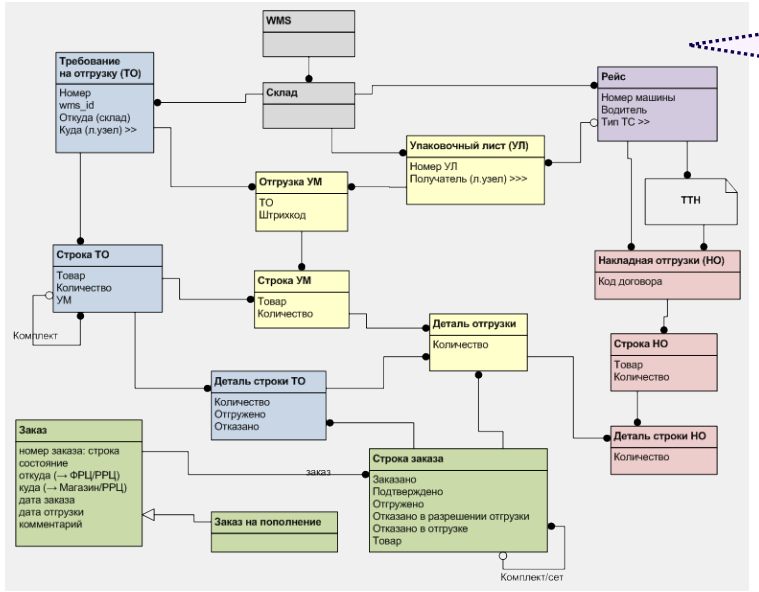


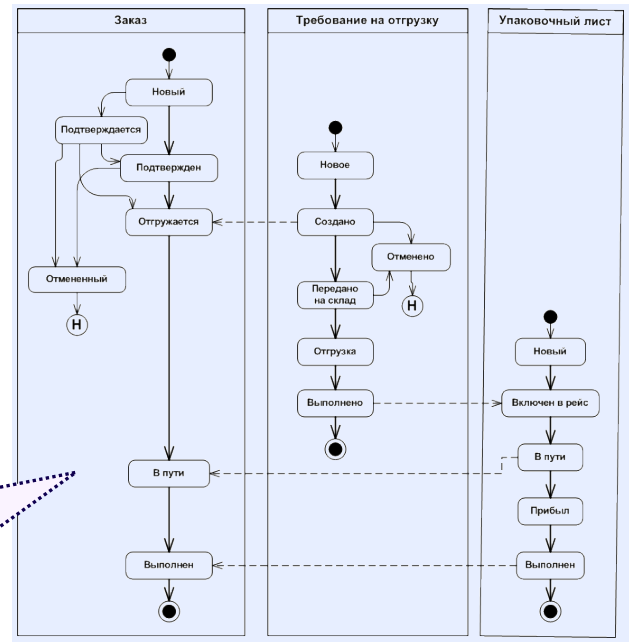
Диаграмма состояний



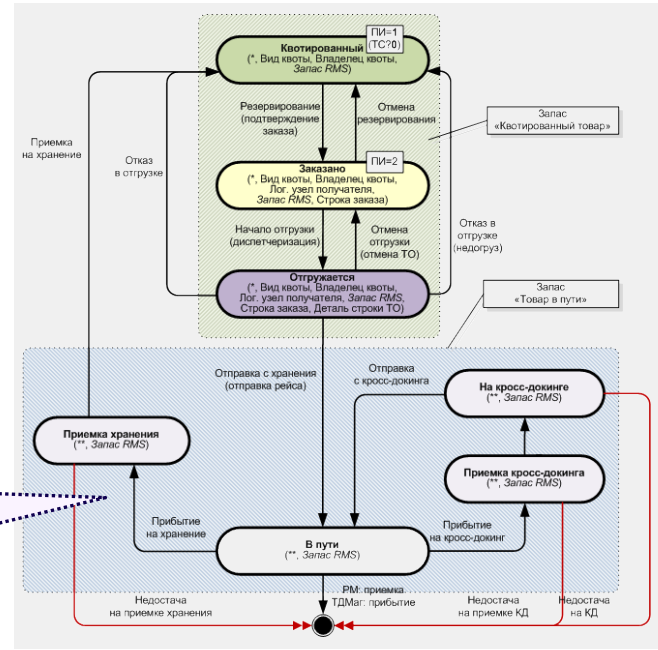
Пример: шаблон для корпоративного приложения



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

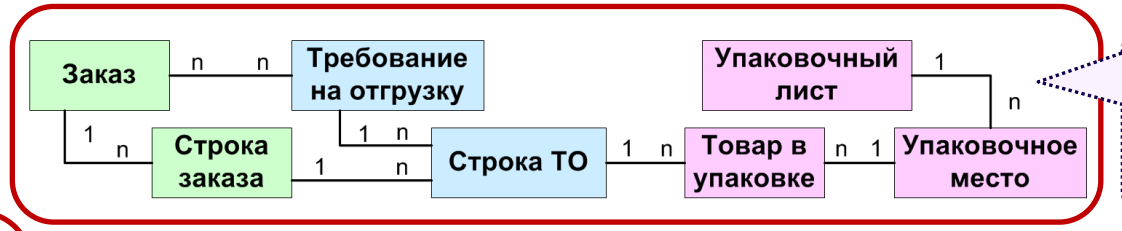


Потом документы исполняют – используем переходы документов между состояниями (State Entity)



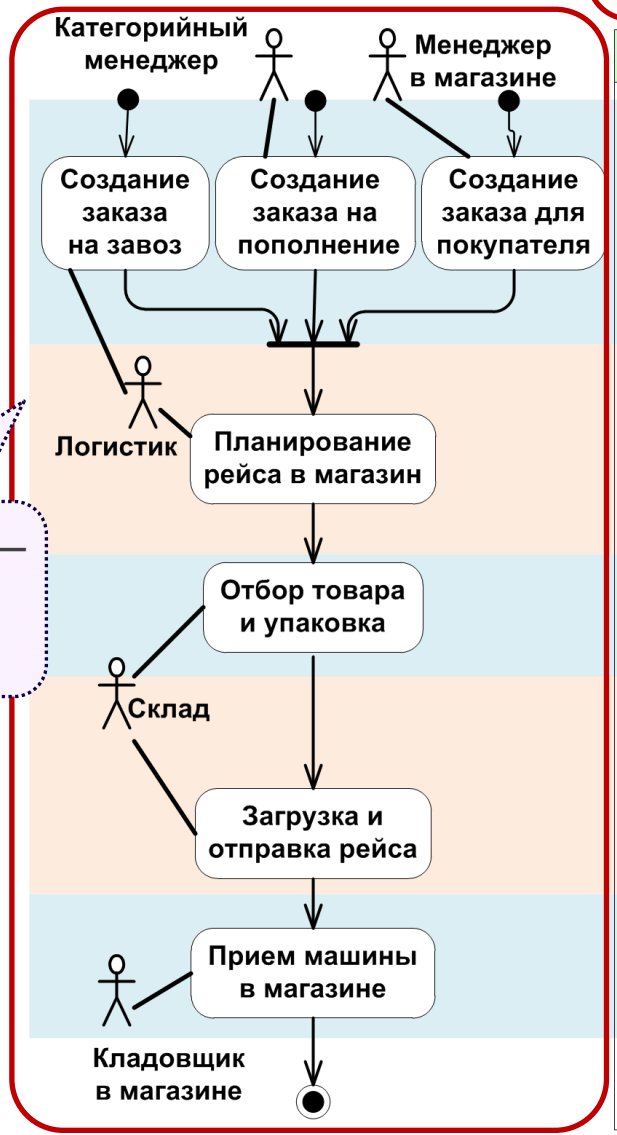
При этом меняются учетные данные, которые влияют на исполнение документов и отражаются в отчетах – используем **учетную модель**

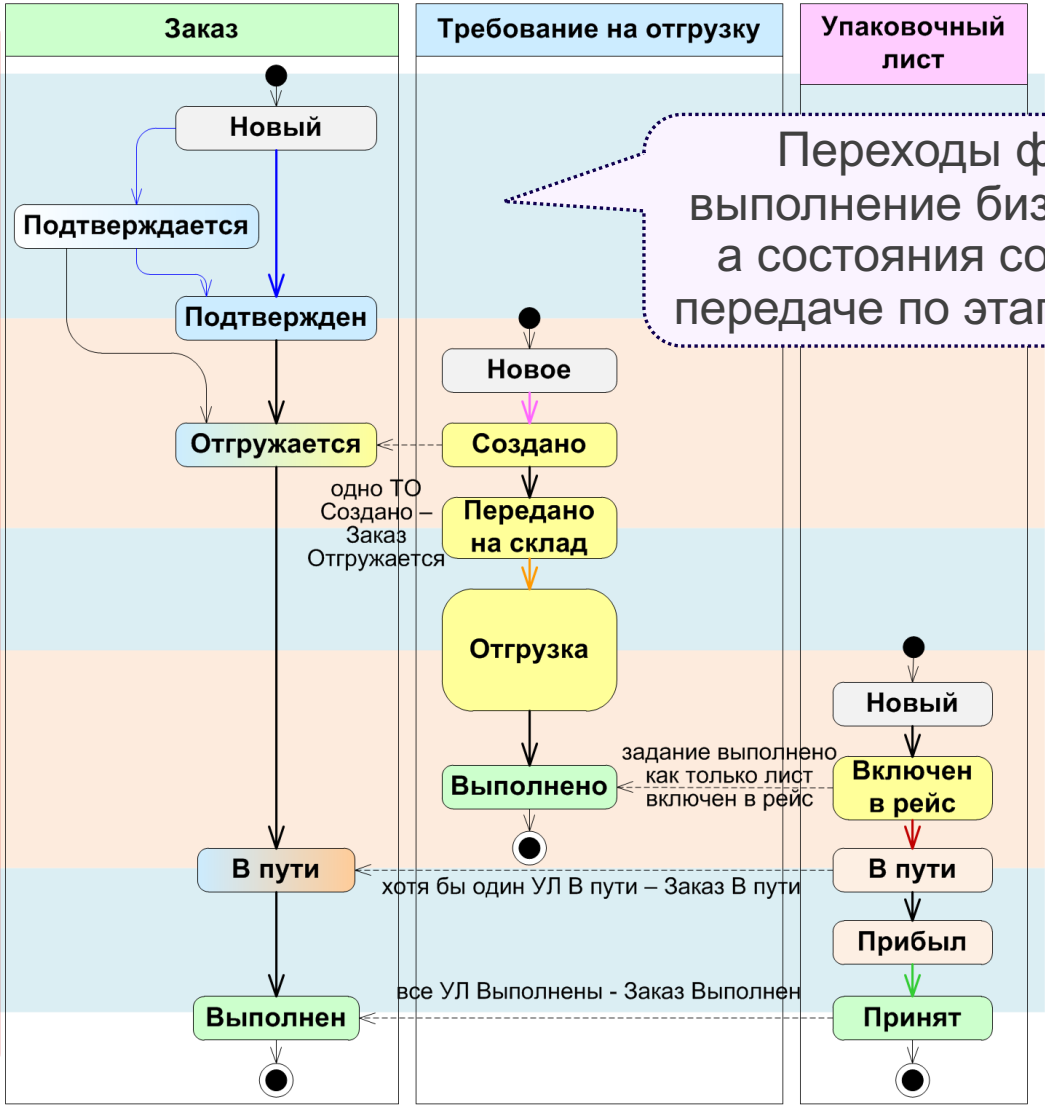
От бизнес-процесса к документам



Выделяем объекты-документы

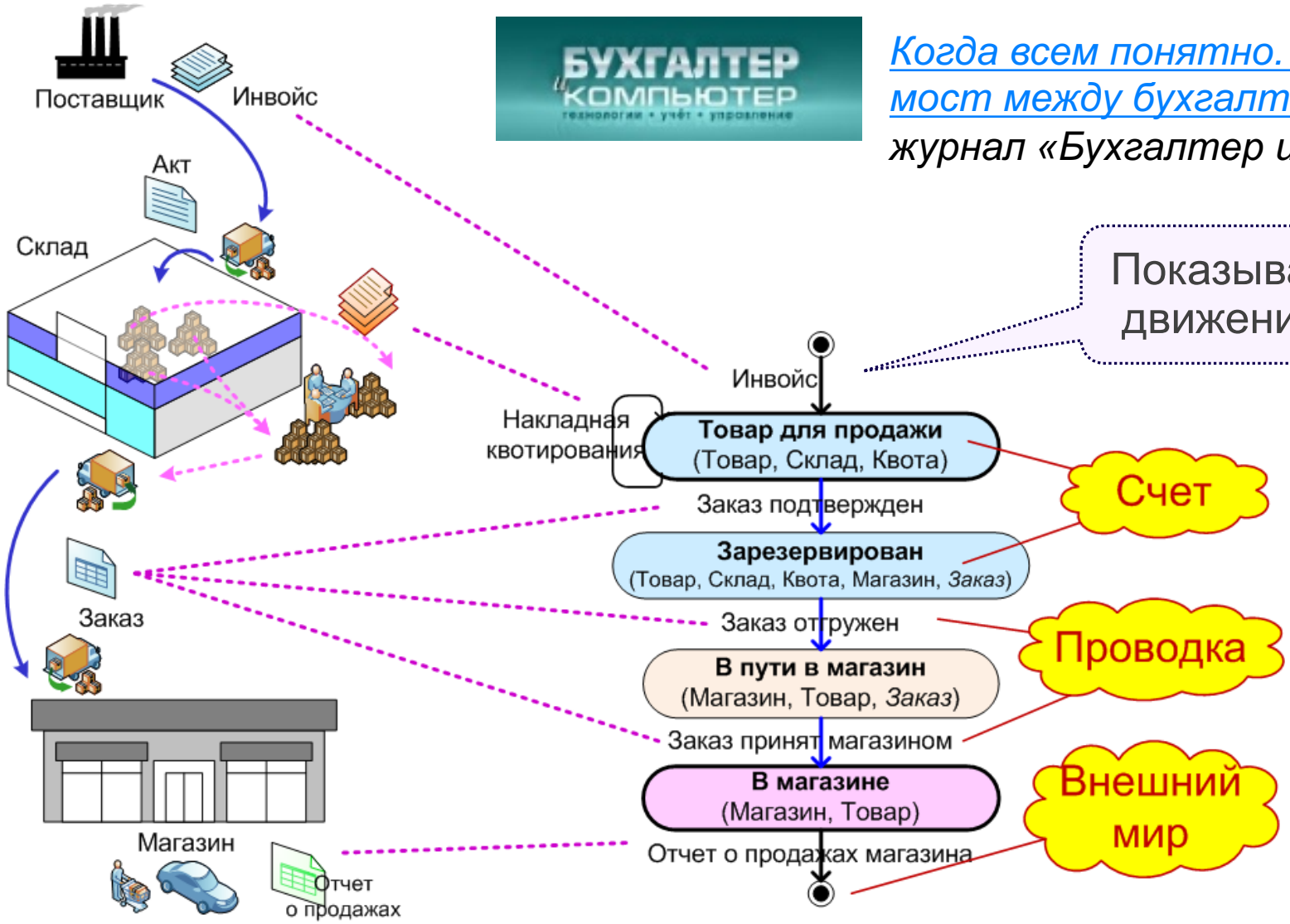
Бизнес-процесс – диаграмма активности





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

Пример: долг клиента

Пример: долг клиента

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

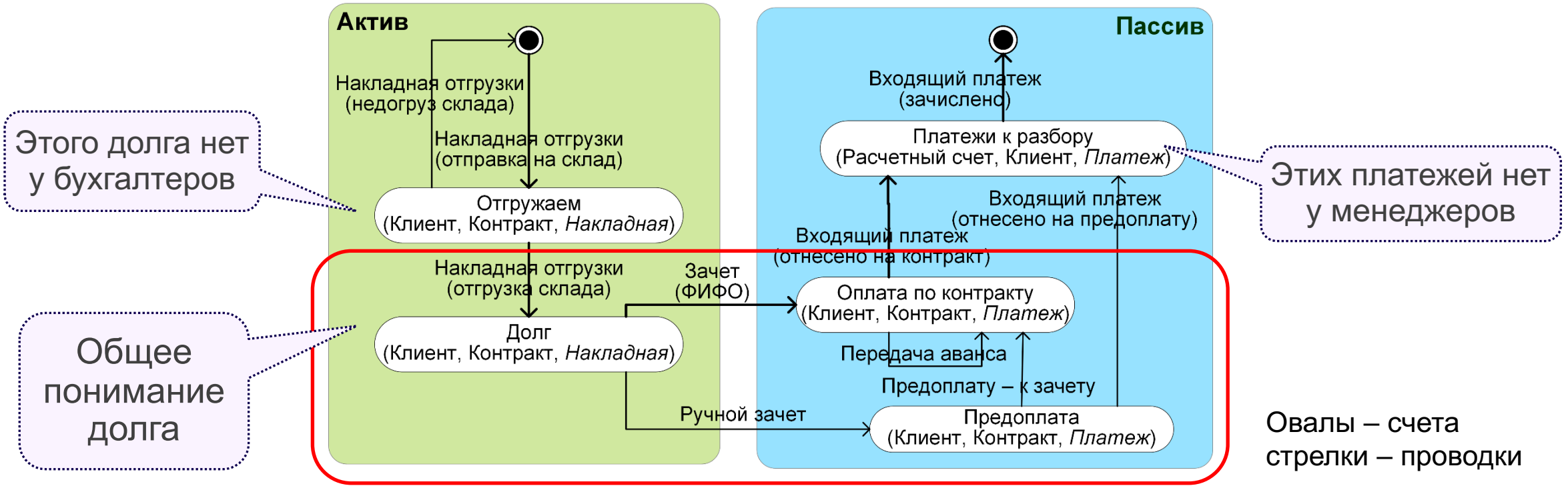
- ☹ Контроль правильности учета запаздывает
- ☹ Менеджеров не получается контролировать через бухгалтерию
- ☹ Имеются две разные суммы долга, что затрудняет принятие управленческих решений
- ☹ Для сверки с клиентом и решения проблем менеджеры должны вникать в ПБУ

Решение от DDD

- 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

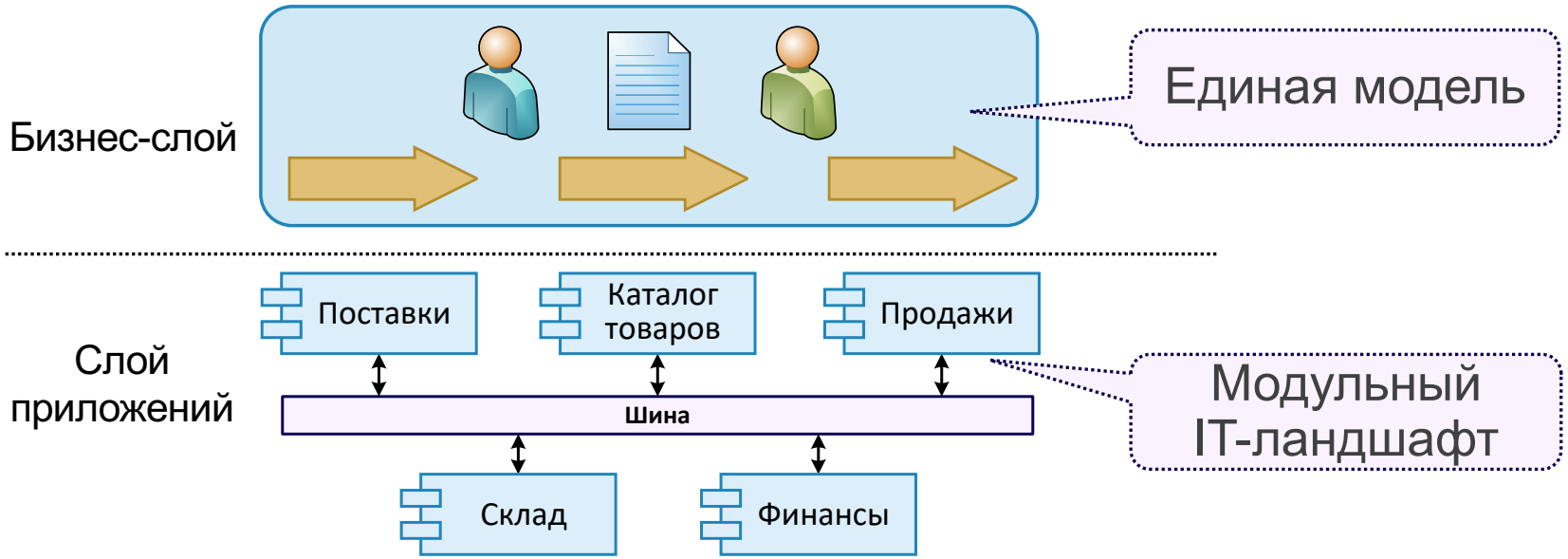
Модель долга клиента



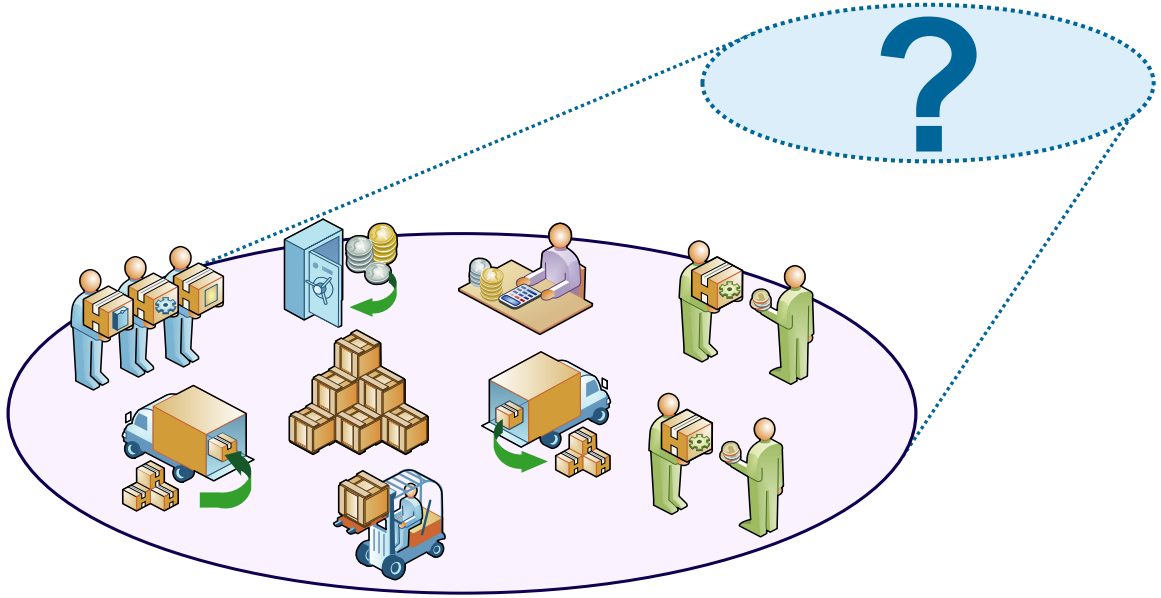
И бухгалтеры и менеджеры понимают данные системы, сверку с клиентом успешно выполняют менеджеры

Модульная модель предметной области

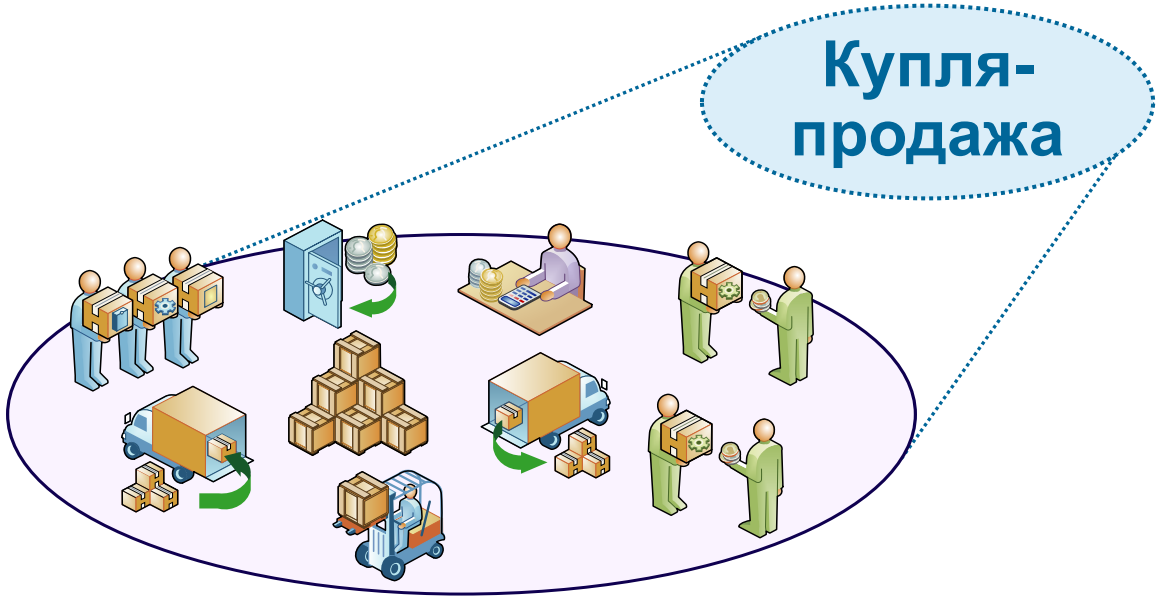
Приложения давно модульные, а **модель предметной области** часто хотят построить на общей непротиворечивой системе понятий



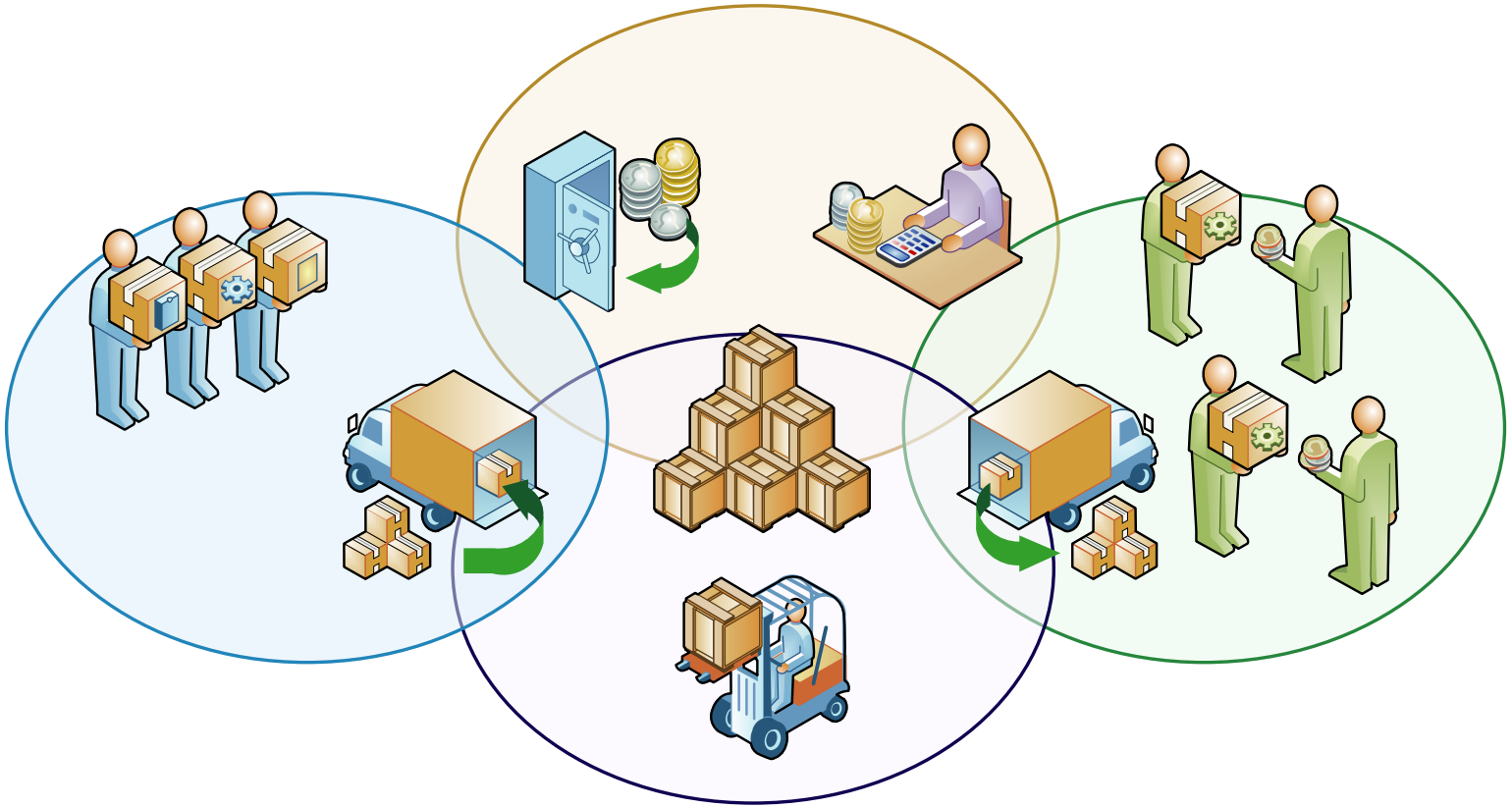
Это основано на убеждении, что существует **непротиворечивая и целостная картина мира**, которую мы можем не знать, но выясняем



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



Решение – **ограниченный контекст**



Как работать с контекстами?

- Предметную область следует разделить на ограниченные контексты **(bounded context)**
- Целостность удерживается за счет карты этих контекстов **(context map)**
- Имена (понятия) включаются в единый язык
- Для изоляции контекстов применяем **те же** подходы ООП, что и для разработки софта
- Для работы с похожими контекстами тоже применяем подходы ООП

Шаблоны работы с контекстами

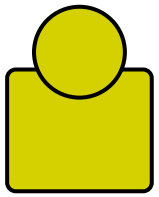
Кейс: товарные документы в торговле

Документы в оптовых продажах

Основной документ – накладная,
она фиксирует факт сделки и сумму за отгрузку

Документы в закупках

В закупках – похоже, но есть нюансы



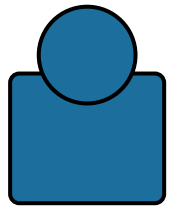
Поставщик

Заказ — Что мы хотели купить

Invoice — Что поставщик обещал прислать

Приходная накладная — Что принял наш склад

Счет-фактура — Для учета НДС (приходит с товаром)

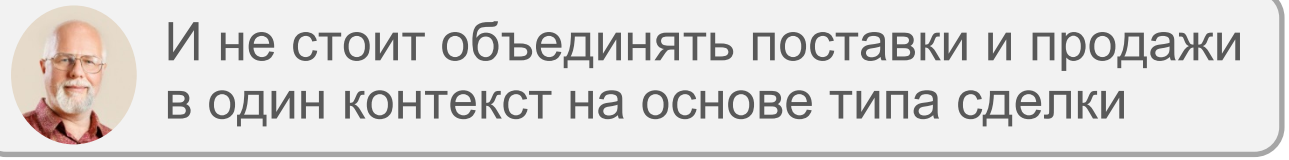


Компания

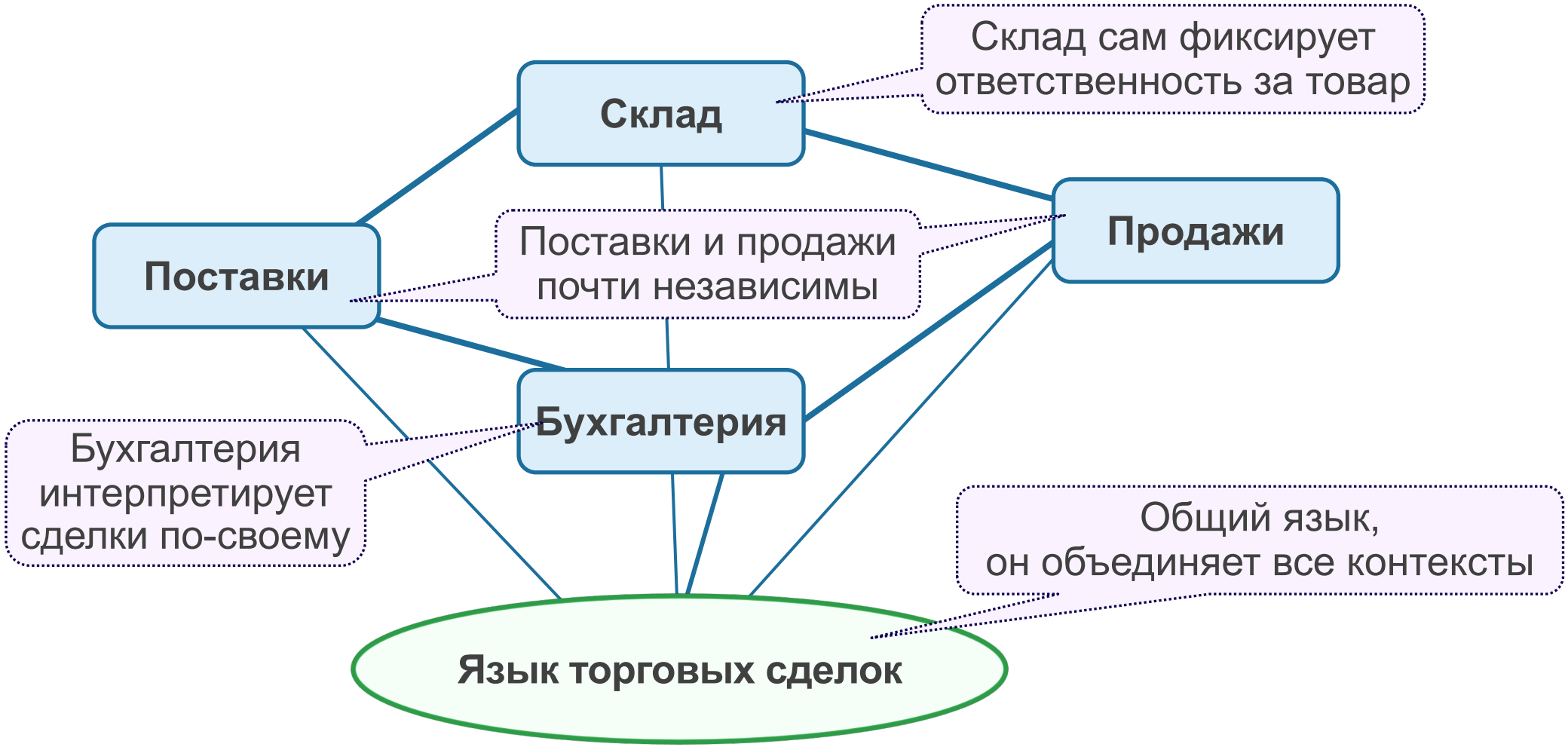
Основным документом стал Invoice: мы его оплачиваем и ожидаем товар, принимая на него заказы

Почему так по-разному?

- Сделка одинакова: купля-продажа
- Но компания занимает **разные позиции в сделке**
- Восприятие **всегда зависит от позиции**



Контексты товарных документов



Что же достигает DDD?

Единый язык + Единая модель:

- Надежная основу для общения всех участников проекта при принятии решений
- Успешно заменяют мелкую россыпь требований
- Позволяют эффективно развивать сложную систему

Это требует дополнительных усилий:

- Формирование единого языка
- Понимание разработчиками предметной области

По опыту, результат окупает усилия.

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

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