HappyDev'13
Больше чем конференция!

Омск. 7-8 декабря 2013

CUSTIS®

Domain Driven Design
Модель вместо требований

Максим Цепков
www.facebook.com/mtsepkov
www.custis.ru

7bits

О чем этот доклад?

- Есть разные способы работы с требованиями
- Выбор конкретного – зависит от проекта

- В сложных проектах уместна работа с моделями
- И DDD – наиболее эффективный способ для этого

DDD – ключ к построению сложных систем
и их развитию вслед за потребностями бизнеса

Схема доклада

Теория

DDD – единый язык проекта и одна модель системы

- Модели были давно, но две: бизнес-область и система
- Единый язык проекта создает общее поле понятий
- И позволяет работать с одной, общей моделью

Практика

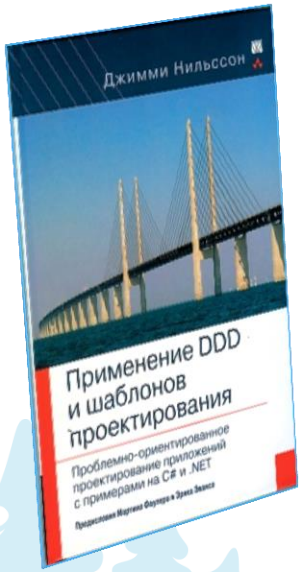
- Единый язык в конкретных примерах
- DDD в корпоративных приложениях

Заключение

Начнем с теории

DDD – как оно начиналось

- 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!)

Основная идея DDD

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

А где здесь ООП ?

ООП – это парадигма моделирования

Парадигма моделирования определяет

- Элементы языка
- Способ их соединения в сложные конструкции
- Визуальный образ для представления
- Способ отражения модели в код

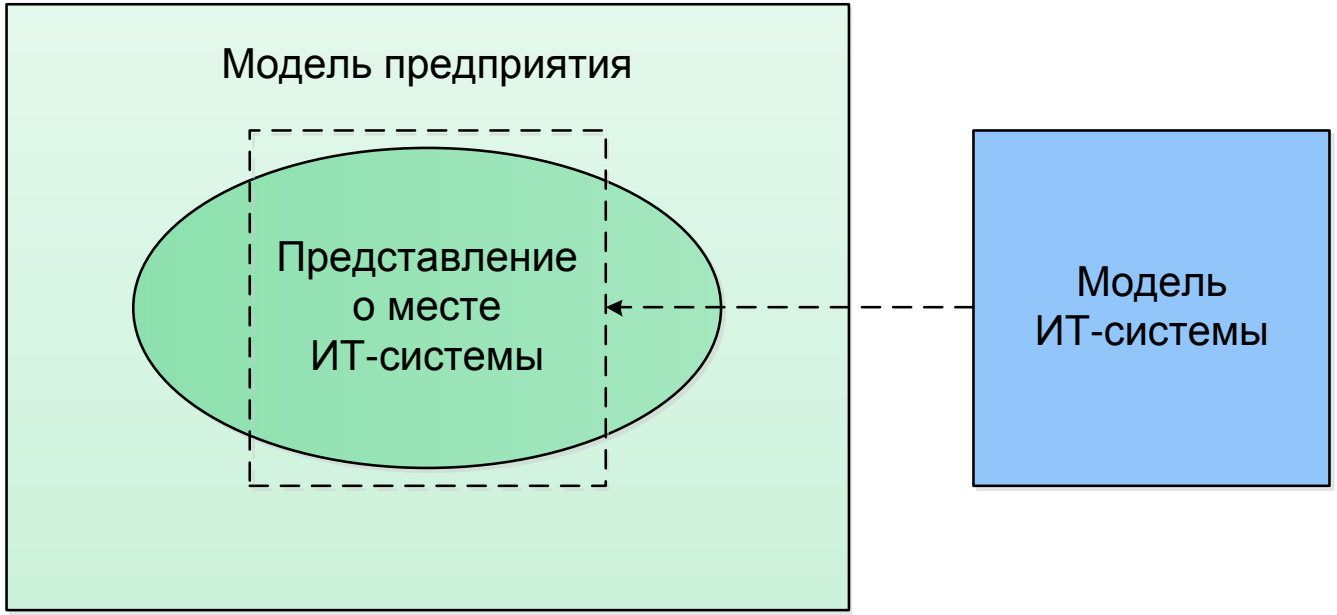
Объекты с атрибутами и методами

Диаграмма классов и другие UML-диаграммы

Типы, соответствующие бизнес-объектам

Я сосредоточусь на разработке модели, а не на ее реализации

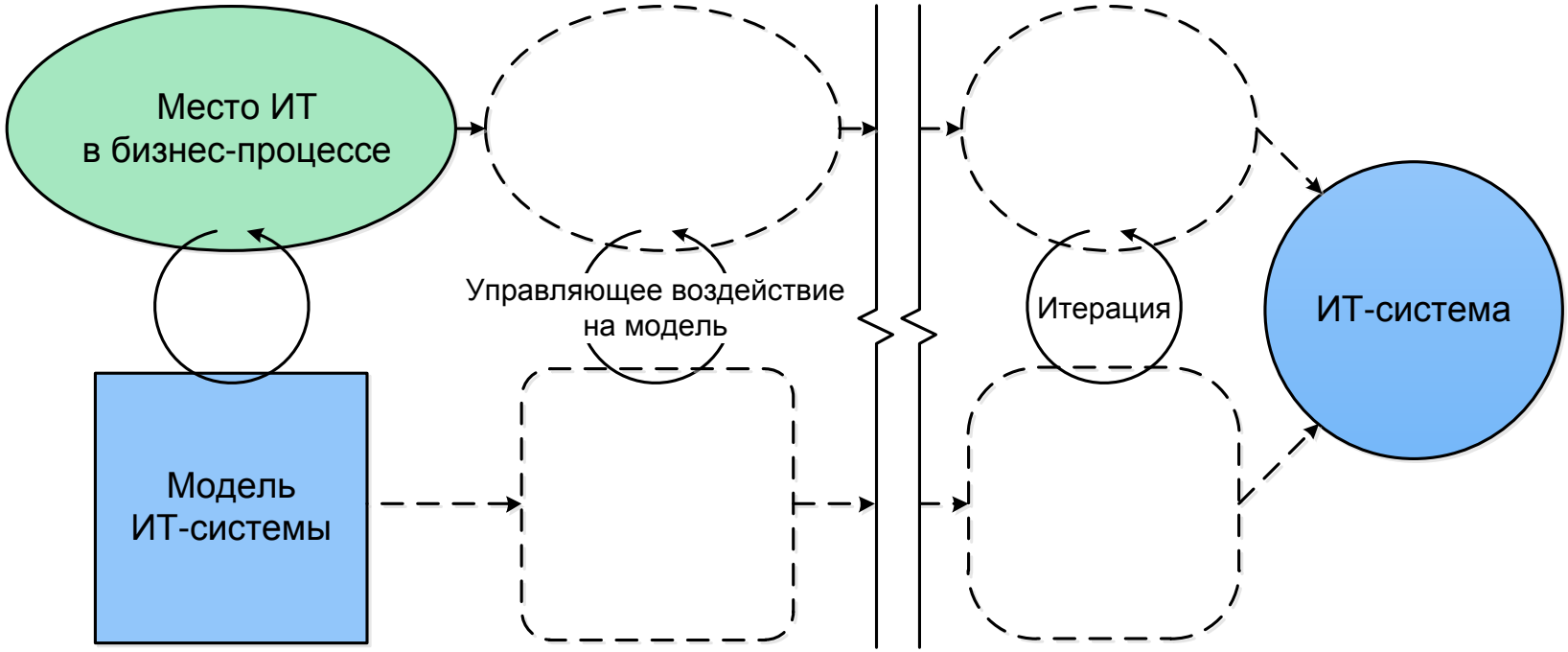
Зачем нужен единый язык?



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

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

Итерационное развитие модели



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

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

Что мы достигаем?

- ☺ Верификация постановок бизнес-специалистами
- ☺ Общее понимание требований на стороне бизнеса
- ☺ Обсуждение модели бизнесом и IT, 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 IT без квалифицированных переводчиков-аналитиков

Пример:
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?

- Шаблоны обладают разной гибкостью и сложностью
- Для выбора нужно понимать требуемый баланс между гибкостью и сложностью 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

- 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 IT из-за сложности

Решение в рамках DDD

- 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?

- Для решения модель надо описать понятно 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

- ☺ 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

DDD для корпоративных приложений

Часть 1 – общая схема

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

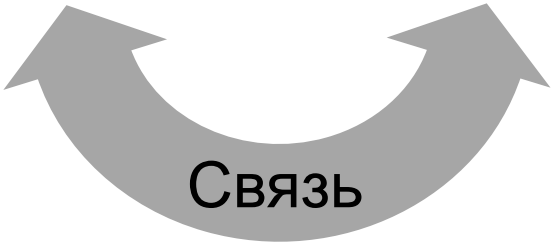
Жизненный цикл документа

Учет – не объектная модель

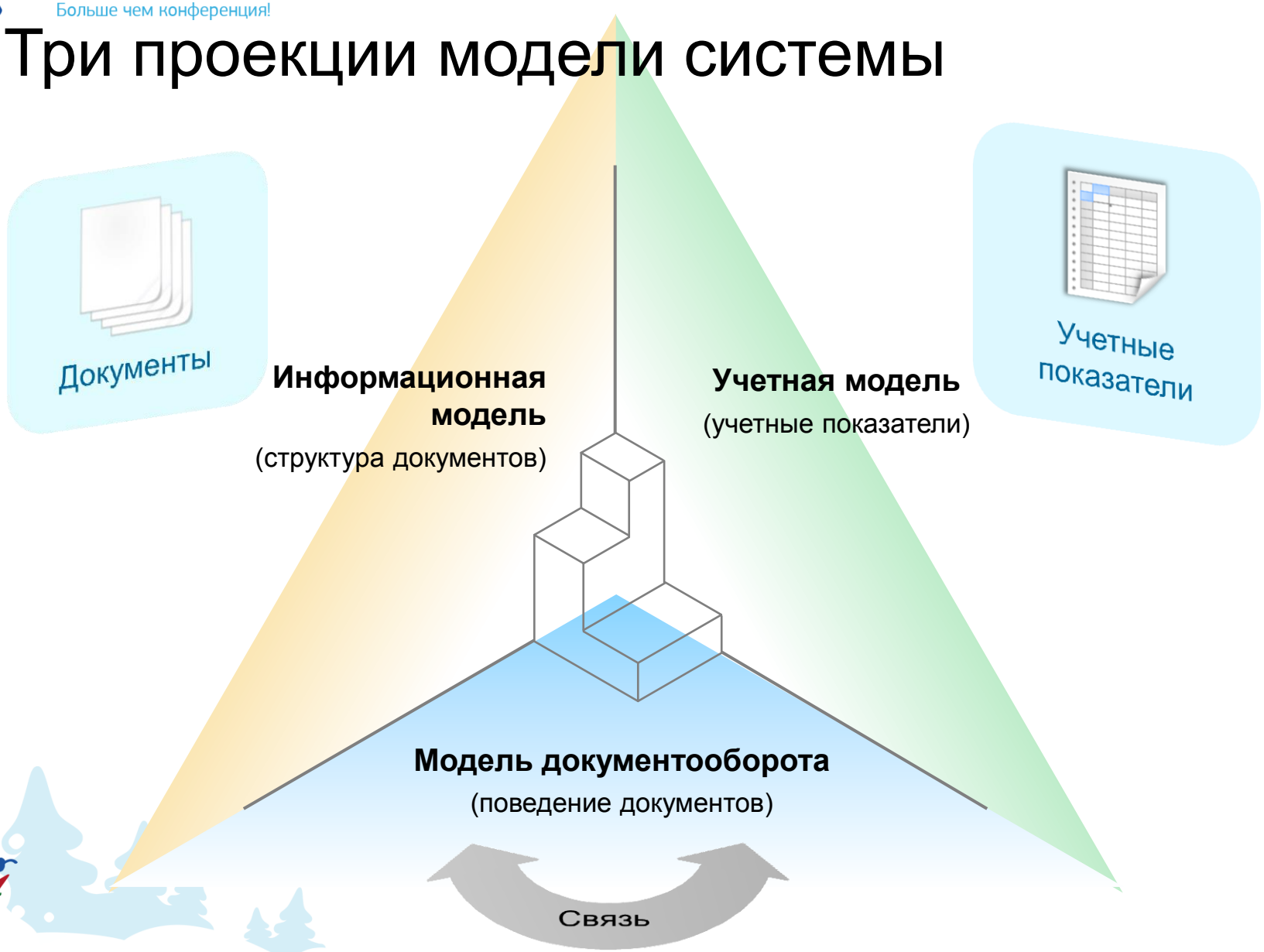
Единая модель системы

Документы

Учетные показатели



Три проекции модели системы



Диаграммы для проекций

Документы и справочники – диаграмма классов

Учет – диаграммы учета

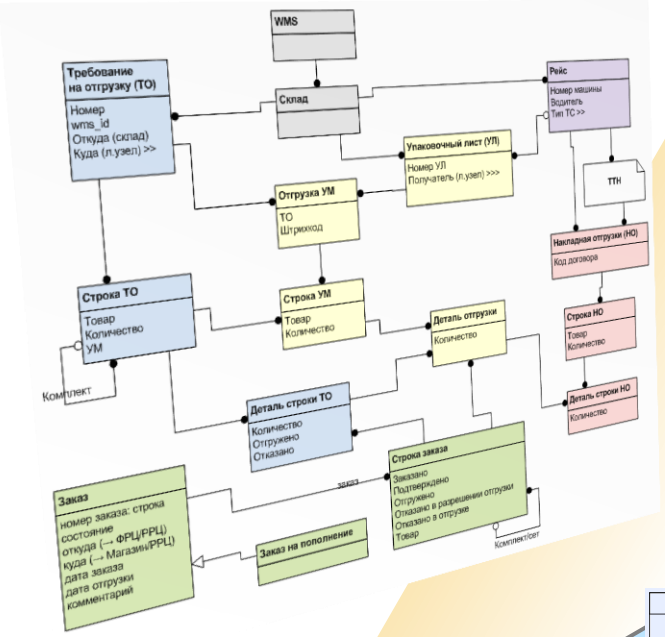
Информационная модель
(структура документов)

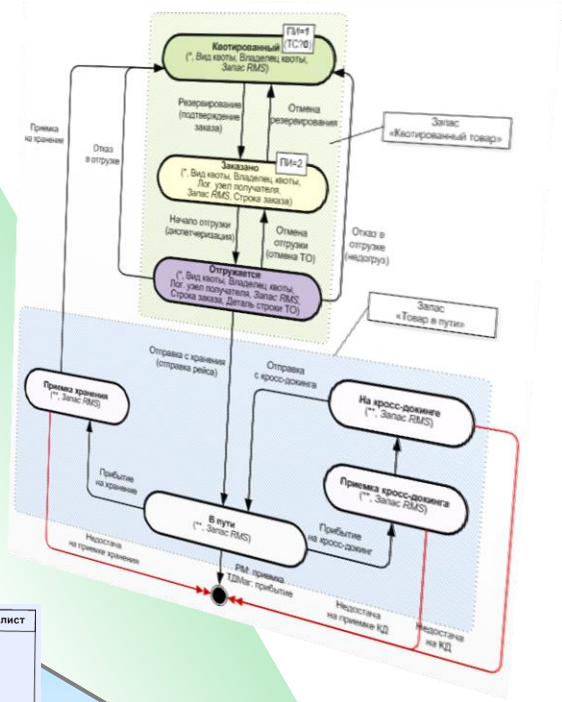
Учетная модель
(учетные показатели)

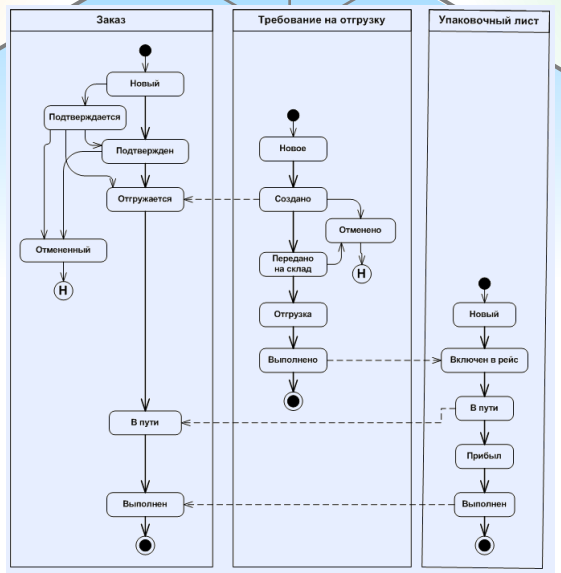
Документооборот – диаграмма состояний

Модель документооборота
(поведение документов)

Диаграммы для проекций







DDD для корпоративных приложений

Часть 2 - Учетная модель

Учетная модель – не объектная

Сложность объектного представления учета:

- Нет идентификации единичного объекта.
- Работа идет с показателями, текущее значение которых меняется.
- Изменение числового значения может менять состояние с точки зрения принятия бизнес-решения.
- Часто интерес представляют агрегаты, а не отдельные значения.

Представление учета оказалось за рамками UML. Для него не придумано эффективного способа.

Учетная модель

Для представления учетной модели мы придумали Диаграммы учета.

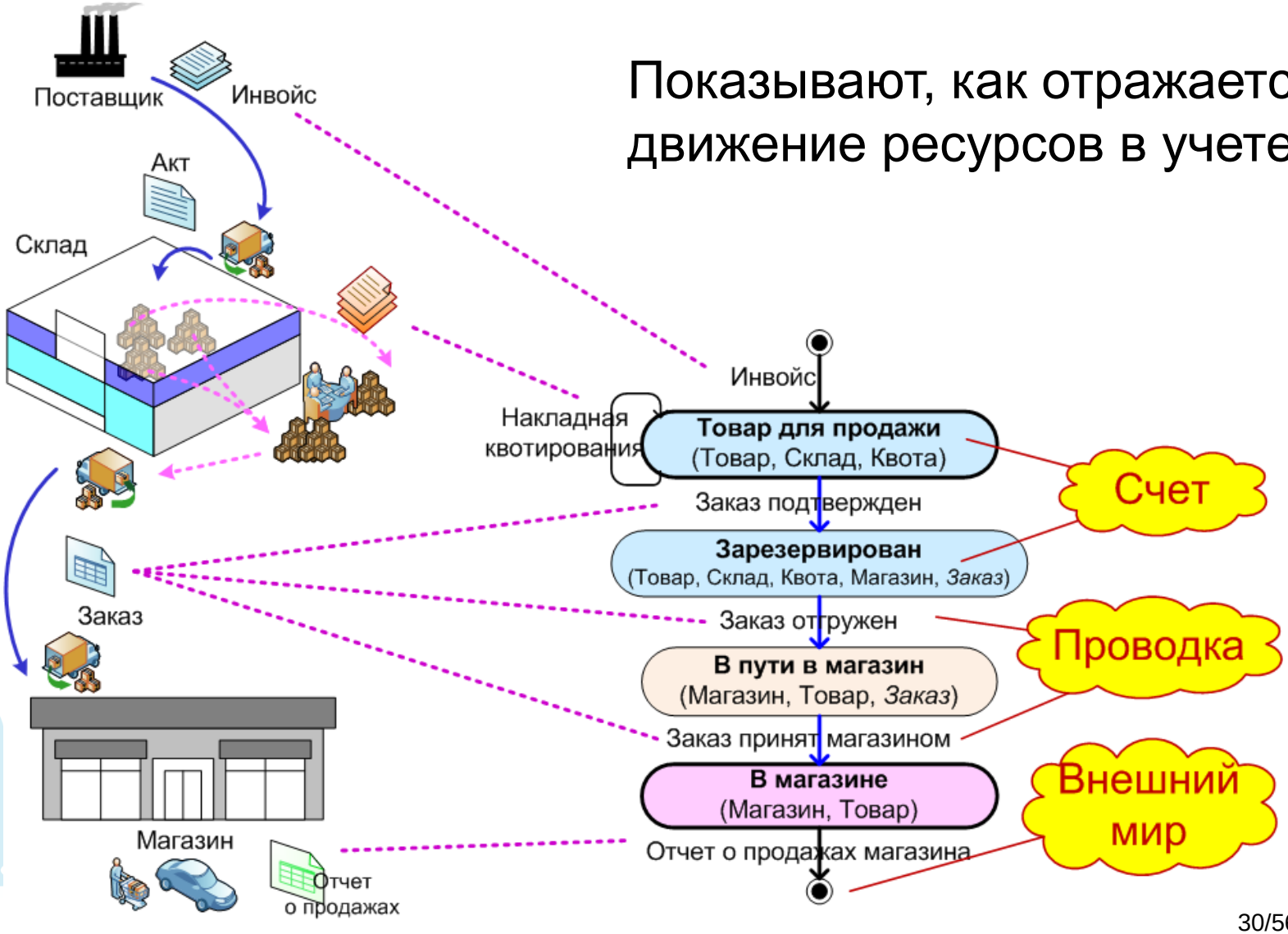
Они показывают элементы учета:

- какие есть синтетические счета и их аналитику
- как проводки перемещают ресурсы по синтетическим счетам
- с какими событиями связано исполнение проводок

Статья
«Диаграммы учета: мост между бухгалтером и разработчиком»
Журнал «Бухгалтер и компьютер» 5-2011 http://lib.custis.ru/Когда_всем_понятно

Диаграммы учета

Показывают, как отражается движение ресурсов в учете



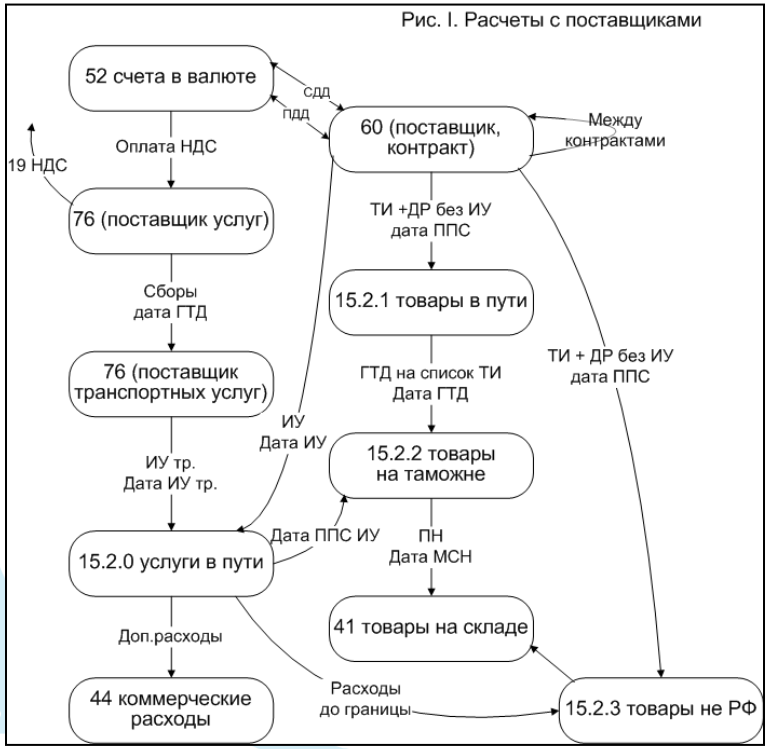
Как живет учетная модель?

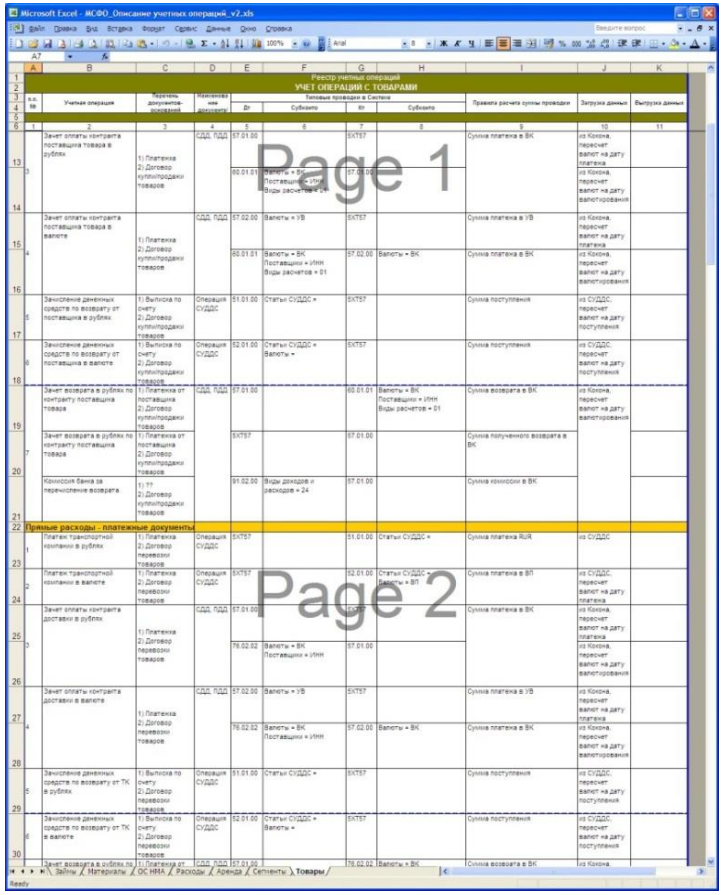
- Уточнения
  - определяем аналитики
  - определяем источники событий
- Изменения
  - добавляем новые участки учета
  - добавляем транзитные счета

Диаграммы учета – для бизнеса

Бухгалтеры могут применять диаграммы учета для разработки учетной политики.

Они нагляднее, чем Excel.





Отражение учетной модели в код

Модель позволяет построить шаблон для реализации.

Есть Patterns for Accounting Мартина Фаулера – отражение учета в объектную реализацию:

- учетные счета и проводки;
- источник проводок – события.

Наш метод

У нас – более развитая реализация:

- хранение аналитических признаков на счетах и проводках;
- ведение остатков и оборотов учетных счетов;
- ведение детальных и агрегированных показателей.

Есть собственный язык описания – GL-XML.

Представить реализацию бизнесу

- Взгляд бизнеса:

  есть журнал хозяйственных операций, возникающих при обработке и проведении документов.

- Реализация:

  проводки создаются по событиям (Event Sourcing, Фаулер), которые возникают в методах документов.

Для прозрачной модели это должно совпадать:
учетные события – суть хозяйственные операции.

DDD для корпоративных приложений

Часть 3 – Документы и Справочники

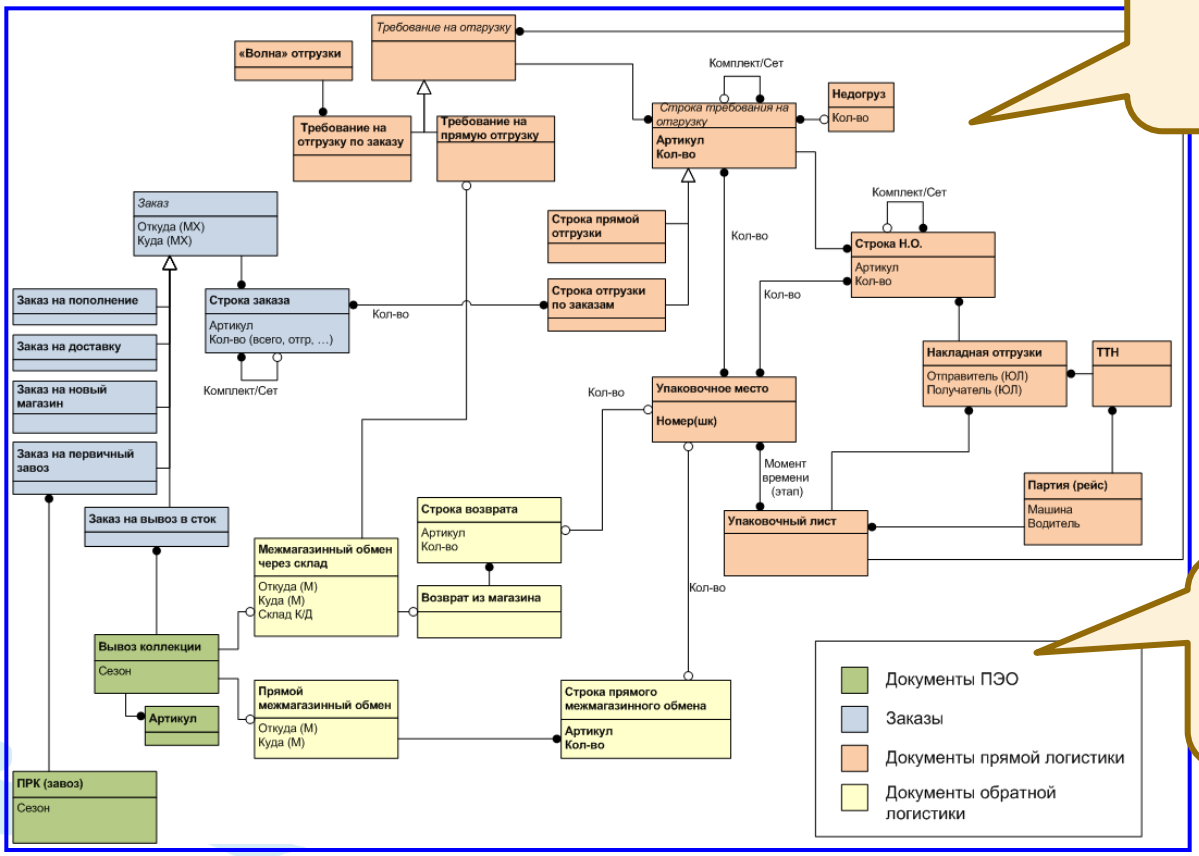
Хорошо работает объектная модель

- Объекты с атрибутами и методами –
  элементы единого языка для предметной области и способ их соединения в сложные конструкции
- Диаграмма классов и другие диаграммы UML –
  визуальный образ для наглядного представления
- Объекты в программе –
  способ отражения модели в реализацию

Модель должен понимать заказчик

Классы соответствуют бизнес-объектам – заказчик видит знакомые названия

Представляем диаграммой классов



Используем цветовые выделения

DDD для корпоративных приложений

Часть 4. Связь документа с учетом

Обобщенный документооборот

- Документ обрабатывается в несколько этапов.
- На каждом этапе определенные сотрудники могут совершать определенные действия.
- Для передачи на следующий этап должны выполняться определенные условия.

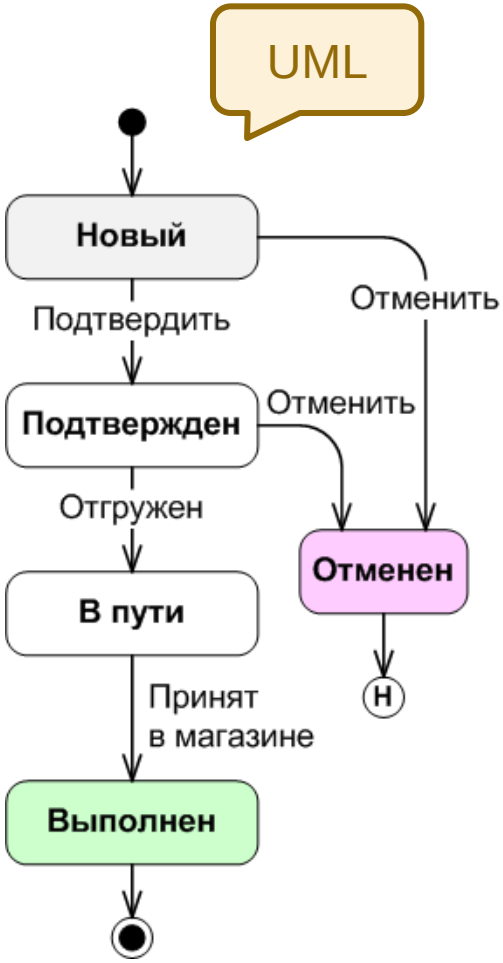
Модель для документооборота

- Документу приписываем состояние.
- Состояние определяет этап документооборота:
  - какие действия можно совершать над документом;
  - кто отвечает за обработку документа;
  - кто имеет права на совершение тех или иных действий.
- Возможные изменения состояний документа образуют граф переходов.

Используем шаблон State Entity.

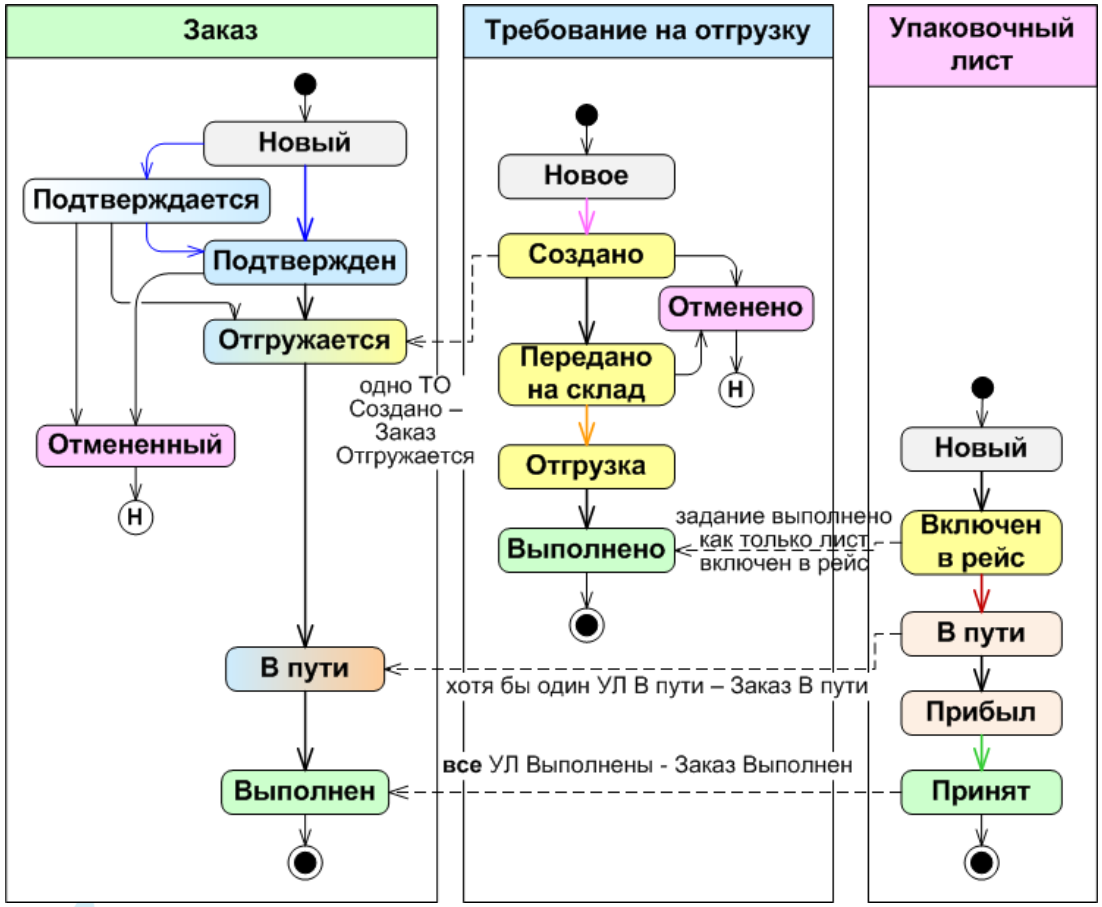
Язык модели

- Документ – объект предметной области.
- Действие над ним – вызов его метода.
- Среди всех методов выделяем переходы и связываем их с состояниями.
- Граф состояний – State machine diagram.



Язык ООП «с расширениями».
Названия состояний и переходов – на языке бизнеса.

Наглядно представляет сложный документооборот



Еще пример: Долг клиента

Задача

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 – опаздывает
- ☹ 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

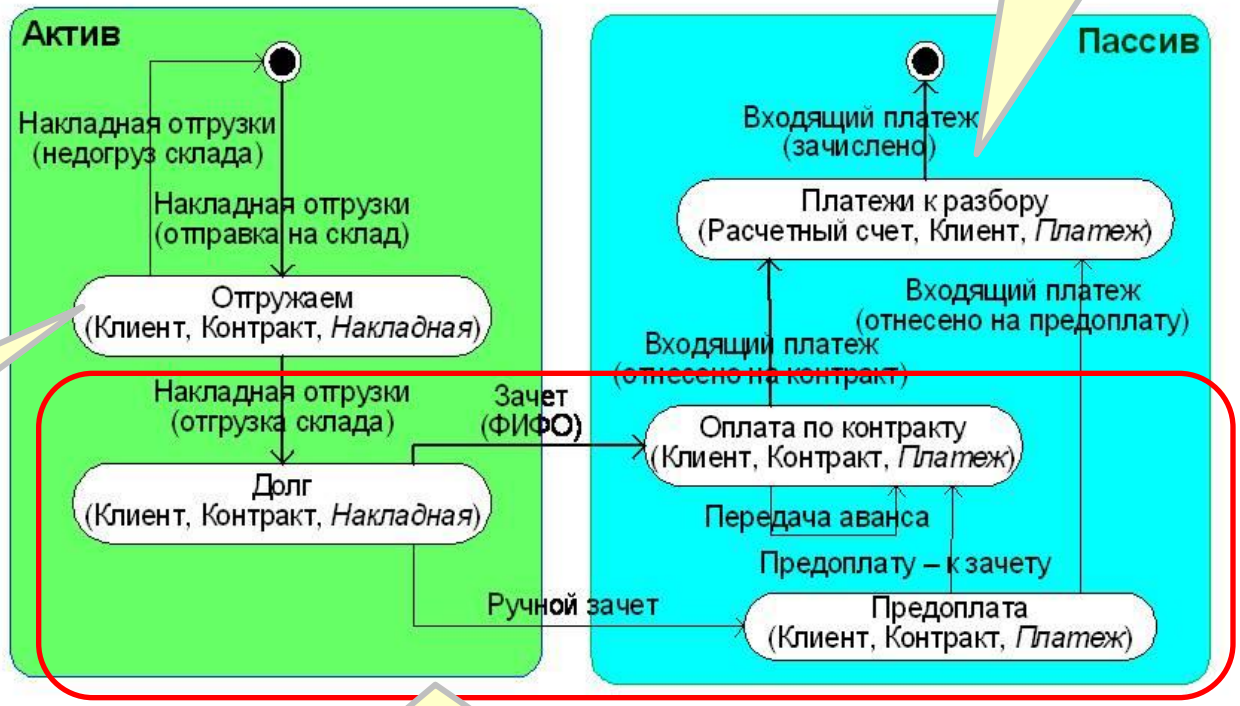
☺ Результат – общий взгляд на предметную область у бизнес-специалистов, описанный в модели, которая реализуется разработчиками

Модель долга клиента

Сверку с клиентом успешно выполняют менеджеры

Этих платежей **нет** у менеджеров

Этого долга **нет** у бухгалтеров



«Общее» понимание долга

Овалы – счета
стрелки – проводки

Результат

- ☺ Согласовано понимание предметной области у различных бизнес-специалистов
- ☺ Реализована модель, отвечающая интересам всех заинтересованных сторон проекта

Практики проектирования – на этап бизнес-анализа

Частные практики

- Стратегии
- Политики
- Выделение общих объектов
- Абстракция через интерфейсы

Технические практики наполняются бизнес-смыслом и служат для общения с Заказчиком

Контексты

Перенос практик декомпозиции на бизнес-анализ

- Понятие ограниченных контекстов
- Различные варианты выделение общего
  - Общее ядро
  - Абстрактное ядро
  - Подключаемые компоненты
  - Крупноблочная структура
  - Уровни разделения обязанностей
- Изоляция и карта контекстов

Заключение

Что же обеспечивает DDD?

Единый язык + Единая модель:

- обеспечивают надежную 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.

DDD позволяет успешно создавать сложные проектные решения и развивать их

Спасибо!

Вопросы?

Другие доклады lib.custis.ru/MaksTsepkov

Максим Цепков www.facebook.com/mtsepkov