

Модель системы – архитектура для Agile-разработки

Ag;)eDays'11

Докладчик:

Максим Цепков (M.Tsepkov@custis.ru)

www.CUSTIS.ru

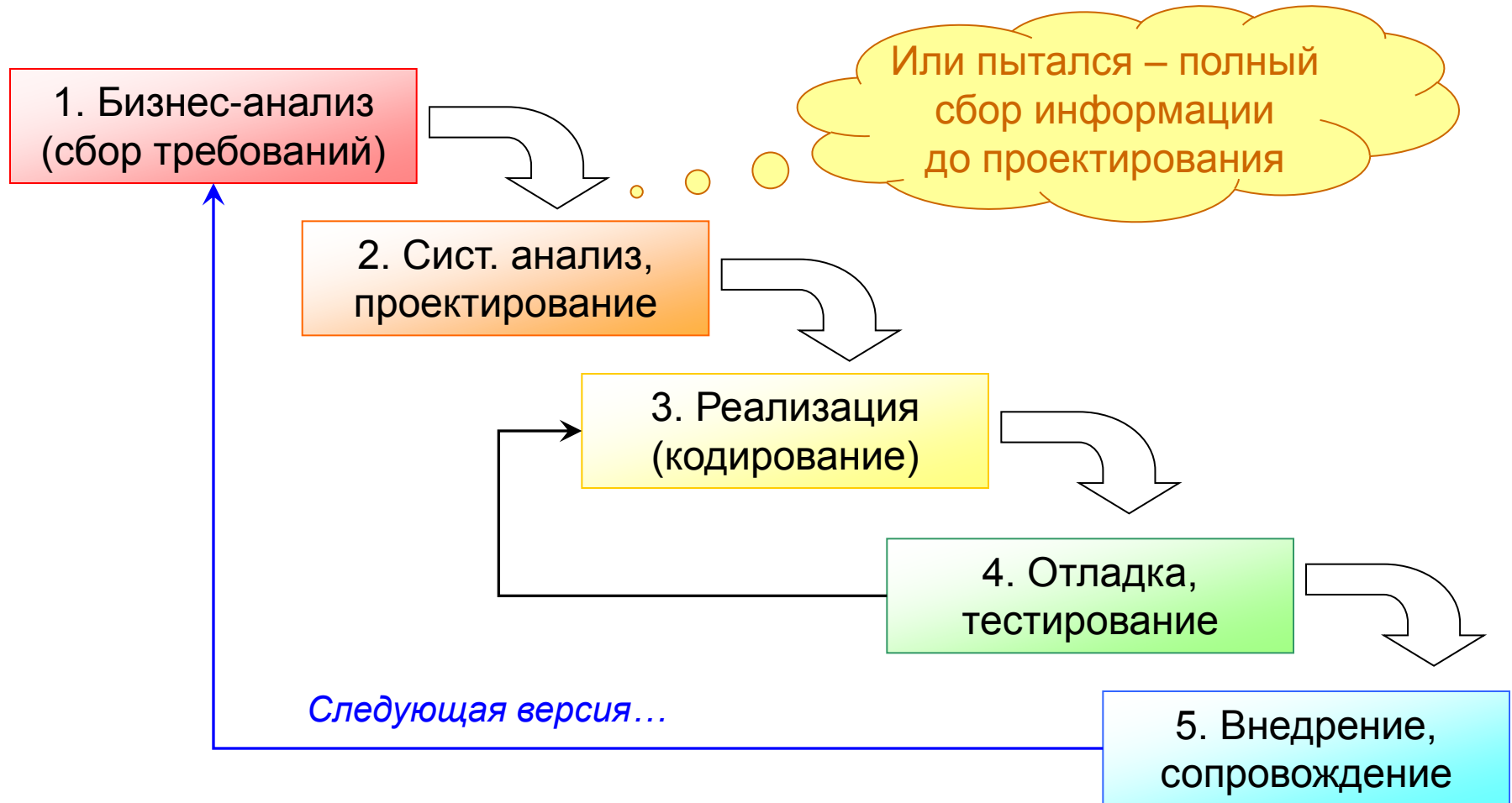
Проблема

Как при итеративной разработке
обеспечивать **целостность**
архитектуры системы

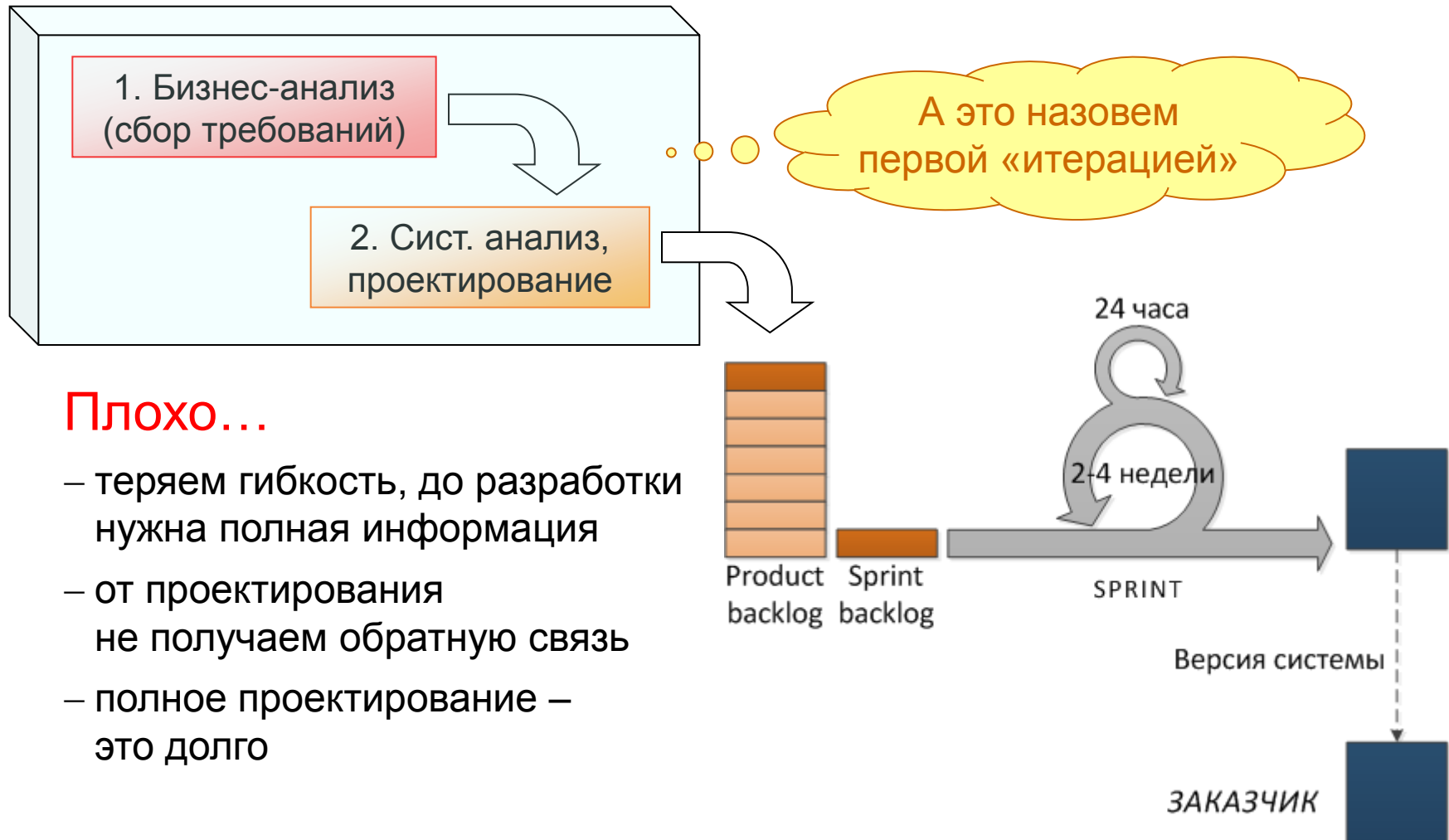


КРИТИЧНО
для БОЛЬШИХ
(многолетних)
проектов

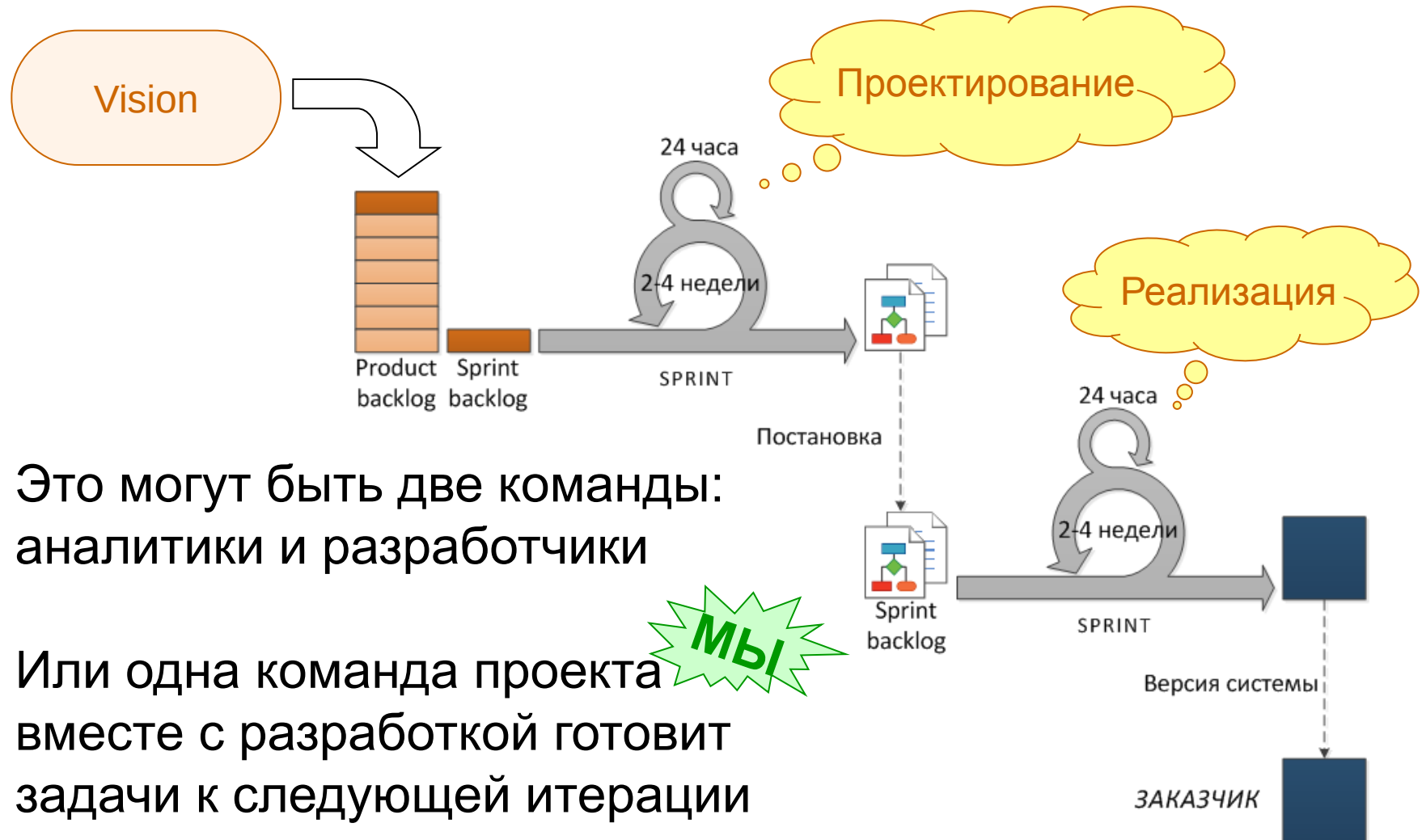
Водопад целостность обеспечивал...



А если сохранить процедуру проектирования в Agile?



Лучше – запустить два конвейера



Это могут быть две команды:
аналитики и разработчики

Или одна команда проекта
вместе с разработкой готовит
задачи к следующей итерации

Как это сделать?

Mainstream сейчас – **DDD**

Классические описания
Эрик Эванс, Джимми Нильсон

для знакомства – смотри материалы, ссылки, слайды и видеозапись
Тренинга Андрея Бибичева <http://lib.custis.ru/ddd-training>



Что предлагает DDD?

Архитектура на основе модели предметной области

Целостность

Модель можно строить итеративно

Итерации

Модель должны понимать и заказчик и разработчик

Единый язык

Как строить модель?

В качестве общего метода – ООП

- объектная модель – хорошая основа разработки
- можно строить итеративно, начав с основных классов
- компактное описание в виде диаграммой классов
- понятный всем IT-специалистам подход
- можно оценивать объем изменений

Объектной модели может не хватать,
тогда Эванс предлагает искать способы дополнения

Почему объектной модели не хватает?

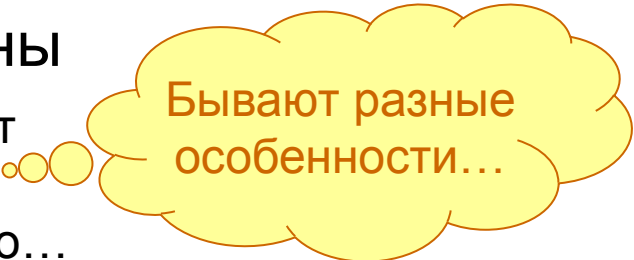
Недостаточно для взаимопонимания

Разработчику достаточно диаграммы классов,
чтобы многое сказать о системе,
а заказчику этого не хватает...

Заказчик может мыслить в не объектной модели

Недостаточно для целостной картины

Например, диаграмма классов не отражает
организацию учета и документооборота,
а для учетных приложений это очень важно...



Бывают разные
особенности...

Задача может решаться в не объектной модели

Надо дополнить объектную модель

Какие требования?

Возможность итеративного проектирования

Единый язык описания –
понятен и заказчику и разработчикам

Цельный и компактный взгляд на систему

Итерации

Единый язык

Желателен
визуальный
образ

Целостность



Как дополнить модель?

Общего решения для всех — не знаем

Но расскажем о том, чем пользуемся сами



Ищем метафору системы

Метафора – лаконичная концепция,
которая много говорит о системе

Визуальный образ – хорошая метафора
Он дает цельное представление о системе
И обеспечивает взаимопонимание

Целостность

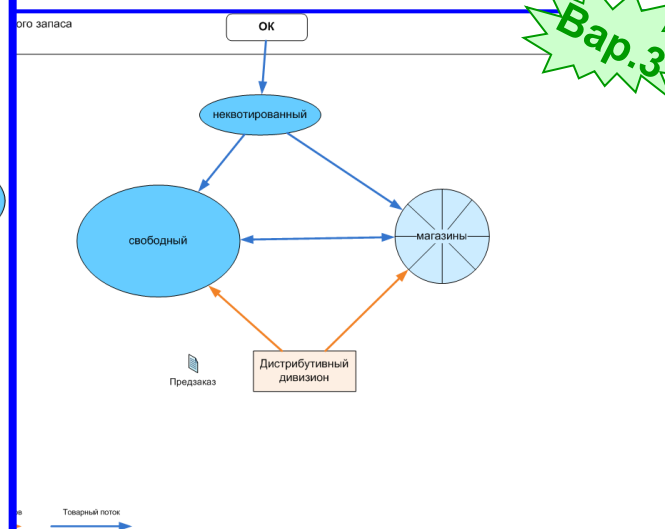
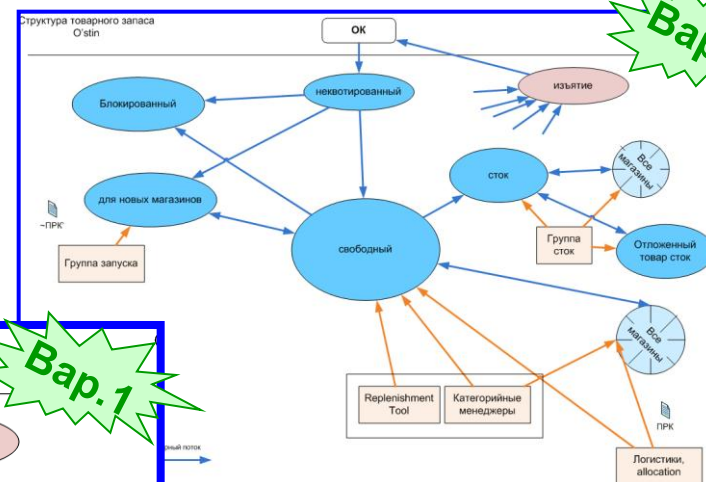
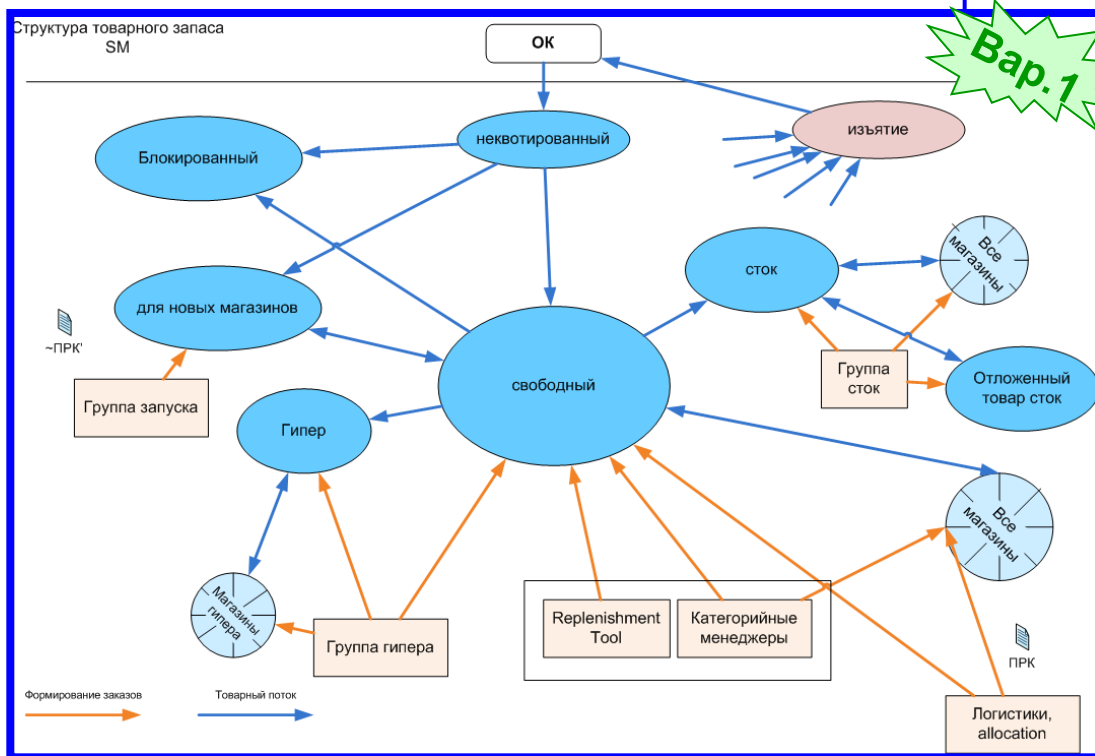
Единый язык

Поиск метафоры начинается с vision проекта
Визуальная основа – стандартна или придумана

Используем типовые образы

Пример:
Деление товара

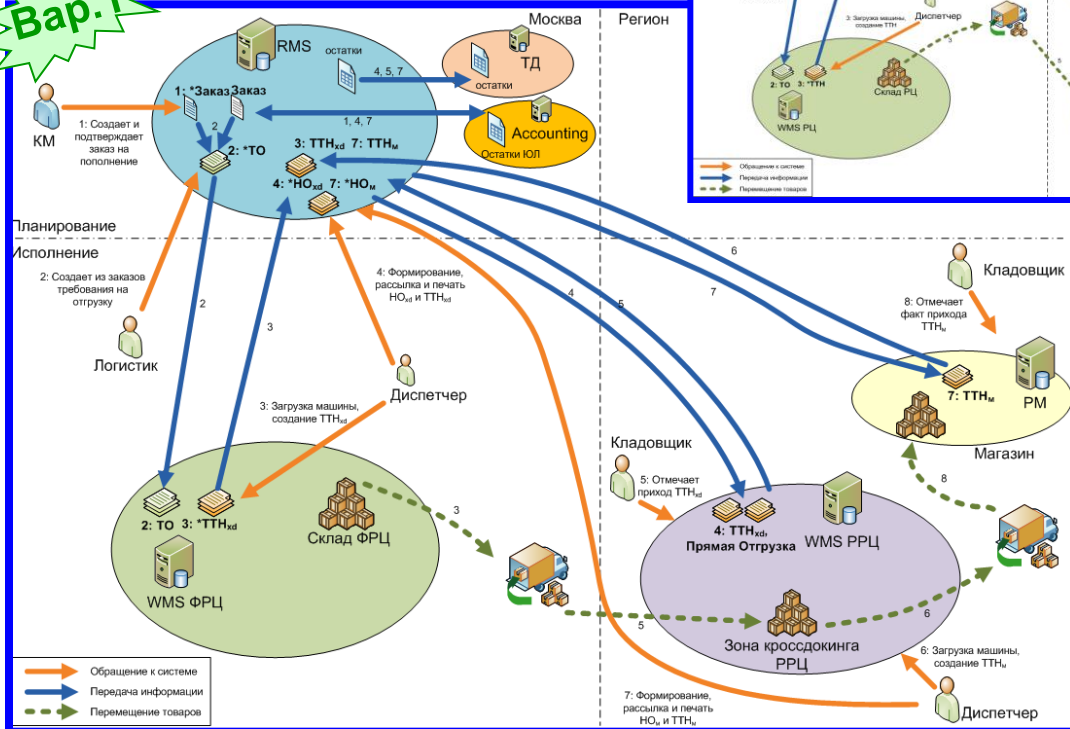
Варианты бизнес-процессов
в простой нотации
Схемы понимаются на лету



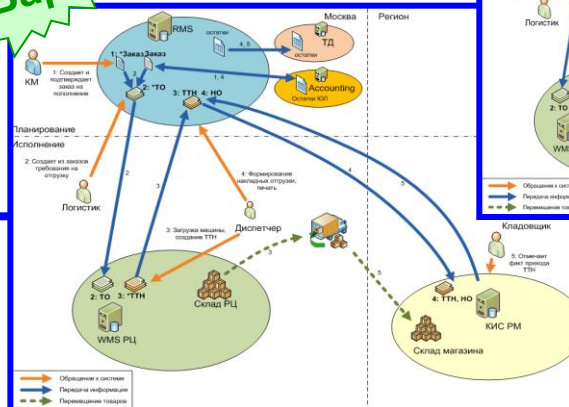
Идем от бизнес-процессов

Пример:
снабжение магазинов

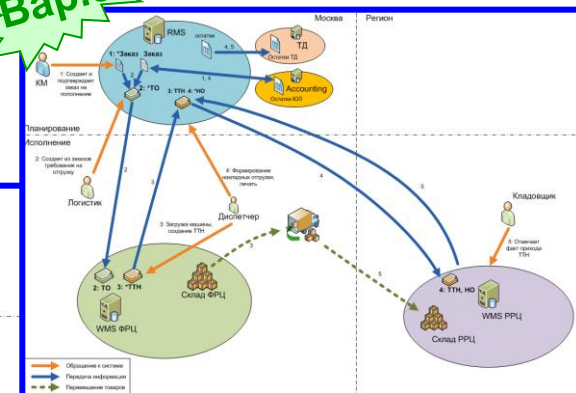
Вар.1



Вар.2



Вар.3



Комплексная схема:

- бизнес-процессы,
- документы,
- информационные системы

Единая витрина для многих взаимодействующих систем

«Лупа» – смотрим на сложную структуру



Что же такое – метафора?

У нас есть (будет) сложная система

И ее архитектурная модель – тоже сложная

Сложную систему понимают через **проекции**

Метафора – это проекция,
наиболее полно представляющая систему

Метафора есть – что дальше?



Делаем стандартные проекции

Для разработки нужны формальные проекции

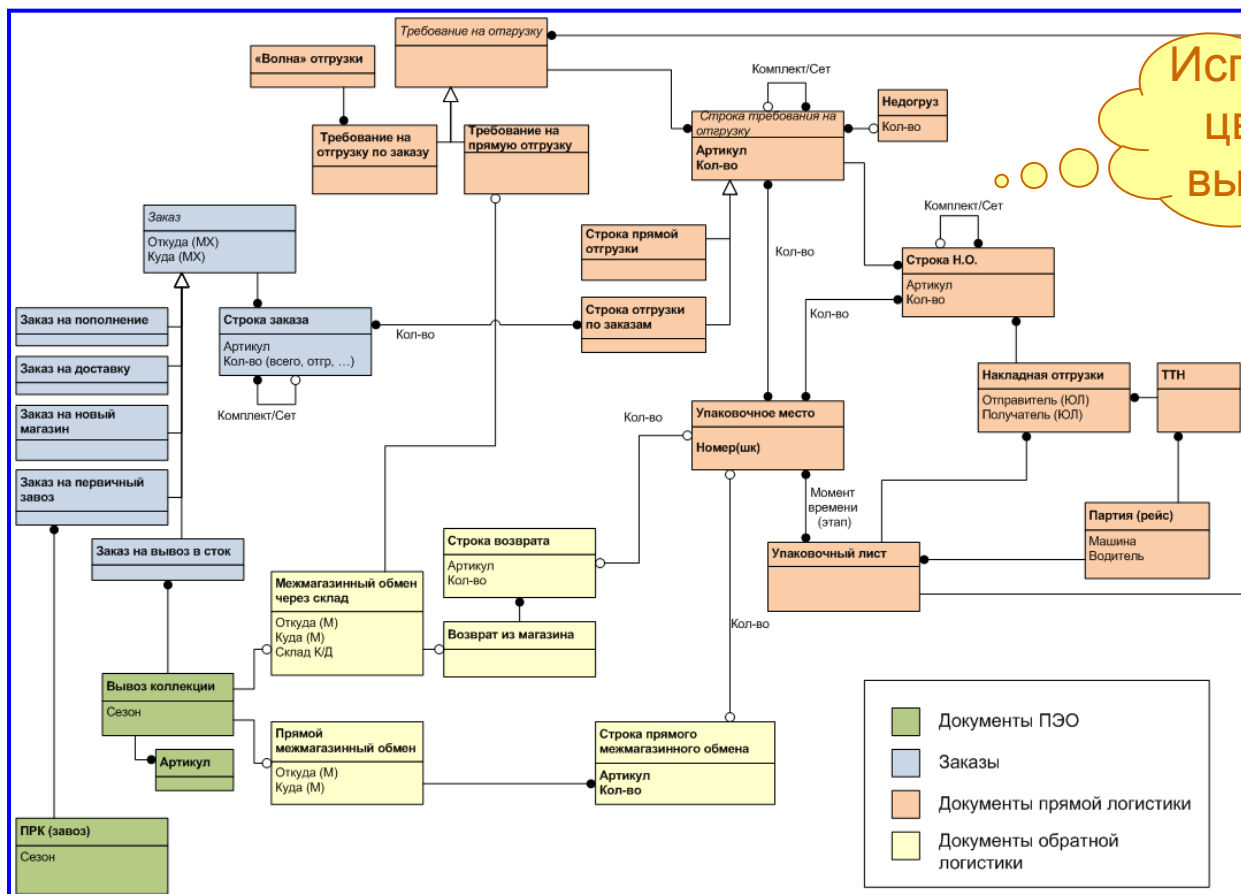
Мы применяем три основных вида:

1. Диаграмму классов
2. Диаграмму учета
3. Диаграмму состояний

Проекция 1 – Объектная модель

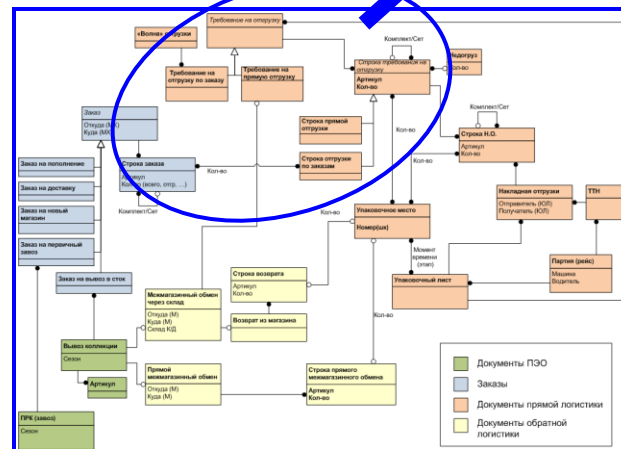
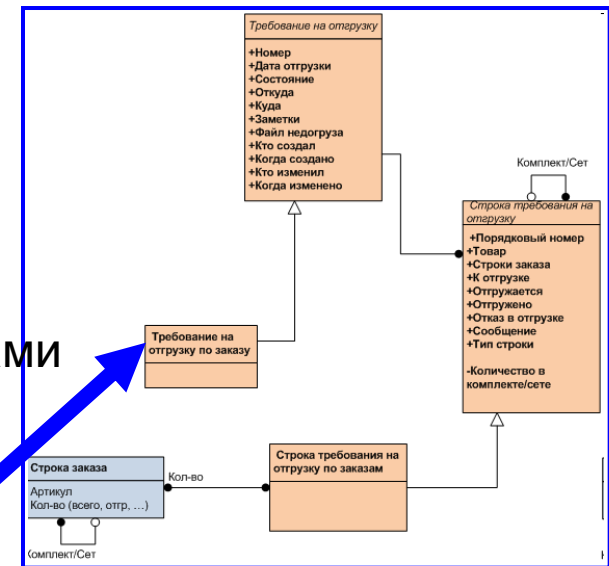
Rich domain model – знакомые заказчику названия

Единый язык



Сначала – **в крупном**, основные классы без деталей

- Уточняем иерархию типов
- Определяем атрибуты и методы
- Проектируем вспомогательные классы
- Уточняем структуру ассоциаций между классами



Рефакторинг объектной модели

Целостность

Бывают ли изменения ранее реализованного? – Да.

- Может меняться набор атрибутов, появляться методы
- Может меняться структура ассоциаций между классами

Постоянный рефакторинг – практика agile

А общая диаграмма классов – снижает его объем

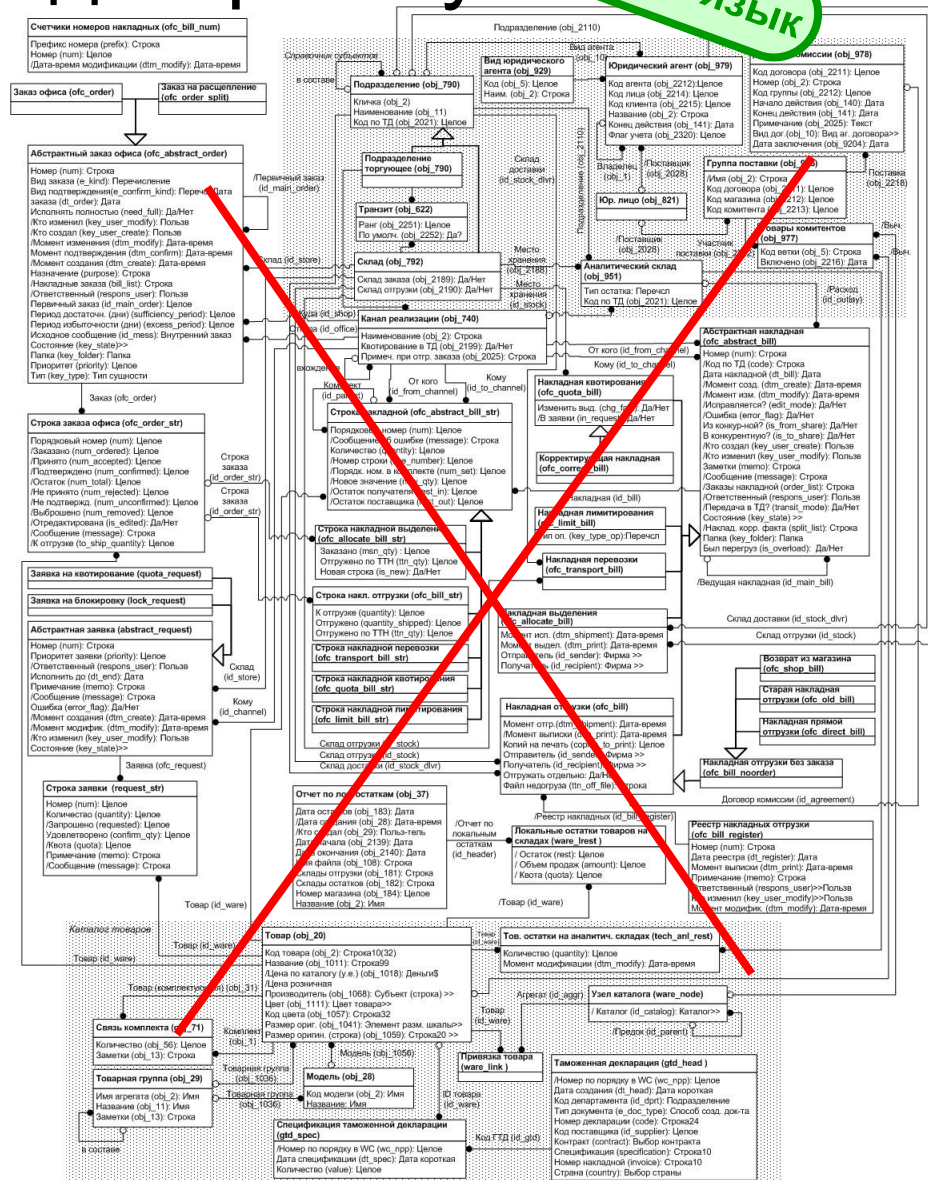
Не нужно усложнять диаграмму

Единый язык

Плохо:

- Стараться изобразить все классы на одной диаграмме
- Отображать вспомогательные атрибуты
- Использовать технические коды
- Использовать сложную нотацию

Диаграммы должны понимать заказчики, а не только разработчики



Проекция 2 – учетная модель

Учет – основное назначение учетной системы

Представление учета – оказалось за рамками UML 😞

И вообще шаблонов разработки...

Есть шаблоны Фаулера, но там – нижний уровень, отражение в объектную модель, а не цельная картина

Мы придумали **Диаграммы учета**

Они дают целостную учетную модель системы

Что такое учет?

Учет – это отражение **реальных** потоков обобщенных ресурсов – товаров, денег, долгов и других

Учет нужен и важен в большинстве управленческих систем, а не только в бухгалтерии

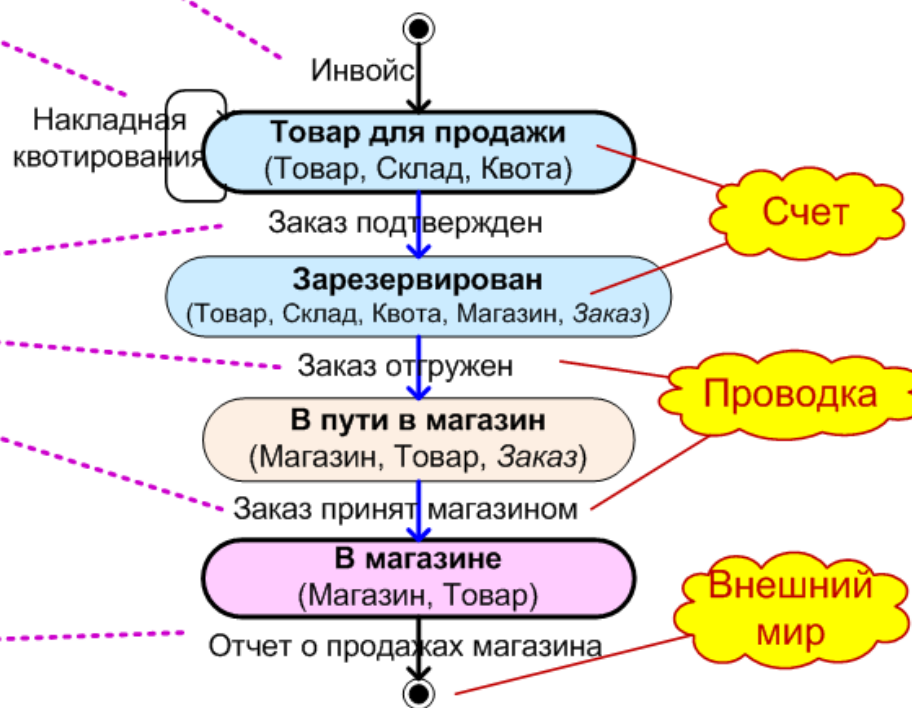
Бухгалтерия же отражает реальные потоки весьма опосредованно



Диаграммы учета



Показывают как отражается движение ресурсов в учете



Подробнее о диаграммах учета

Выступления на конференциях

ЛАФ-2010 – <http://lib.custis.ru/Accounting-diagrams>



Презентация
и видео

Диаграммы планов счетов –
средство моделирования и проектирования учета



Презентация
и статья

SECR-2010 – <http://lib.custis.ru/Simplify-security-accounting>

Учет ценных бумаг – сделать сложное простым

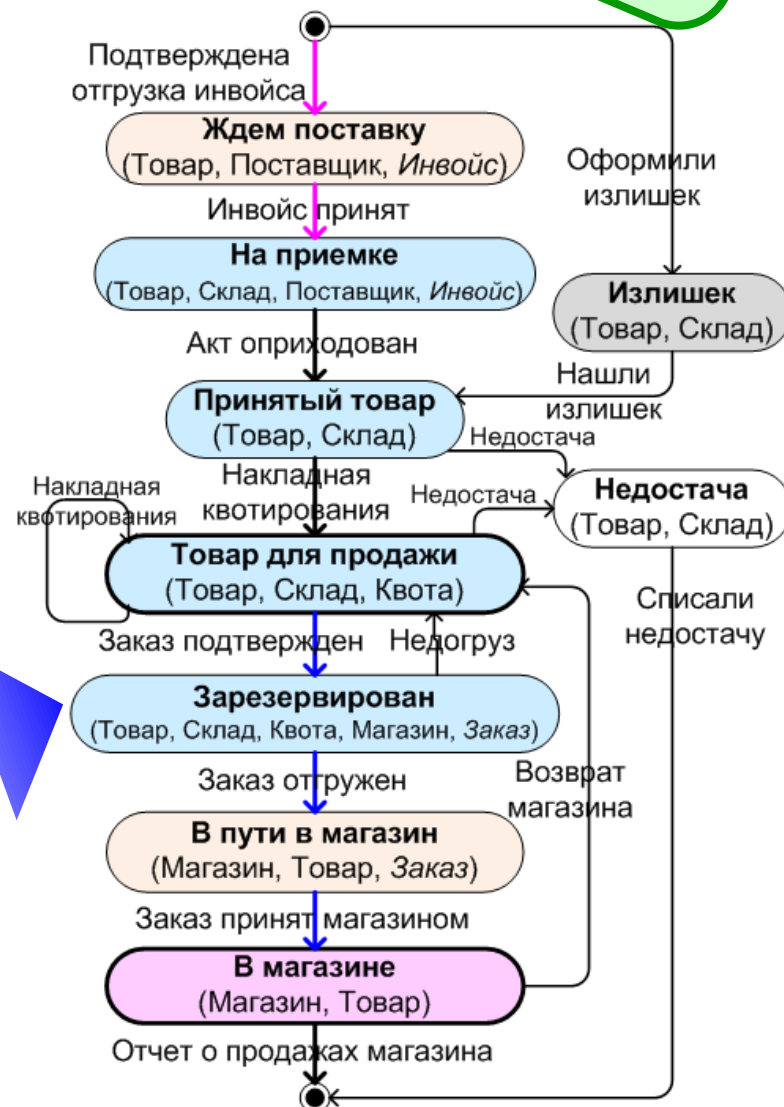
Жизнь учета с развитием проекта

Уточнения

- определяем аналитики
- определяем источники событий учета

Изменения

- добавляем новые участки учета
- добавляем транзитные счета



Что дают диаграммы учета

Представляют учет в виде,
понятном и разработчикам и заказчикам
Позволяют заказчику проверить учетную модель

Единый язык

Дают целостное представление о учете в системе
Позволяют менять доменную модель и
документооборот, контролируя работу учета

Целостность

Проекция 3 – состояния документов

Связывают Учетную и Объектную модели

Учет следует из документов

ВЗГЛЯД БИЗНЕСА

Есть **журнал хозяйственных операций**,
возникающих при обработке и
проведении документов

РЕАЛИЗАЦИЯ

источник учета – **события**
(Event Sourcing Фаулер),
они возникают в методах документов

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

Единый язык

Связь учета и документов – как сделано

Несколько
расширили

Для документов применяем design pattern *State Entity*

- У документов есть атрибут – **состояние**
- Изменение состояние – через вызов метода объекта
- Изменение состояние называется также **переходом** документа
- Учетные события возникают на переходах документов

Состояние документа определяет:

- текущий этап обработки документа
- отражение документа в учете
- ответственность за документ
- права на совершение действий

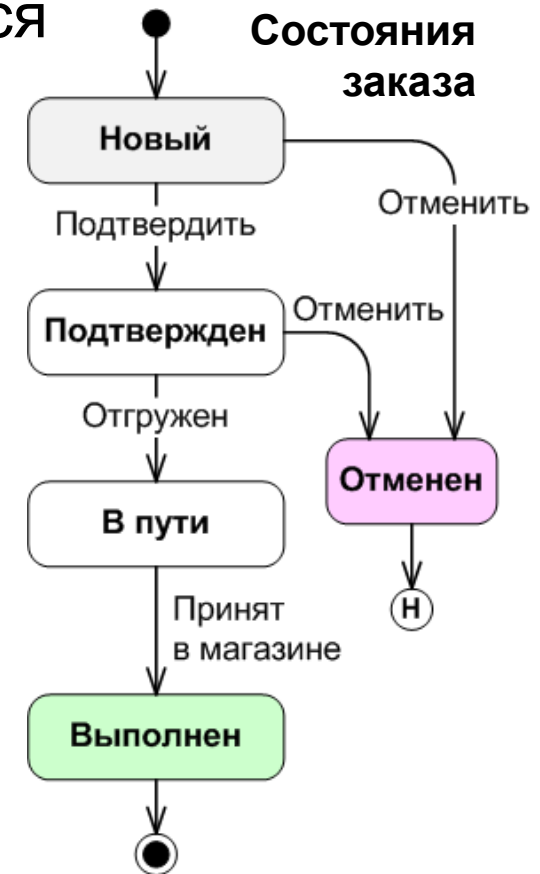
Диаграмма переходов документа

Граф состояний документа изображается на State Machine Diagram (UML)

- узлы – состояния
- ребра – методы-переходы

Состояния и методы
называем в терминах бизнеса

Единый язык



Итак, **единая модель системы**

Модель системы представляется в проекциях

- метафора системы – неформально и в крупном
- диаграмма классов – структура объектов и их методы
- диаграмма учета – учетная модель системы
- диаграмма состояний документов – документооборот

Единый язык

В терминах предметной области

Связь проекций – через методы-переходы, они есть на всех диаграммах

Целостность

Разрабатывается итеративно

- сначала – основные классы и учет в крупном
- затем, **по областям** – детальная проработка классов и учета, проработка состояний документов

Итерации

Пример сложной модели

Документооборот связанных документов



Заключение

(что я хотел сказать)

Да будет итеративная архитектура!

Архитектуру можно создавать итеративно

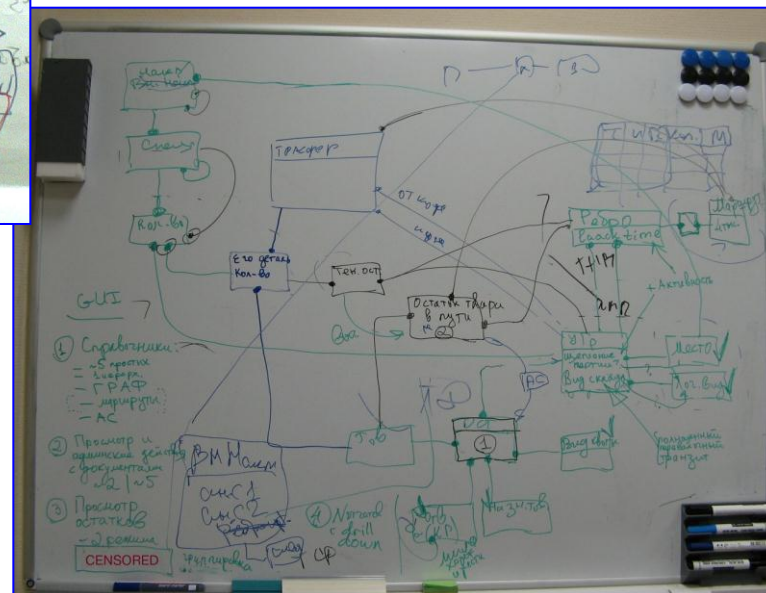
При этом – сохранять её целостность

Это можно делать с помощью модели

Стоит потратить время и силы, чтобы найти образ

Диаграммы – служат основой общения

Все начинается с доски...



Да поймет заказчик разработчика!

Модель системы можно превратить
в мост между разработчиком и заказчиком

Их общение делает разработку эффективной

Спасибо!

Вопросы?

Максим Цепков (M.Tsepkov@custis.ru)