CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

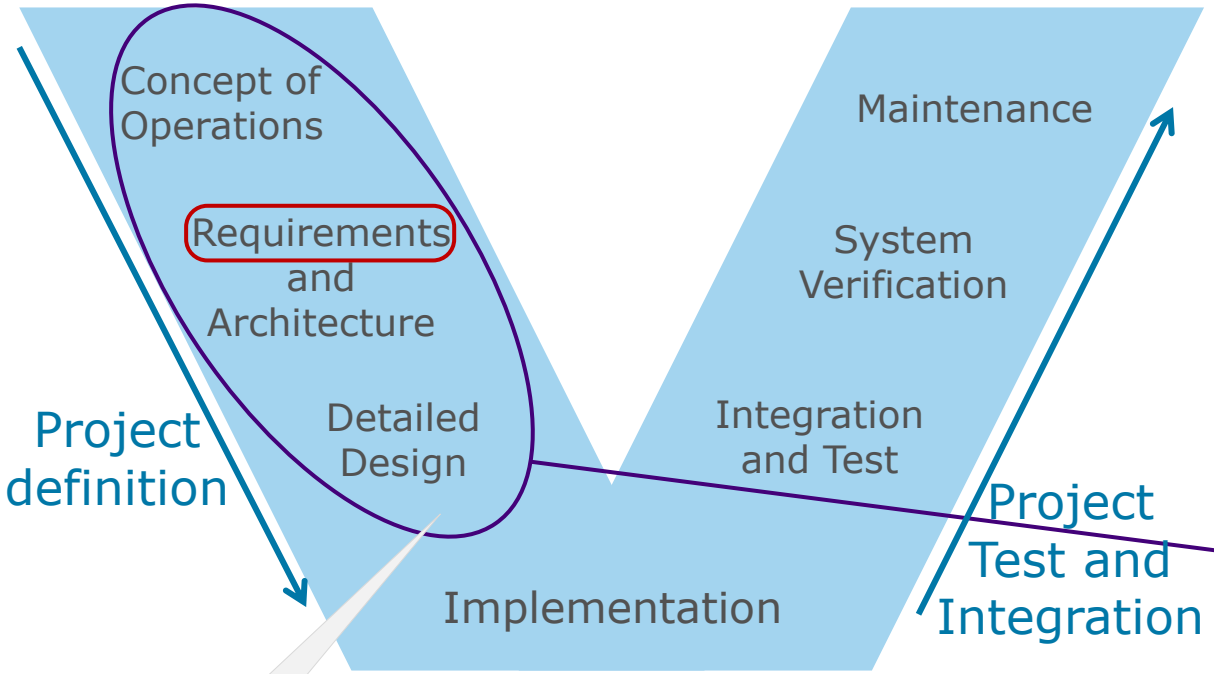
Требования или модели — как писать постановки

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

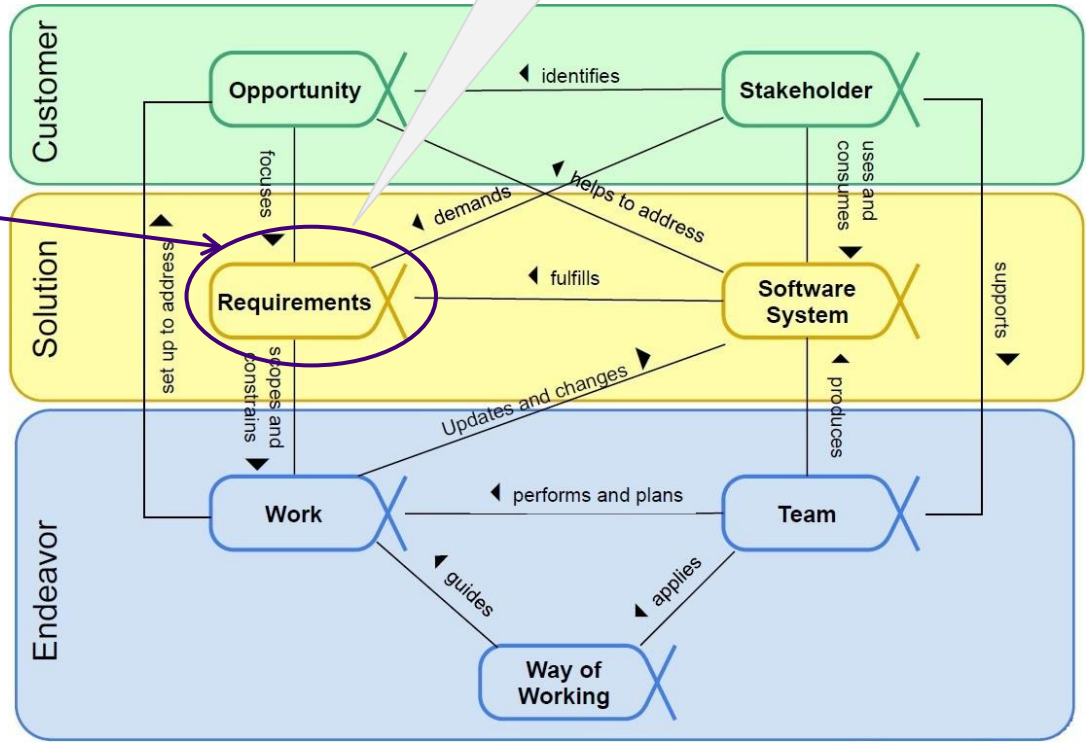
AnalystDays
Москва, 21–22 апреля 2023

ANALYST DAYS #16

Что такое «постановка»



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



Постановка или **проектное решение** — всё, что разрабатывают в ходе проектирования системы

V-Model (Wikipedia)

Краткая история

CUSTIS

- **Проектирование систем** — искусство, как стиль у художника
- Софт можно создавать **правильным методом,** его знают гуру
  - Когда Rational собрал трех методологов, Гради Буча, Ивара Якобсона и Джеймса Рамбо, у этой веры появились основания — родился UML. **Но не случилось**
- Легкие методы для инкрементальной разработки: user story, дизайн на салфетках
- DDD для сложных проектов

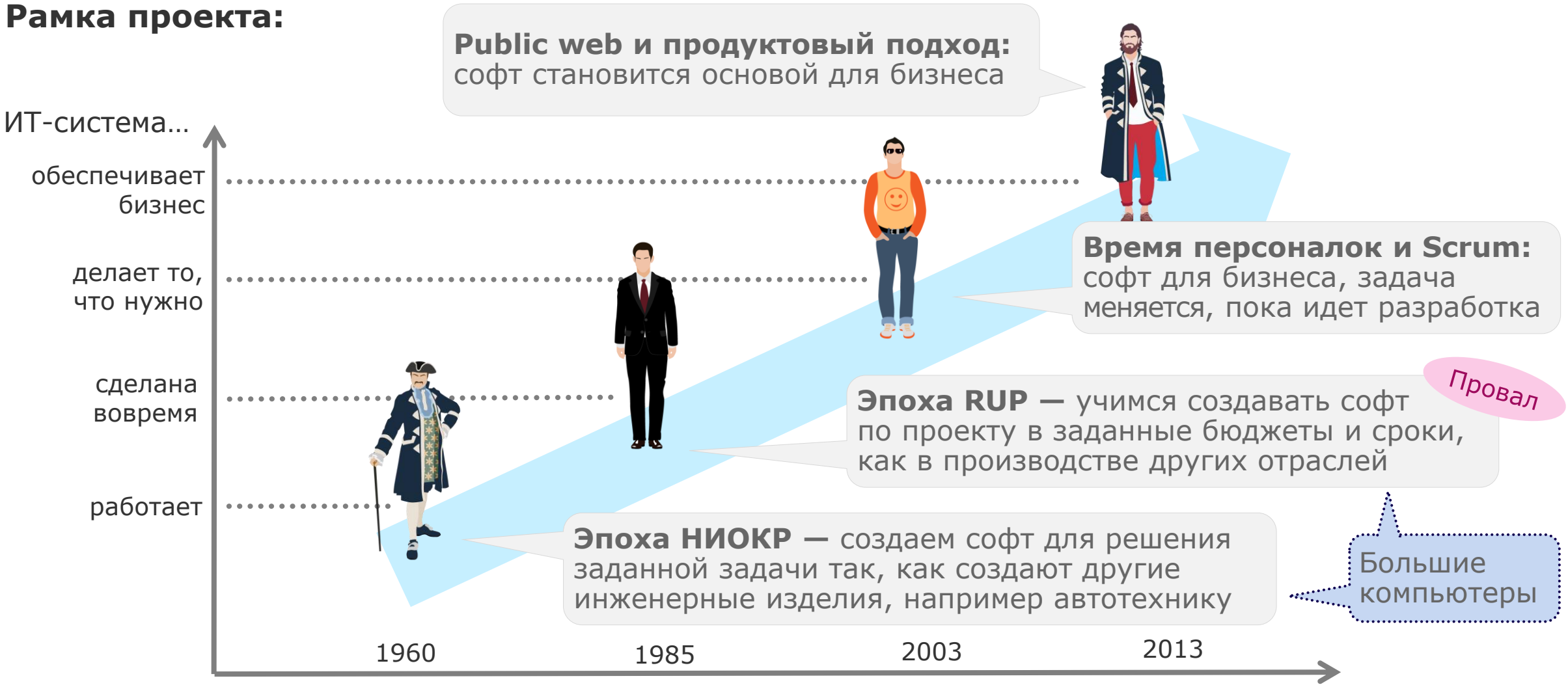
И разные гибридные методы, исходя из специфики проекта и опыта

Требования проекта и опыт команды часто противоречат: выбирая метод, надо представлять спектр возможностей

История культур ИТ-проектов

CUSTIS

Рамка проекта:



Пример: шаблоны документов

CUSTIS

Задача: для эффективной работы операционистов, вводящих документы, сделать механизм шаблонов для ввода типовых документов

Создадим гибкую систему шаблонов, с помощью которой можно настроить шаблон для любого документа

Настройка шаблонов окажется сложным даже для опытных сотрудников. Про удобный поиск и права доступа — забудут

Будет длительная проработка задачи, определение критериев эффективности и описание «идеальной системы»

Шаблоны окажутся жесткими, пригодными только для узкого класса ситуаций, о которых подумали, а расширить будет сложно

Быстрая серия прототипов и решений с последовательным усложнением шаблонов

Первые версии оказаться совсем не адекватными, на демо нужны конечные пользователи и фасилитация. Со сложными и редкими кейсами могут быть проблемы

Узнав о задаче, наметить целевые группы для первых версий и работать именно с их кейсами. Может появиться несколько альтернативных механизмов, в том числе легких и дешевых, например образцы документов вместо шаблонов

Если постановка неверна и нужна другая система?

Жаль, что все это выяснилось так близко к сдаче проекта. Все сделано по требованиям — **вы должны это принять.** А потом мы готовы сделать новый проект за новые деньги

Частые демонстрации **работающего** софта позволяют проверить адекватность его задачам заказчика и **скорректировать движение** проекта

Если при очередной демонстрации выясняется, что софт не позволяет решить задачу для бизнеса, **команда вместе со стейкхолдерами заказчика ищет решение.** Успех проекта — если такое решение реализовано. Деньги и сроки — предмет переговоров

Классические постановки

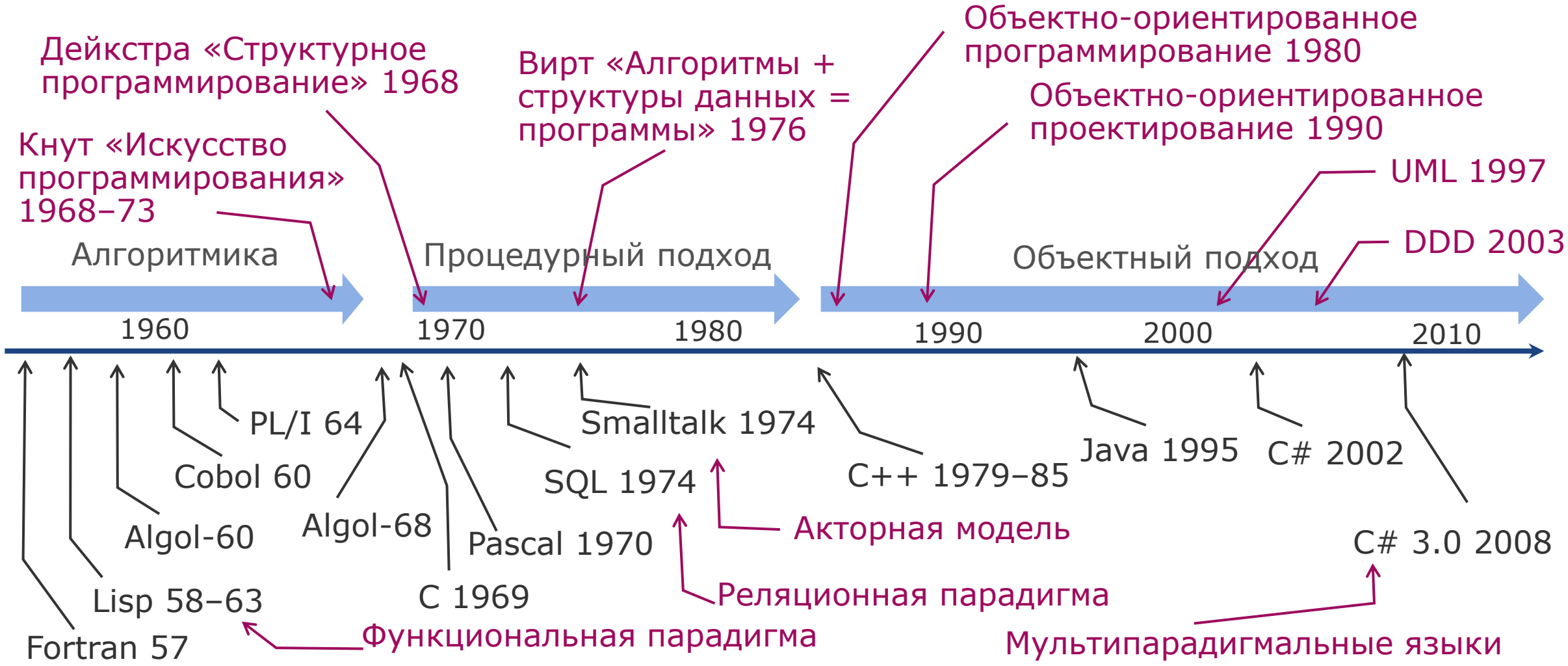
Классические приложения

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.

Развитие программирования и проектирования

CUSTIS



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

Процедурный и объектный подход: пример

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

Если есть доставка самовывозом и курьером, то в процедурном подходе особенности во всех алгоритмах, а в объектном — делаем подтипы

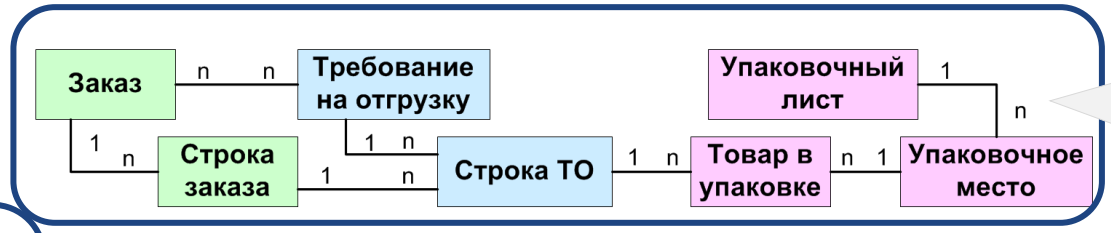
Процедурный и объектный подход

CUSTIS

- Процедурный подход: проектируем структуру БД, интерфейсы и алгоритмы обработки. Иногда — процедуры API backend и API RPC
- Объектный подход: типы и статусы объектов, методы бизнес-логики, интерфейсы в стиле **naked object** — витрины объектов и методы, REST API
- Разница в том, где бизнес-логика — в методах объектов или отдельно
- В реализации может быть анемичная модель транспортных объектов и соответствующие тем же объектам контролеры для бизнес-логики
- **DDD** распространил объектный подход на модель предметной области: вместо словаря мы делаем онтологию понятий и связей, декомпозируя область на фрагменты и применяя методы инкапсуляции, наследования и другие, — концепция **bounded context**

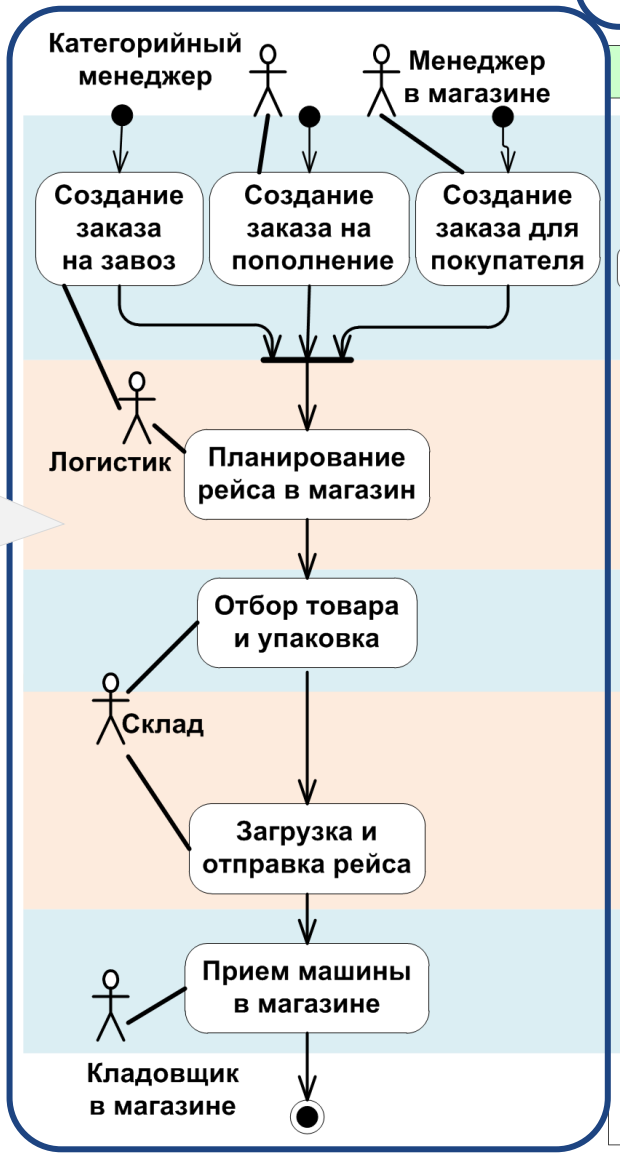
Классическая постановка

Снабжение магазинов



Выделяем объекты-документы

Бизнес-процесс: диаграмма активности



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

CUSTIS

Разделение труда в разработке софта

Понимает ли разработчик смысл приложения?

CUSTIS

Не понимает

Понимает

Создает

Понимать смысл софта — лишнее, аналитики растолкуют на техническом языке, что сделать

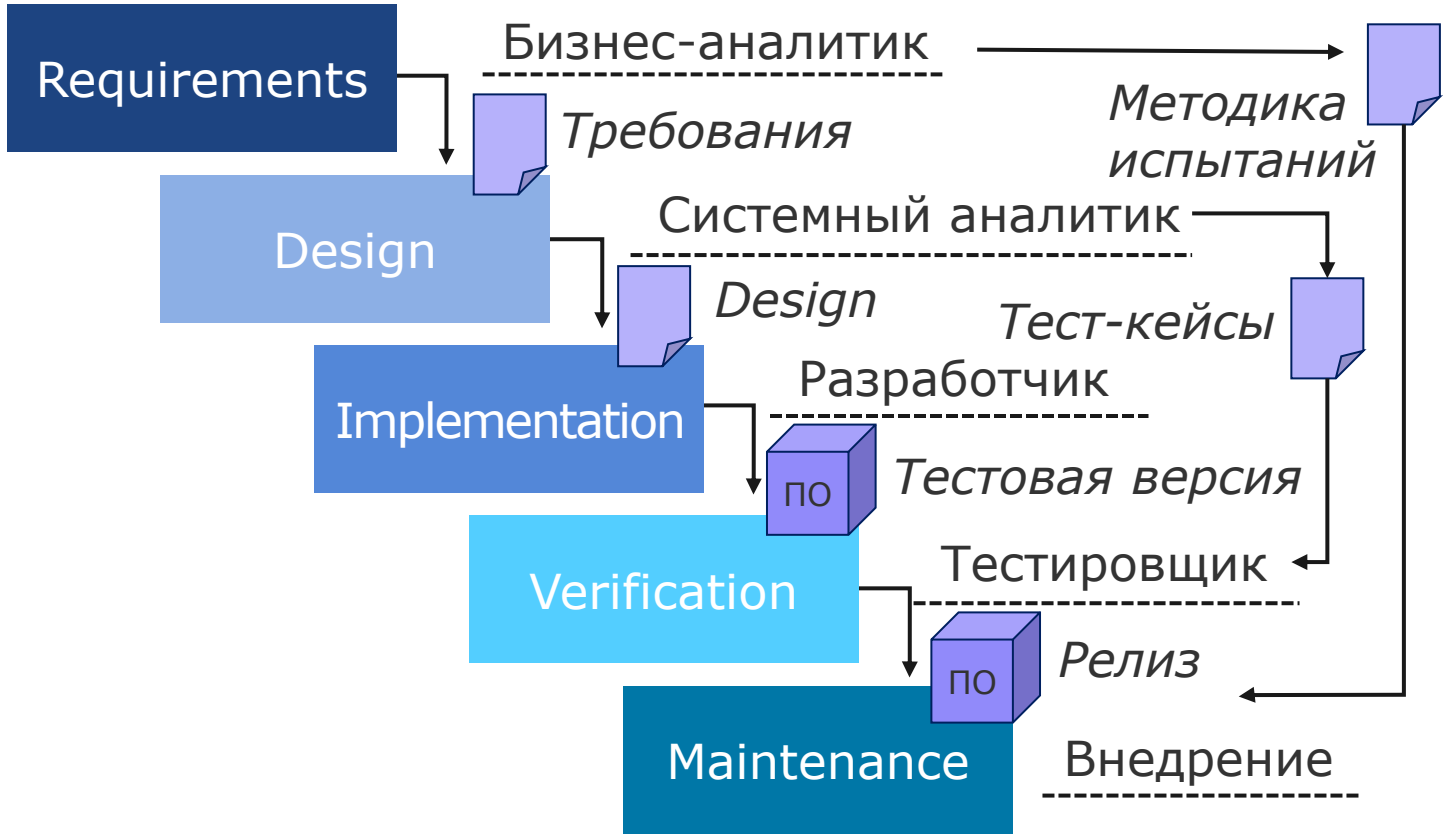
Разработчик понимает цели софта и предлагает технические решения и идеи

Разработчик должен взаимодействовать с заказчиком и понимать его язык, цели и использование софта

Разделение труда выделило позиции аналитиков, работающих с постановками

Разработчик в команде проекта активно участвует в формировании целей софта

Водопад — специализация по фазам проекта



Каждой фазе — своя роль

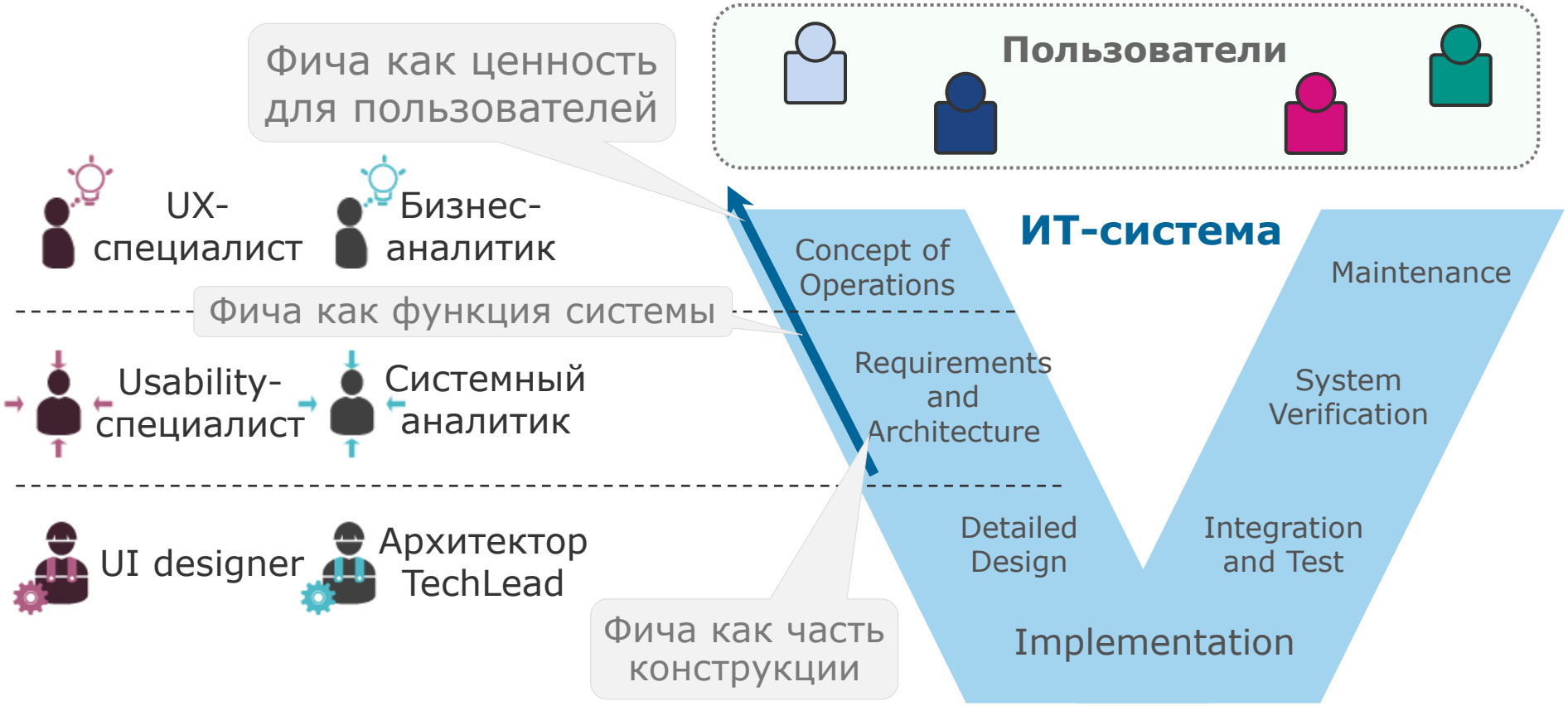
Роли выполняются разными людьми или командами

Передача работы формализована **через артефакты**

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

Уровни требований на V-модели

CUSTIS

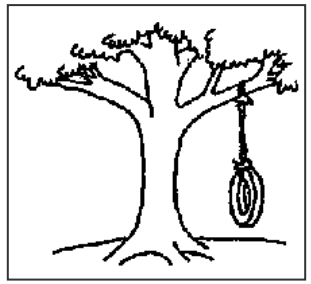


В реальных проектах такого детального разделения труда не бывает, есть аналитик или UX и разработчики, иногда — архитектор

Каждая передача искажает смысл

CUSTIS

Что хотел заказчик



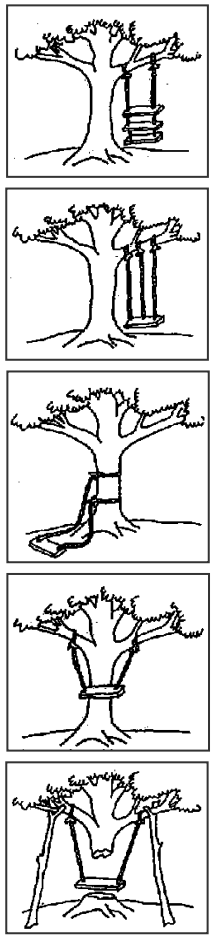
Requirements → Бизнес-аналитик

Design → Системный аналитик

Implementation → Разработчик

Verification → Тестировщик

Maintenance → Внедрение



Старый известный образ

Предсказуемость результата — Cynefin framework

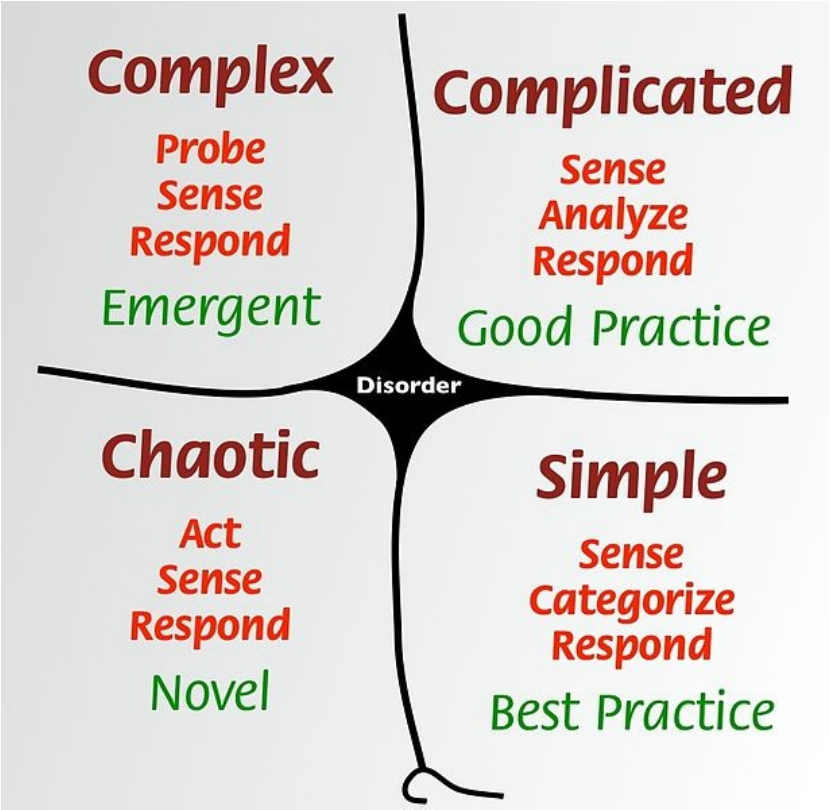
Классические постановки – для **сложной (complicated)** области

Запутанная

Объектов и связей «причина — следствие» слишком много, **модель построить нельзя,** можно экспериментально подбирать действенные методы работы, но **без гарантий** результата

Хаотичная

Реакция на действия непредсказуема, можно **экспериментом** нащупывать путь



Сложная

Связи «причина — следствие» могут быть выявлены **исследованием,** и в исследованной части действовать уверенно

Простая

Известные связи «причина — следствие», есть регламенты и правила, по которым достижение цели гарантировано

Деление областей **зависит от квалификации:** простое для одного — сложно для другого, а **сложное** для одного может другому казаться **запутанным** или хаотичным

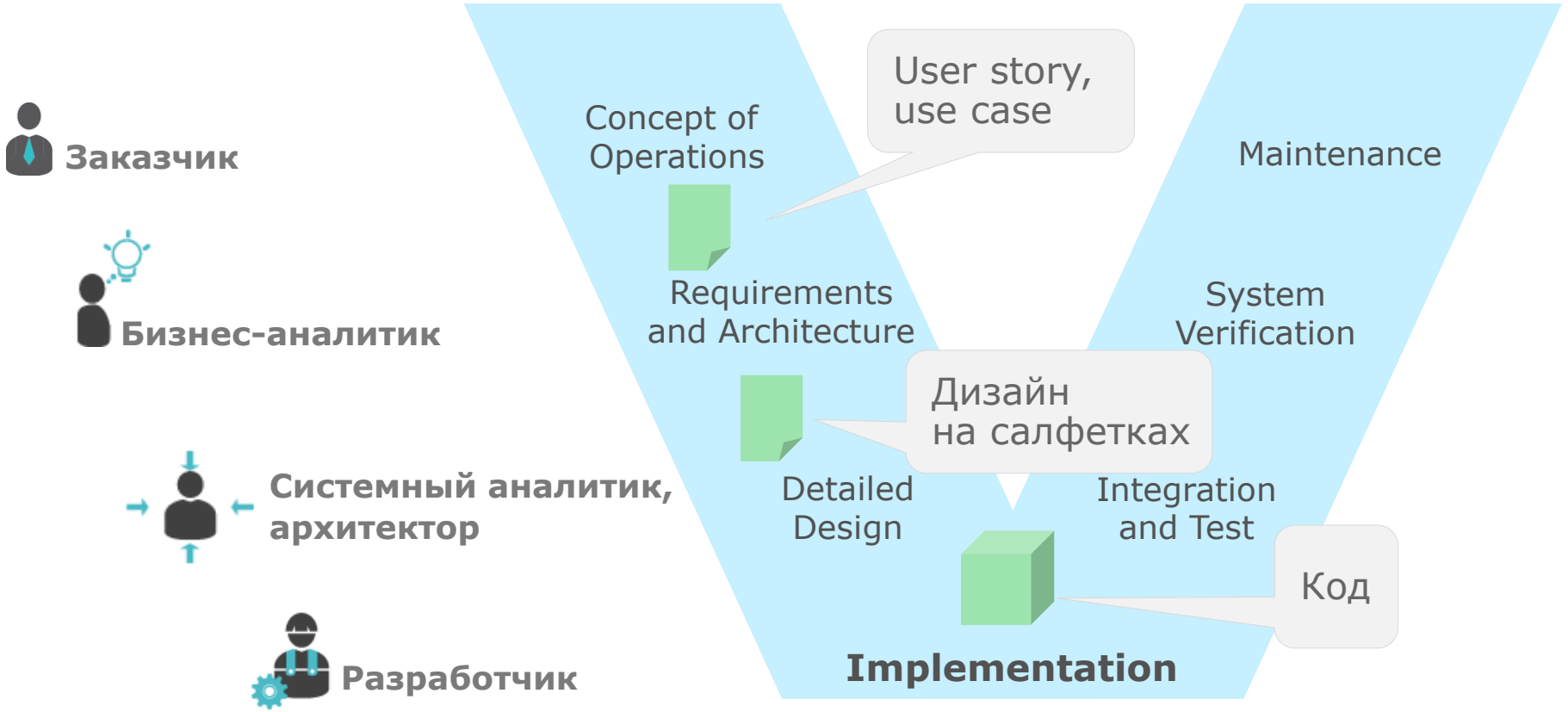
Cynefin framework (Wikipedia)

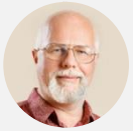
CUSTIS

Agile: light-версия требований и дизайна

Light-версия требований и дизайна

CUSTIS





Хорошо подходит для небольших систем или сервисов

Короткие задачи для итераций

CUSTIS

User story — небольшие истории, каждая имеет ценность

Формат: Как <роль> я хочу сделать <действия> для того, чтобы <цель>

Часть <цель> объясняет смысл действий пользователя, позволяя выбирать из альтернативных решений

Метод use case был доработан для инкрементальной разработки Иваром Якобсоном, Use Case 2.0: большой use case делим на slice

Истории компонуют в ценный функционал, приоритизируют и выбирают в спринты

Фокус на взаимодействии: **story mapping, customer journey map, event storming** — форматы встреч, а не артефакты

Сценирование большого проекта

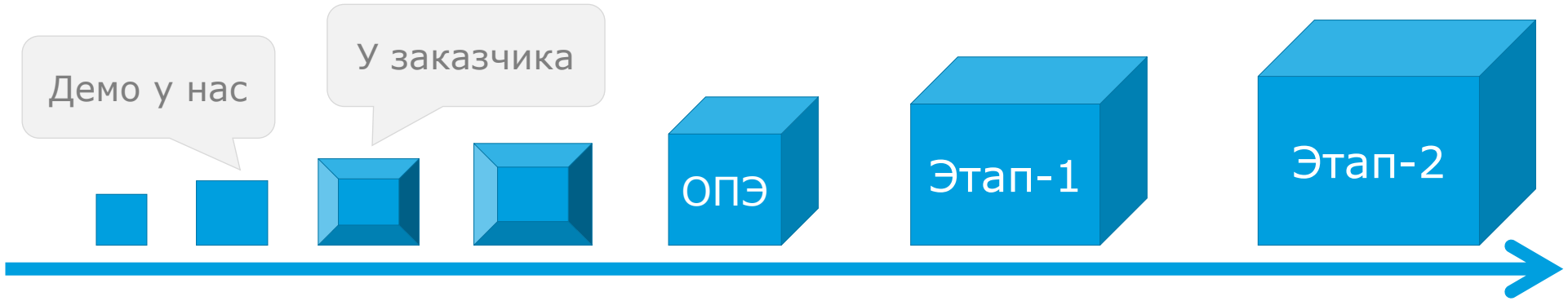
CUSTIS

Для замены существующей системы часто нужен большой функционал, однако часто можно пойти на поэтапное внедрение, выделить MVP

Если разработка для запуска в эксплуатацию 4–6 месяцев, разбиваем **на 2–4 демо**

На демо представляем интересный бизнес-заказчику **целостный** функционал

Первые демо — на своей территории — так **заказчик знакомится с командой**

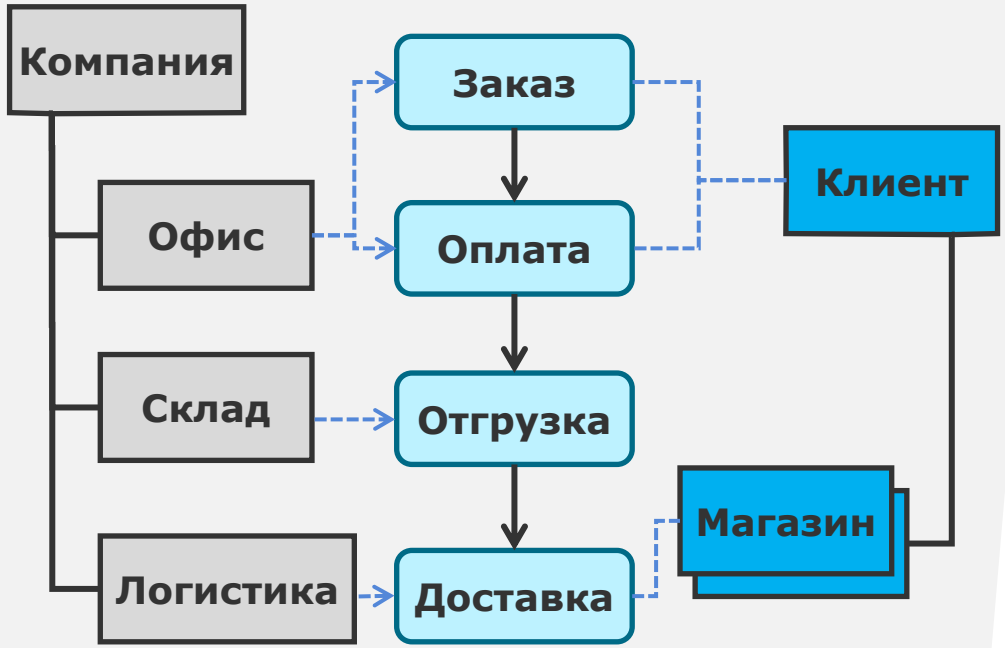


Модели вместо требований

Два представления о реальности

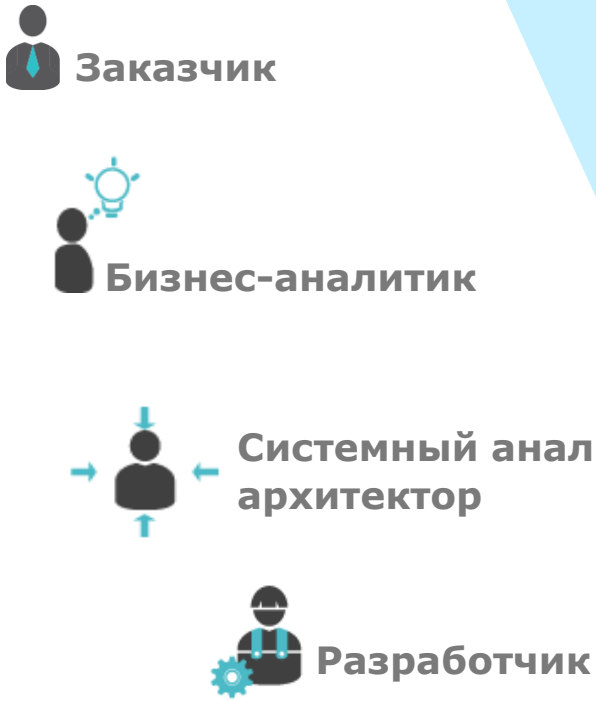
CUSTIS

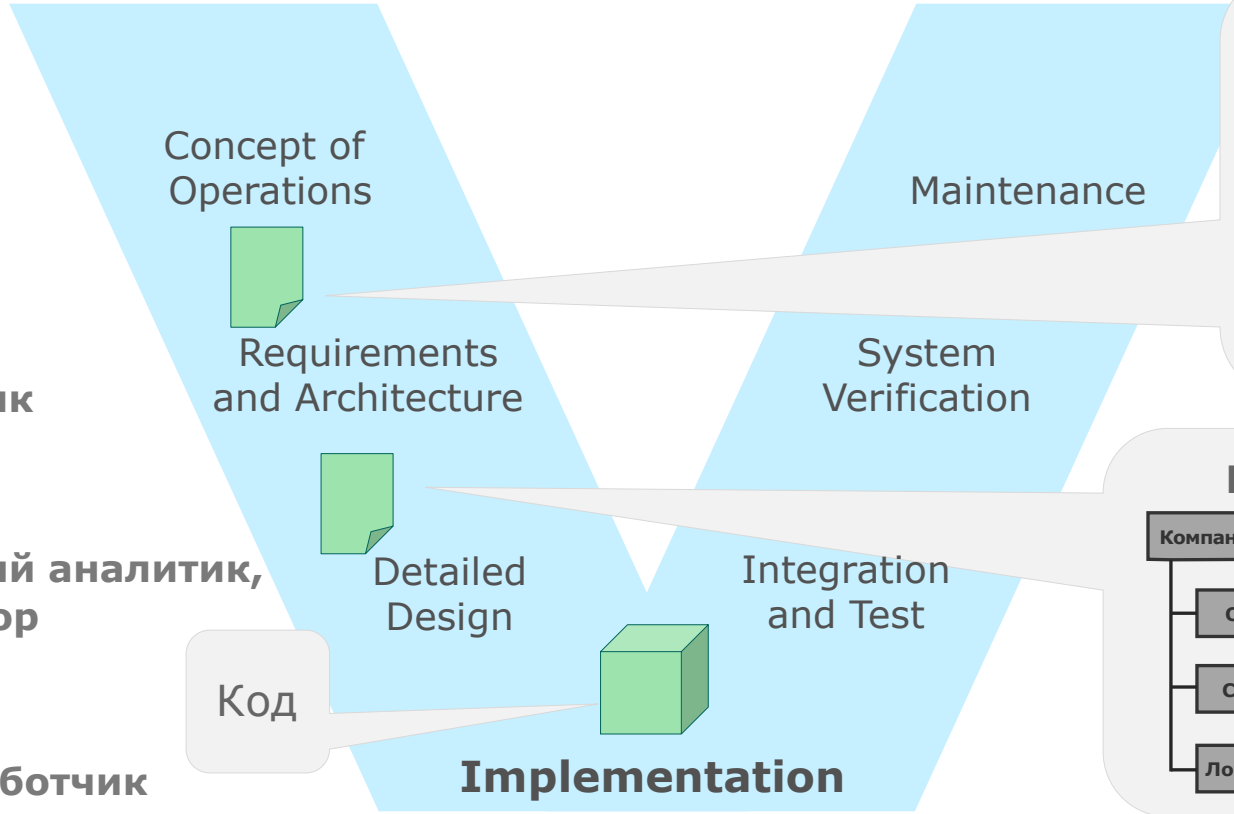
Оптовые продажи магазинам и торговым сетям

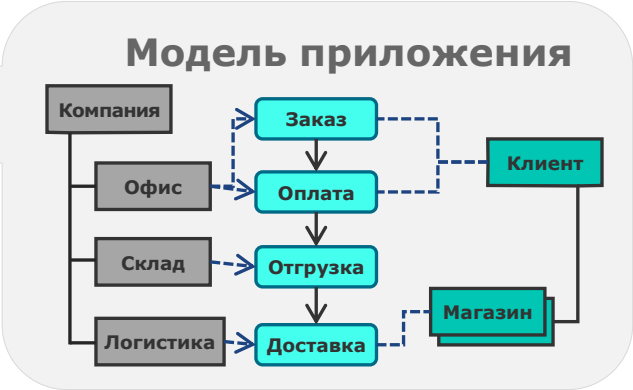


Модели в классических постановках

CUSTIS



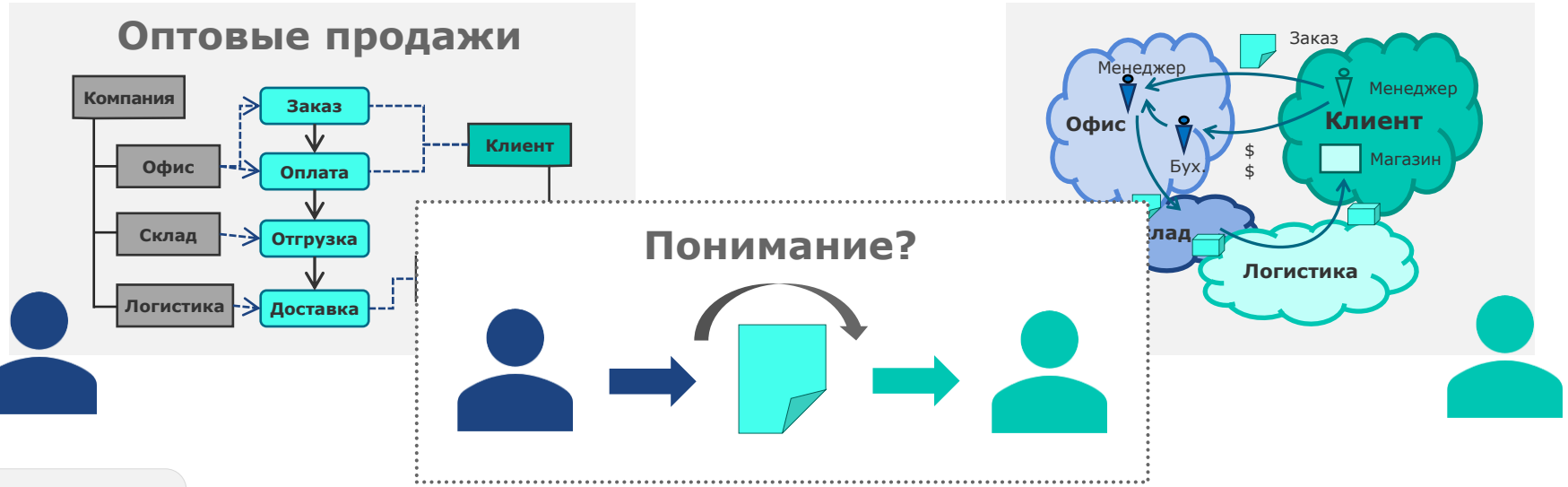


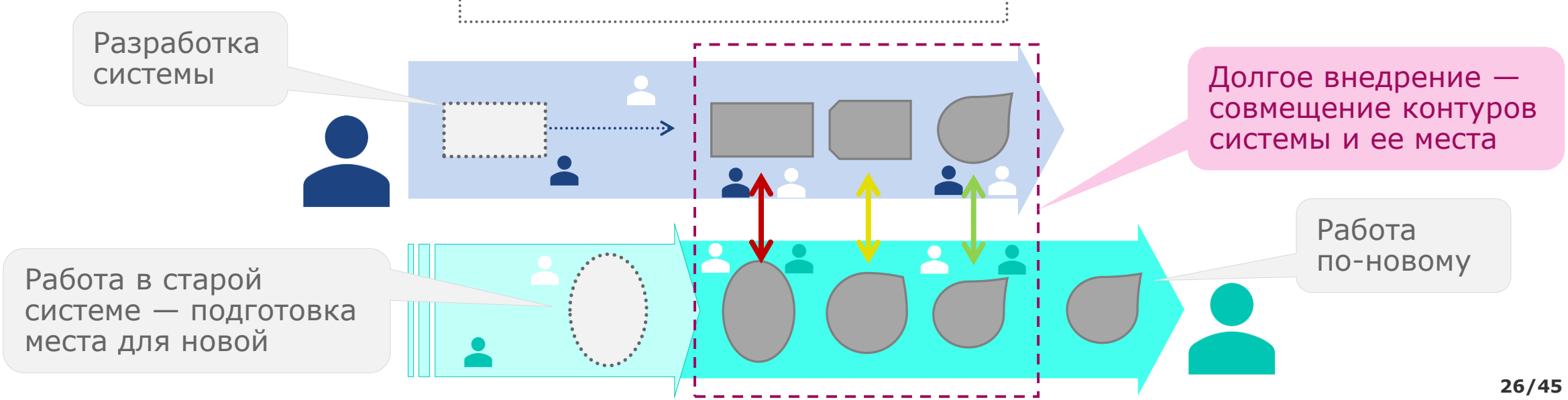


Проблема: каждый уровень отдельно — сложно изменять и поддерживать соответствие

Проблема двух моделей

CUSTIS

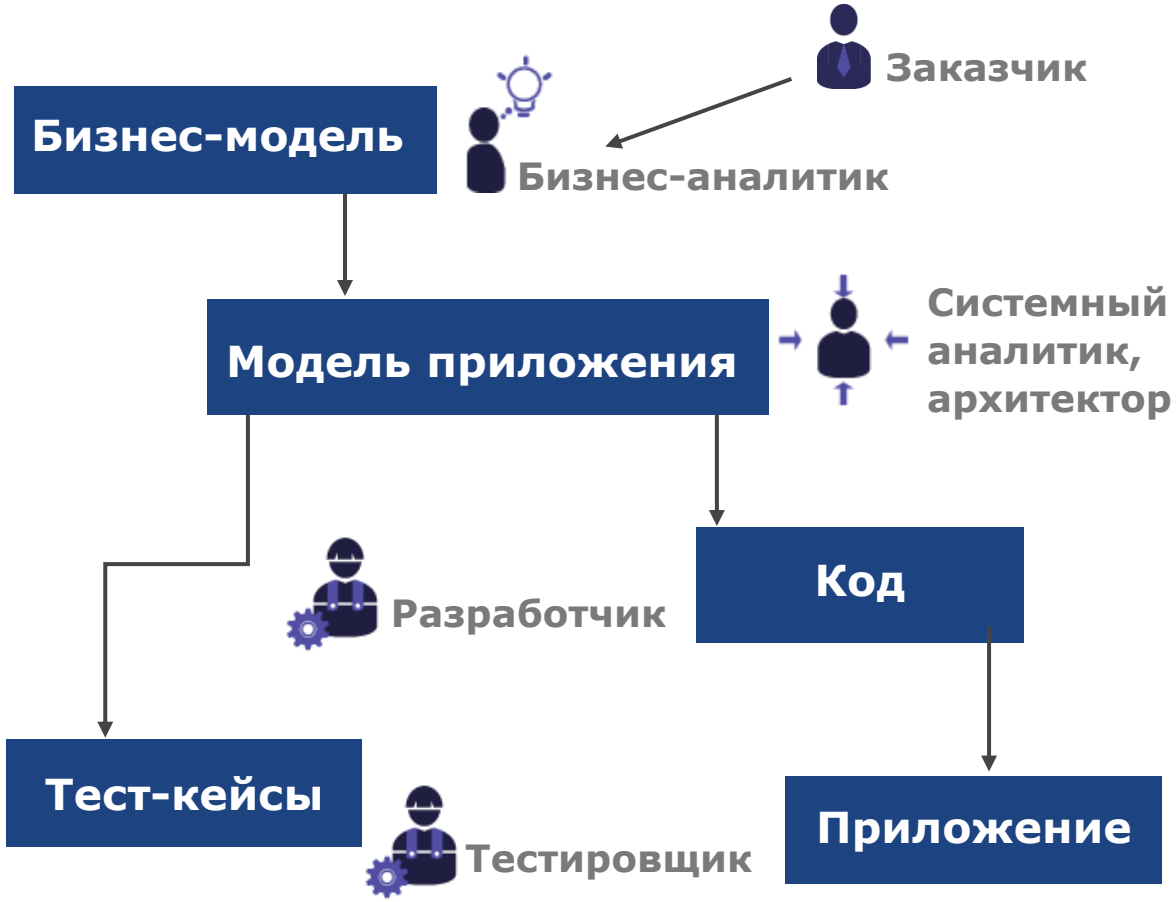




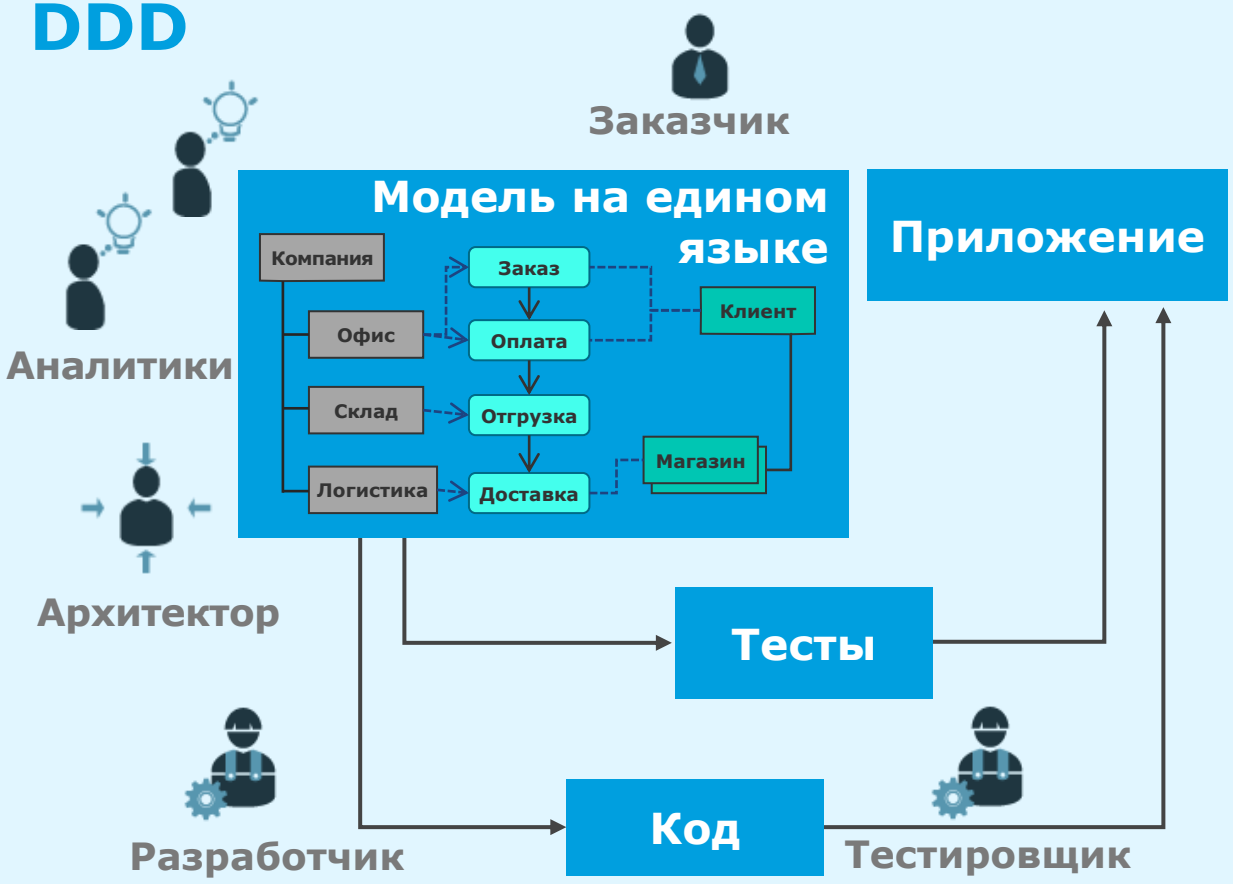
DDD: единый язык и единая модель приложения

CUSTIS

Классика



DDD



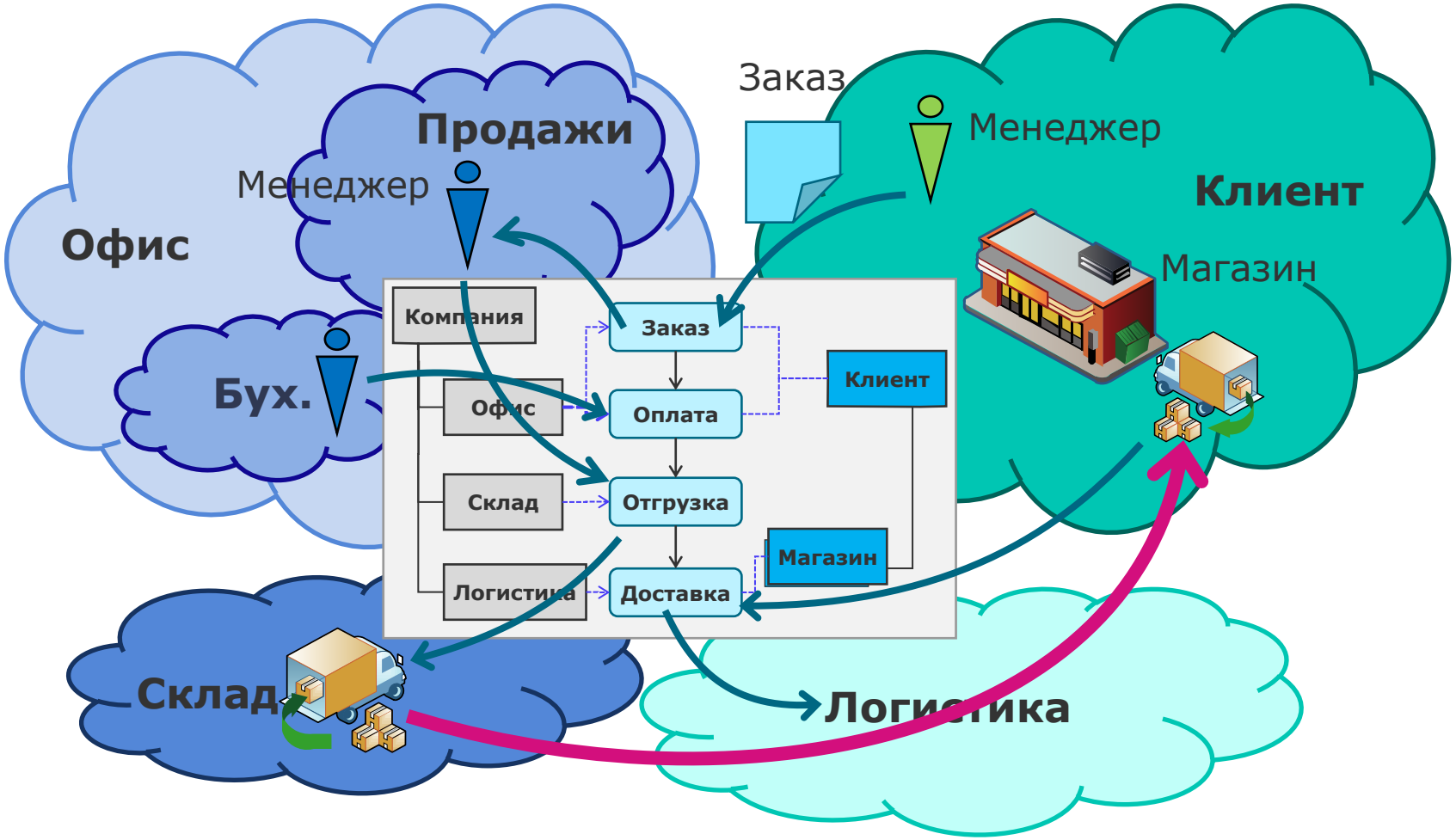
Единый язык создает общее пространство общения

Система дополняет ИТ-ландшафт

CUSTIS

Система становится основой

CUSTIS



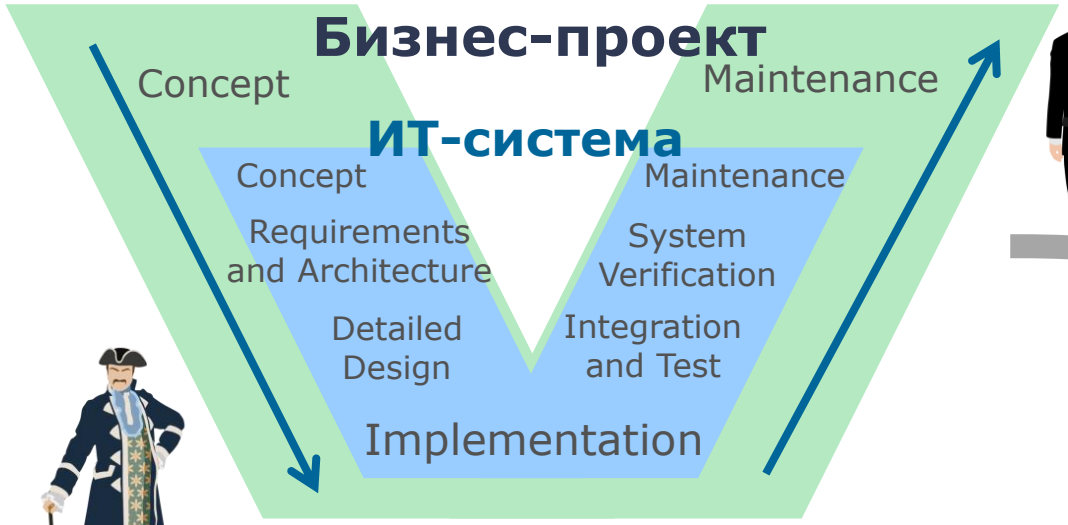
Три категории постановок

CUSTIS

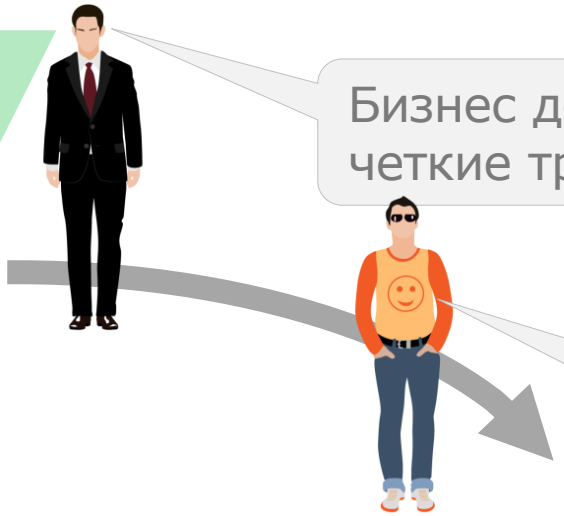
1. Разработка новой системы поверх фрагментарной и малой автоматизации:
   - выясняем модель бизнес-процессов
   - создаем модель системы
   - работаем над ее встройкой в процессы, изменяя их
2. Доработка существующей системы: проектируем изменения модели существующей системы и ее встройки в процессы
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.**

ИТ- и бизнес-проект объединяются



Есть **только** ИТ-проект

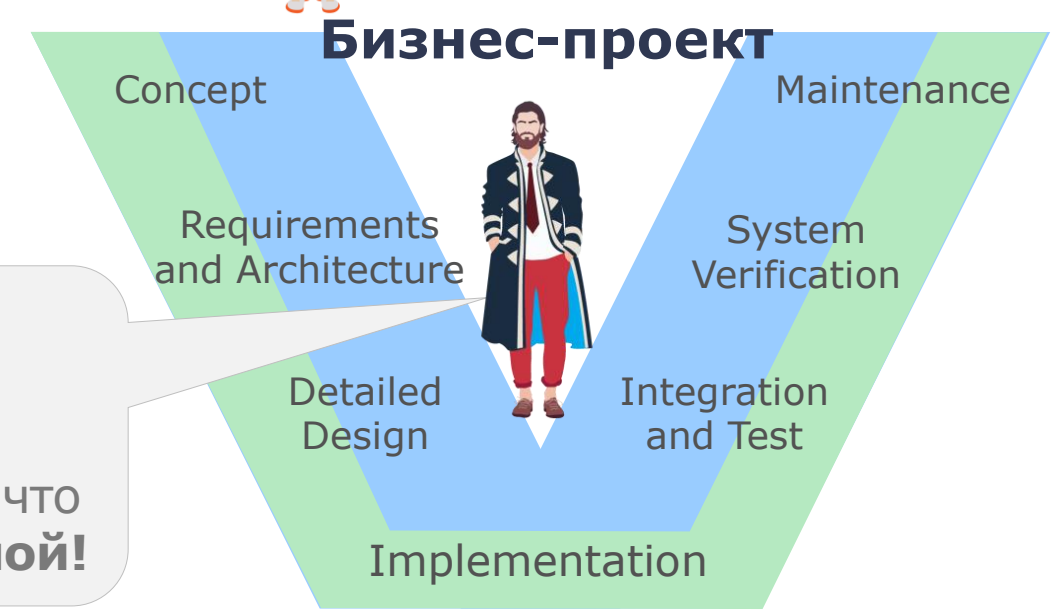


Работа по заказу

Бизнес должен дать четкие требования к ИТ

Сервис

Меняем ИТ в темпе изменений бизнеса



Партнерство

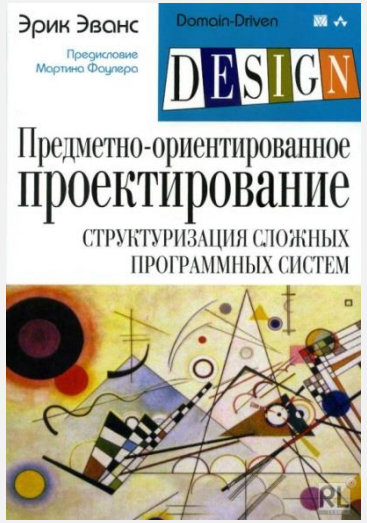
В современных проектах **граница** между ИТ-частью проекта и изменением бизнеса **подвижна,** и о ней надо принимать решения в ходе проекта. **Диджитализация** означает, что **проект будет единым, а ИТ-часть — главной!**

CUSTIS

Domain Driven Design

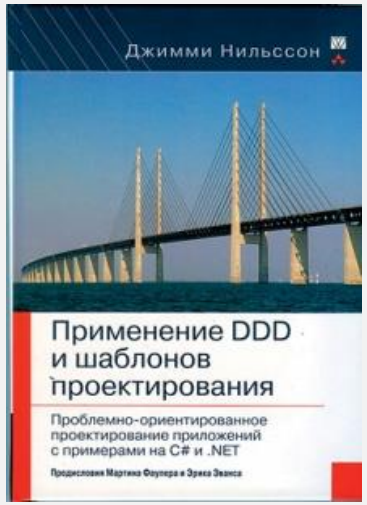
DDD: источники

CUSTIS



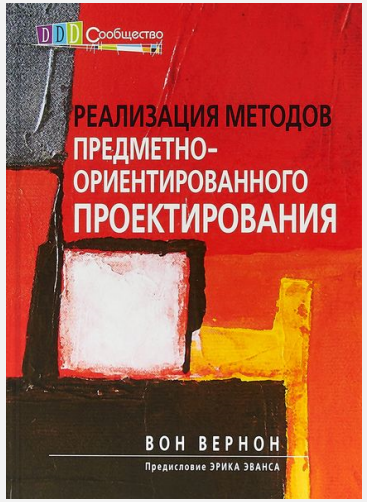
Концептуальная книга Эрика Эванса

- 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

Модель предметной области — модульная

CUSTIS

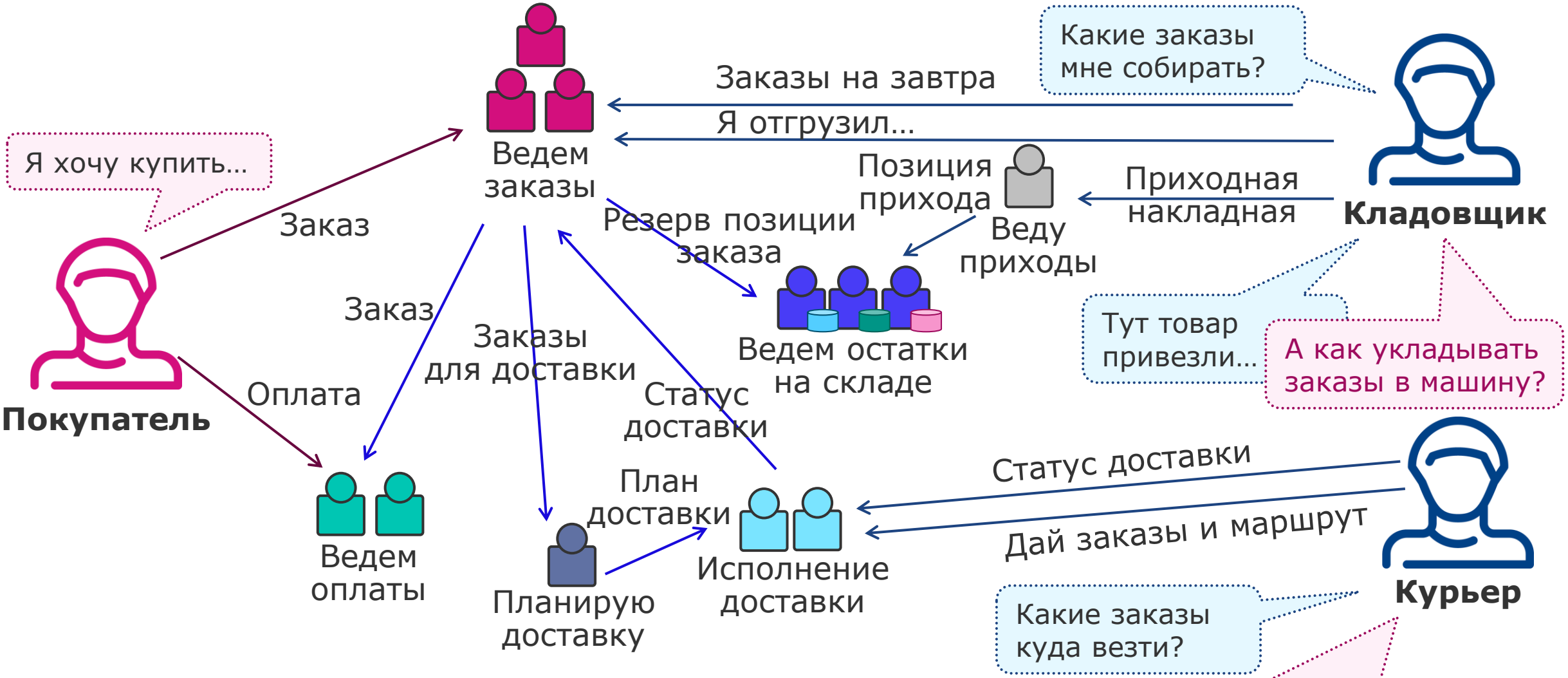
- Предметную область делим на ограниченные контексты
- Для каждого из них — свой единый язык и своя модель
- Целостность удерживает карта контекстов
- Для изоляции и выделения общего в моделях работают методы ООП
- Модели учитывают позицию фирмы: поставки и продажи — разные контексты и модели, хотя сделка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

Модель для сервисной архитектуры



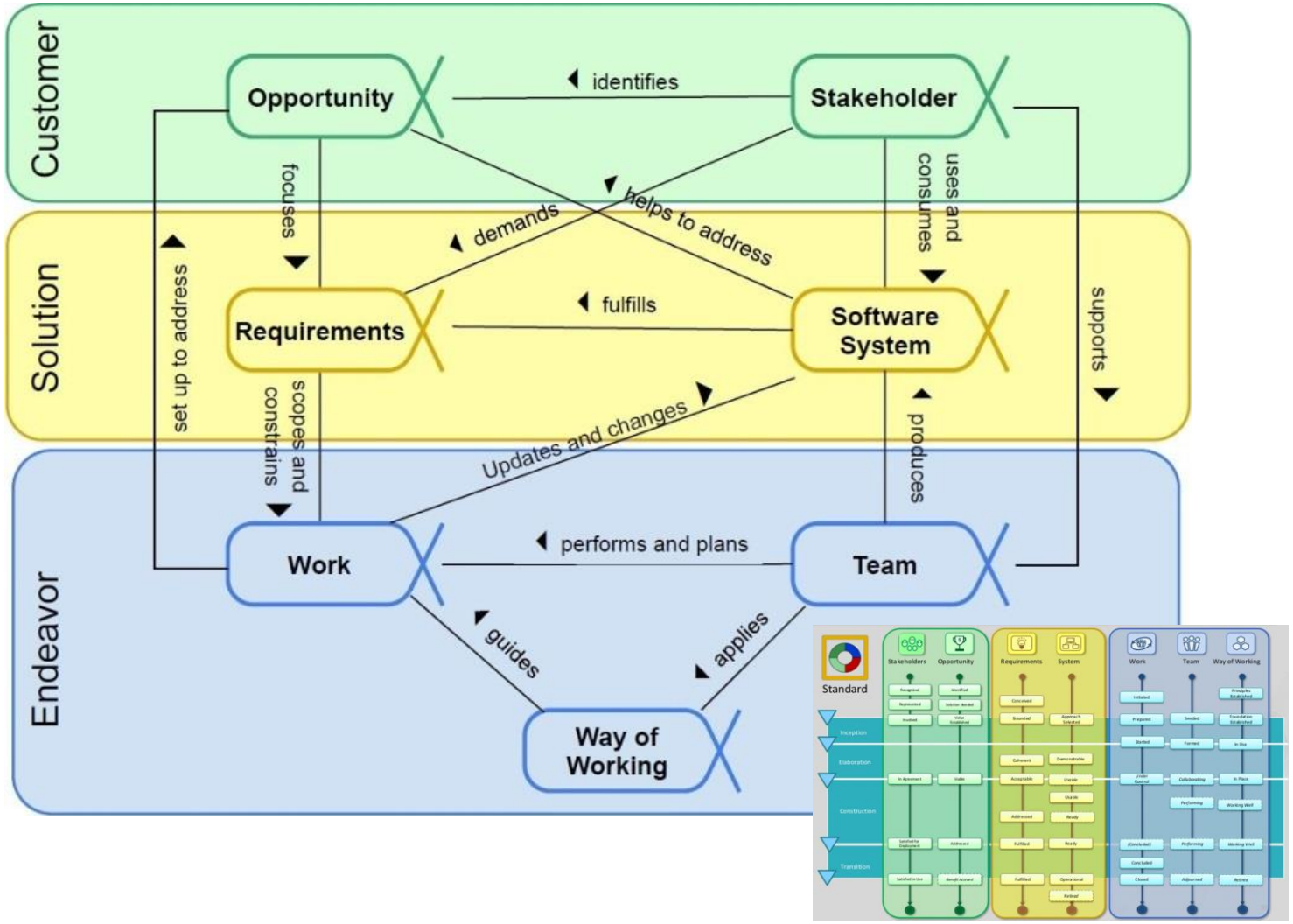
Из доклада «Визуальное проектирование масштабируемых приложений»

CUSTIS

OMG Essence — способ описать метод

Источники: Спецификация на сайте OMG и библиотека практик на сайте Ивара Якобсона (требуется регистрация)

Фокусы внимания вместо линейной схемы проекта



На одной схеме **разработка**, **эксплуатация** и **развитие**, различие — в масштабе и lifecycle объектов

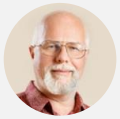
В область управления включены заказчики и бизнес

Есть описания большинства методов — в стандарте и practice library у Ивара Якобсона

Анатолий Левенчук перенес в Системную инженерию

Требования и возможности в Essence

CUSTIS



Вольный перевод

Opportunity	Req
Identified	
Solution Needed	Conceived
Value Established	Bounded
Viable	Coherent
	Acceptable
	Addressed
Addressed	Fulfilled
Benefit Accrued	

Обнаружены коммерческие или социальные возможности

Необходимо ИТ-решение

Определена ценность решения

Сроки и стоимость решения приемлемы

Демонстрируется, что решение доставляет ценность

Ожидаемый эффект достигается

Согласована потребность в ИТ-решении

Ясно назначение и область системы

Описаны основные характеристики

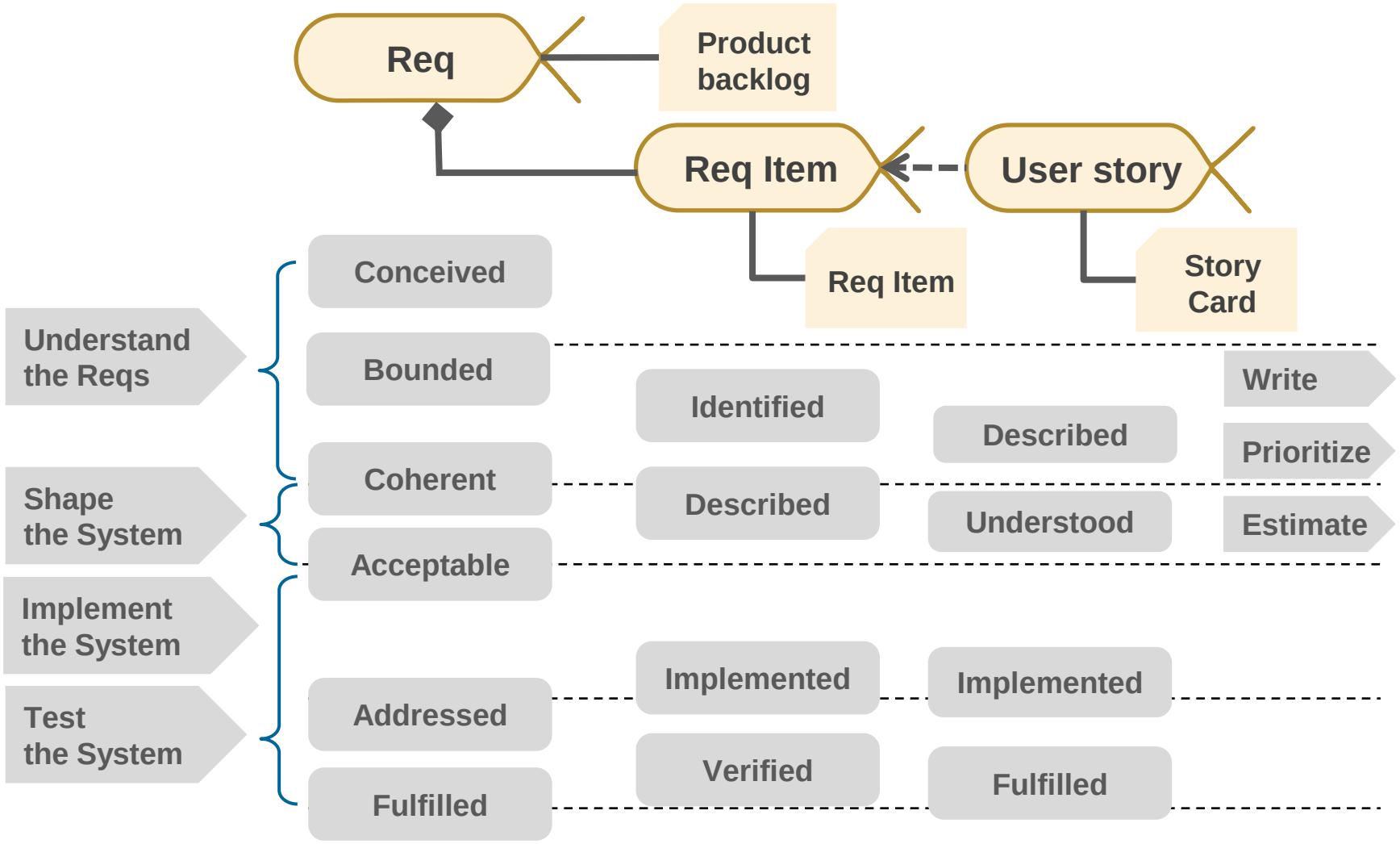
Описание принято стейкхолдерами

Значительная часть требований удовлетворена

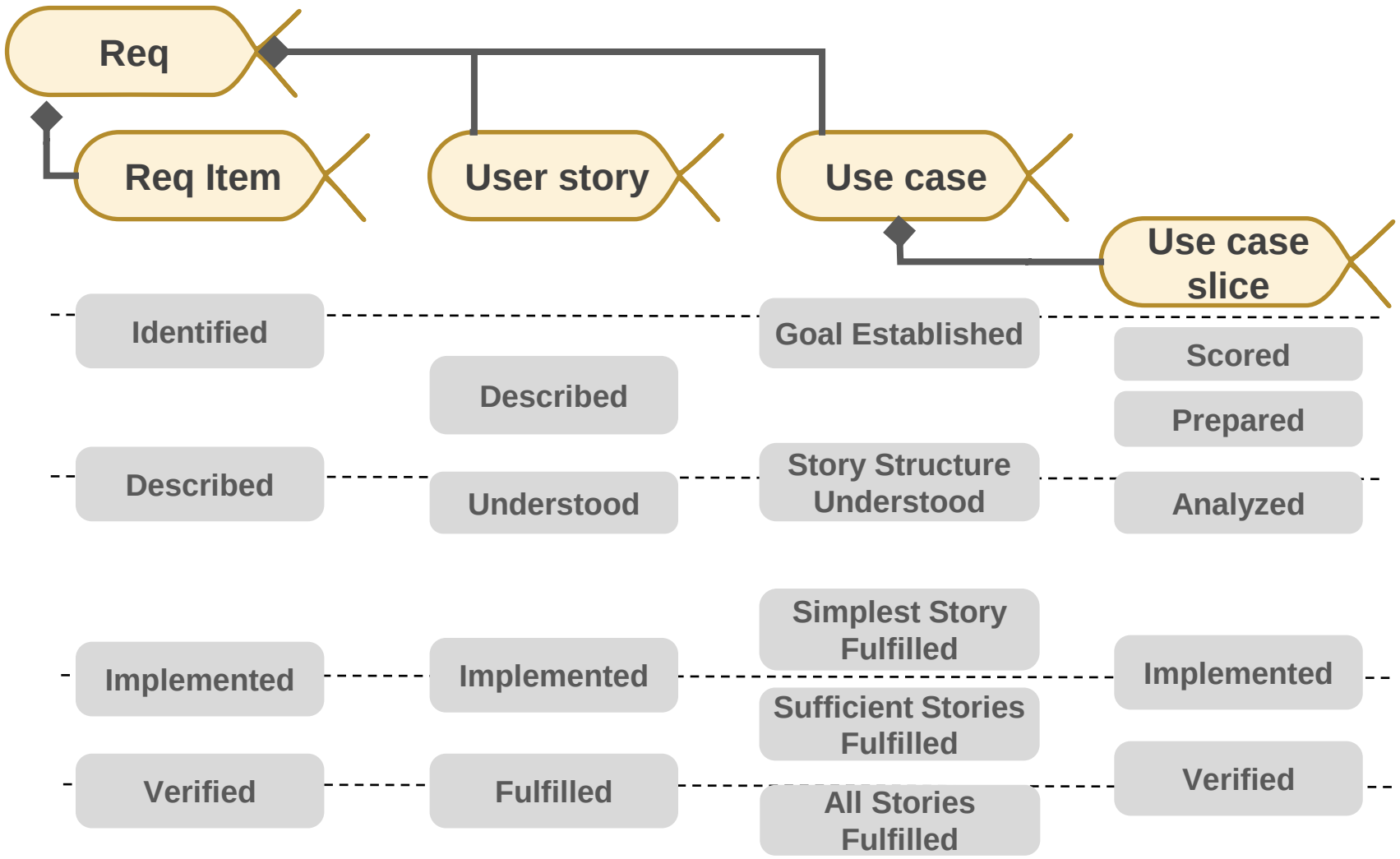
Требования удовлетворены

Декомпозиция требований

CUSTIS



User story и use case



Выбирайте метод из ситуации

CUSTIS

- Представляйте весь спектр разнообразия методов
- Помните: соответствие требованиям не гарантия успешного внедрения
- Для небольших сервисов или систем хорошо подходят легкие методы
- Для сложных и долго развивающихся систем лучше использовать DDD
- Не забывайте при проектировании замены убрать отпечаток старой системы

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