Постановка от модели бизнеса до детального дизайна требований: как делать и кому

Максим Цепков

IT-архитектор и бизнес-аналитик

Навигатор и эксперт по миру Agile,
бирюзовых организаций и Спиральной динамике

http://mtsepkov.org

На сайте много материалов по архитектуре и анализу, Agile и менеджменту самоуправления, и моделям soft skill, мои доклады, статьи и конспекты книг

Ижевск, UIC.dev
7-8 октября 2024

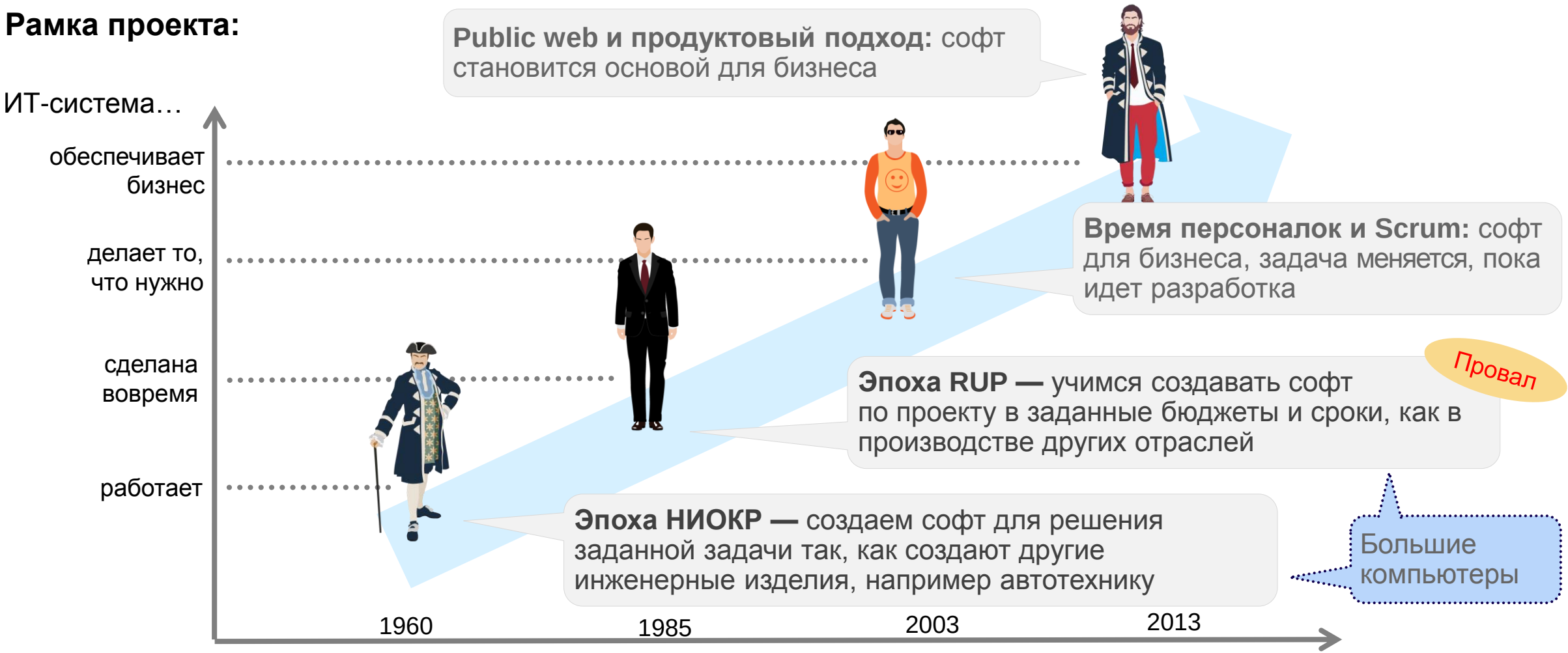
В чем проблема?

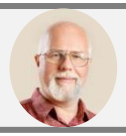
- Постановка – документы, которые мы пишем и обсуждаем до кодирования
- Есть широкий спектр способов работы с постановками
  - От полного отсутствия – услышали задачу и быстро сделали
  - До длинного процесса из многих стадий: бизнес-модель → требования → дизайн
  - Включая промежуточные варианты и различные формы документов
- Какой использовать в проекте?
- И что именно делает аналитик, а что другие при работе над постановкой?

Многие убеждены, что есть идеальный вариант,
и страдают, что в их проекте выбран неверный

Вариантов много, ведь проекты – разные

Рамка проекта:





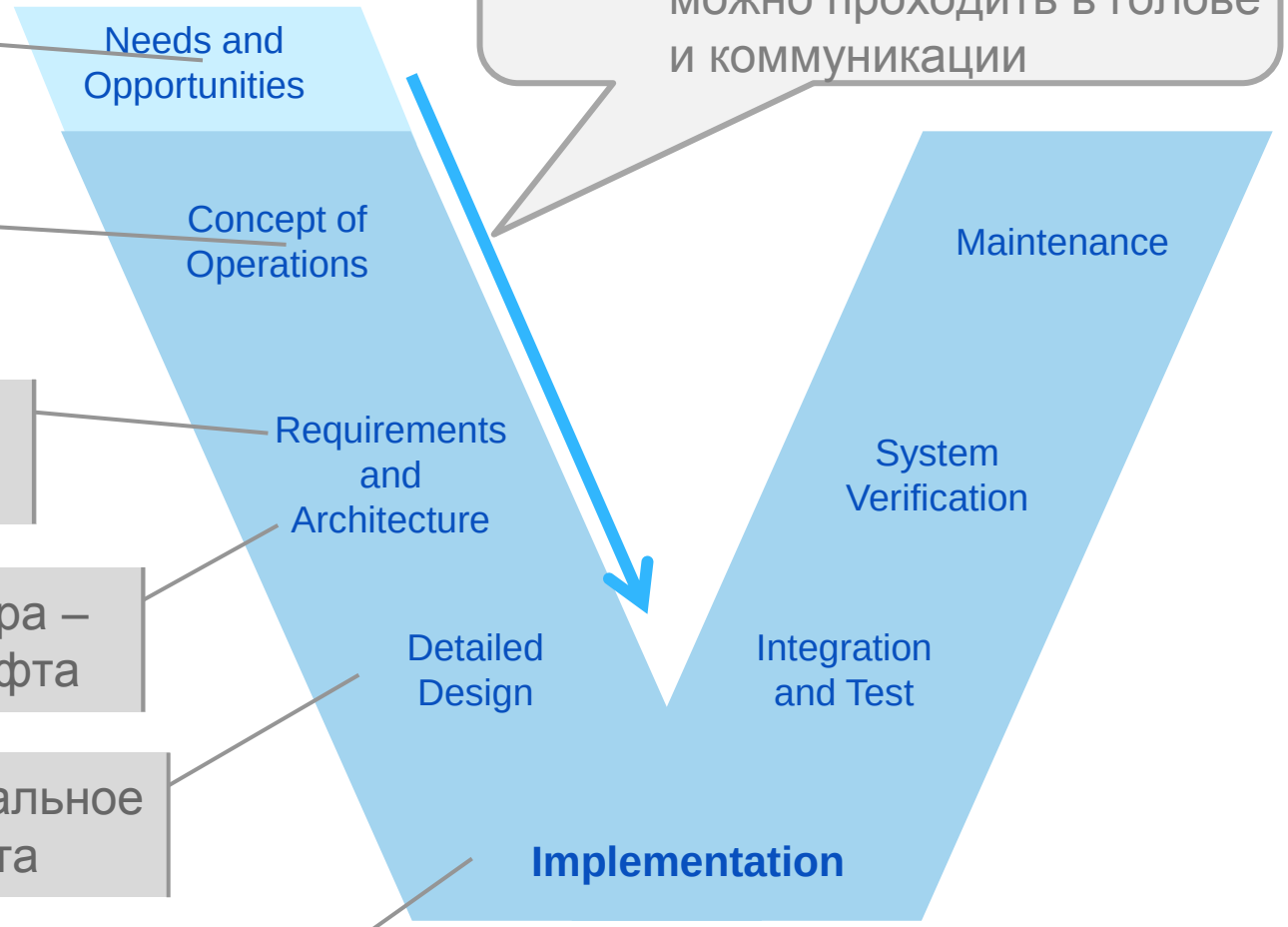
Каждая эпоха порождала свои учебники, эпоха ушла, а они – остались

Путь проекта – V-модель

Этот путь не обязательно требует документов, его можно проходить в голове и коммуникации

1. Начинается с идеи сделать хорошо некоторым людям с помощью софта
2. Далее надо придумать, как они будут действовать, используя софт
3. И сформулировать требования к софту, исходя из способов работы
4. Затем проектируется архитектура – принципиальное устройство софта
5. Затем – дизайн, детальное устройство софта
6. И реализация в коде

Различный уровень заказа



[V-Model (Wikipedia)]

Легкие варианты постановок

Работа без постановок (и аналитиков)

Inhouse и разработка продуктов могут обходиться без постановок

- Разработчик или группа поговорили с заказчиком
- Поняли, что надо сделать – и сделали
- **Быстро** показали результат, устранили замечания, пошли дальше

Уровень заказа может быть любой

Что для этого нужно?

- Освоение разработчиками, хотя бы некоторыми, бизнес-области
- Готовность разработчиков **вести коммуникацию**
- Работа разработчиков в ходе внедрения продукта

Такой способ предлагает минимальный Scrum

Бизнес

User story
или аналог

Concept of
Operations

Maintenance

Requirements
and Architecture

System
Verification

Дизайн
на салфетках

Detailed
Design

Integration
and Test

Короткие итерации, демонстрации и поставки версий обеспечивают соответствие софту потребностям бизнеса

Код

Implementation

Команда

Схема рабочая, но есть проблемы
при большой сложности софта

Где проблемы жизни без постановок

- У заказчика может отсутствовать человек, который расскажет, что нужно – это требуется собирать исследованием
- Языки разработчиков и заказчика могут существенно различаться
- Способ требует высокого уровня доверия между командами
- Если разработка идет долго, теряется целостность архитектуры и дизайна, появляется эффект спагетти-кода, который невозможно развивать

Удержание замысла и целостности

- Концепция системы – цели создания и верхнеуровневые полагания
- Метафора системы – эффективная практика, если получается придумать
- Архитектура системы – основа конструкции

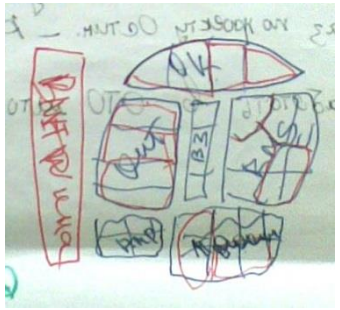
Я говорил о разных метафорах в докладе «Визуальные модели корпоративного приложения» осенью 2018 на AnalystDays#9

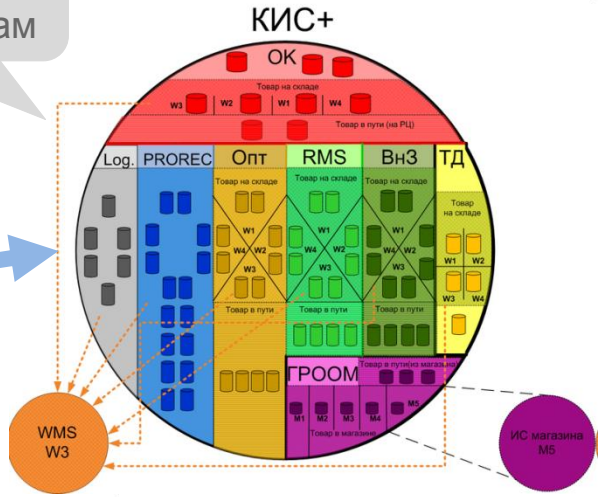
Необходимо

- Определить набор схем и описаний, поддерживаемых актуальными
- Порядок актуализации документов
- Цели, для которых выполняется актуализация и проверять их достижение

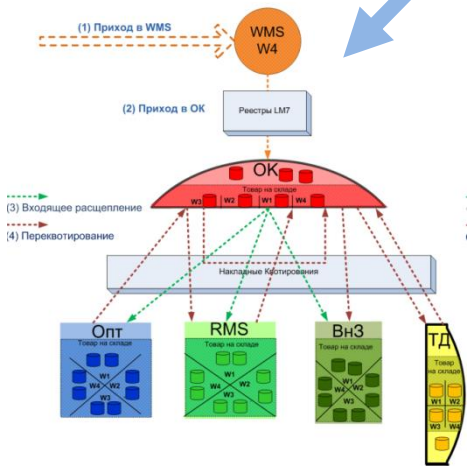
Варианты удачных метафор из моего опыта

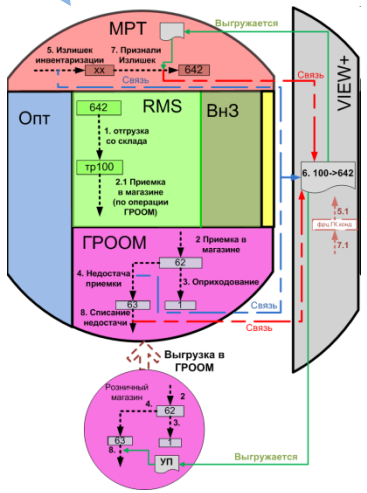
Сложная структура деления товарного запаса по системам



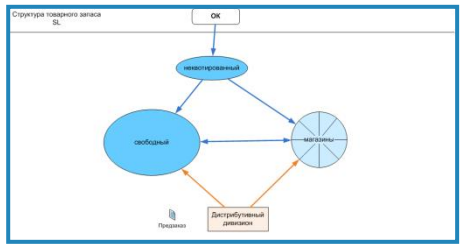


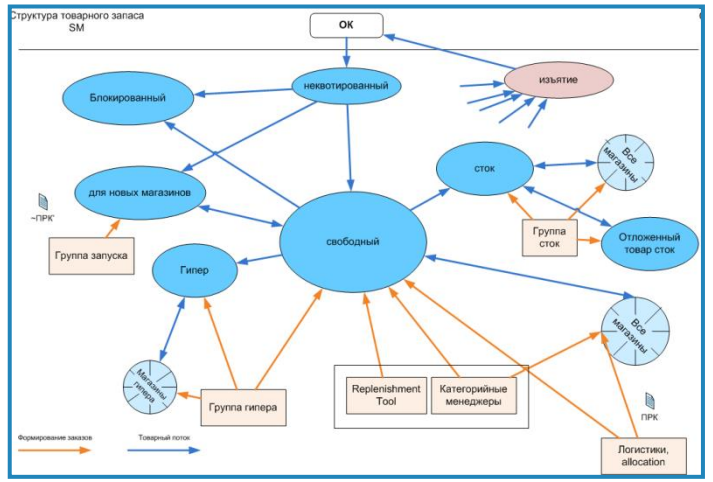
1 фаза



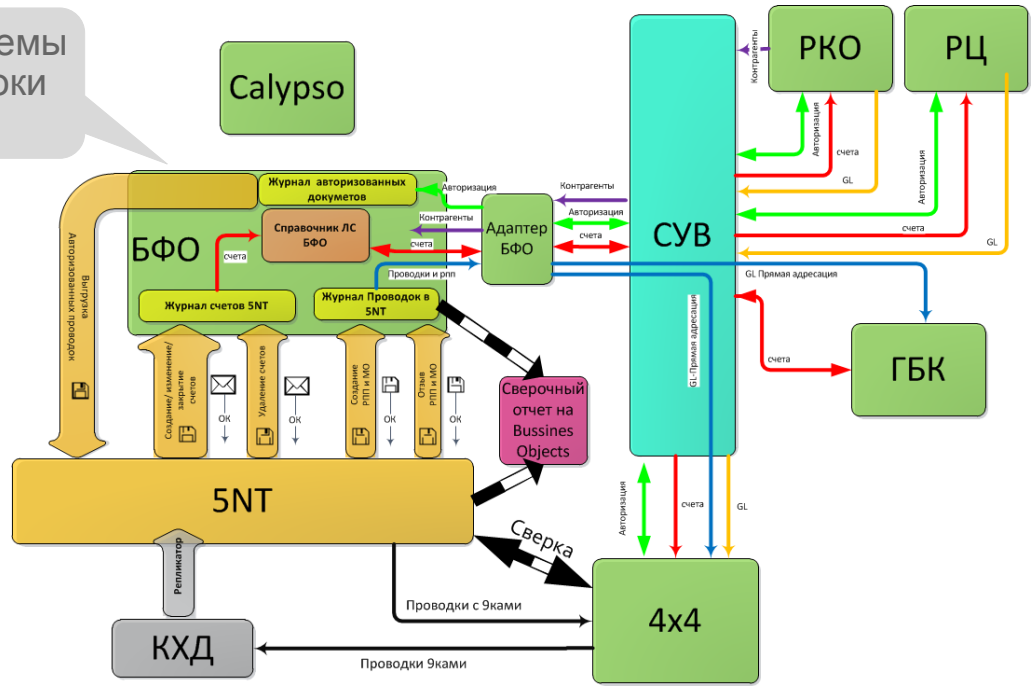


Простая структура деления товаров по ответственным





Роль системы через потоки данных



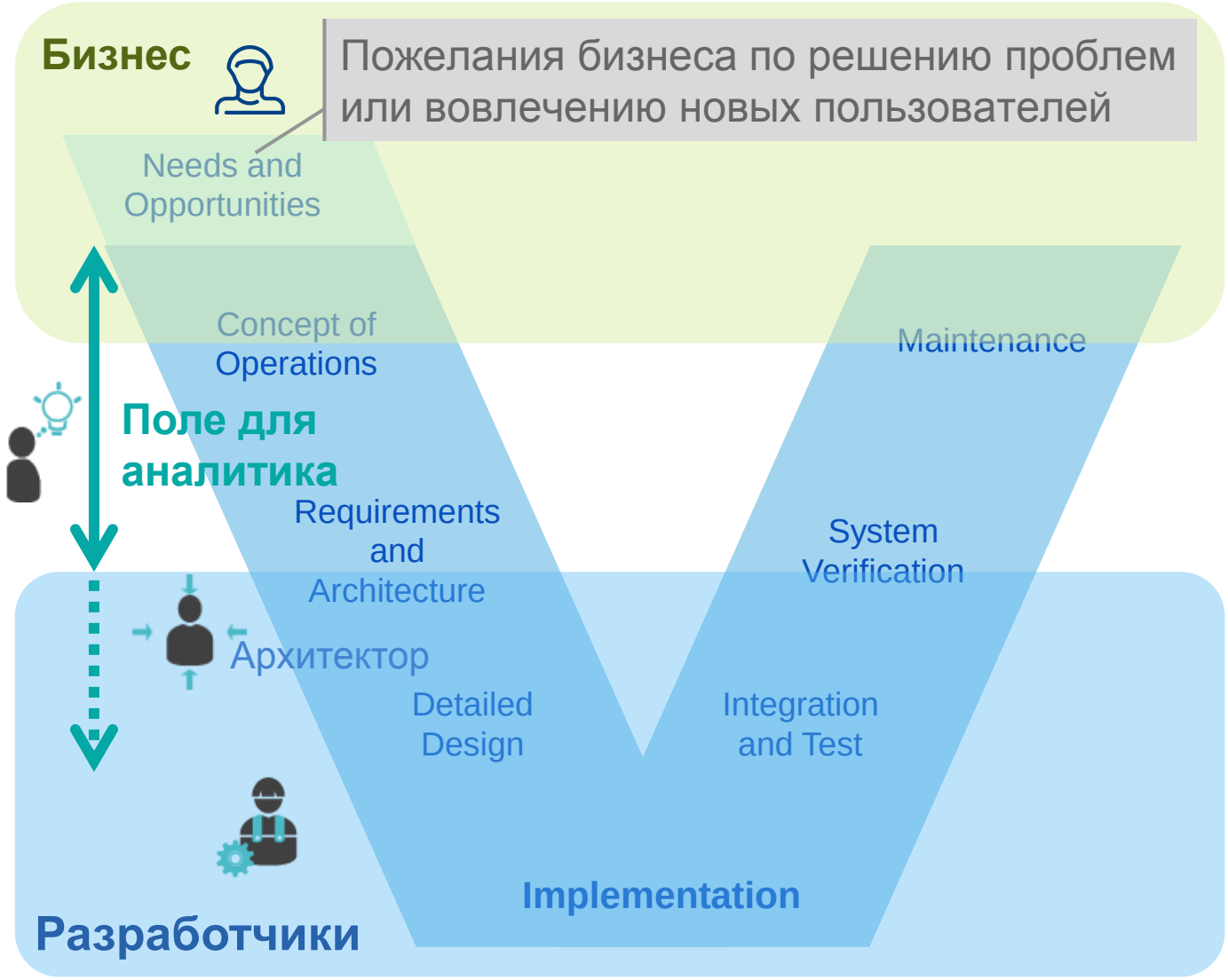
Поле для аналитика

Разрыв между бизнесом и разработкой

Что может делать аналитик?

- Описание бизнеса: интервью с заказчиком, описание бизнес-процессов и требований
- Проектирование и согласование модели объектов
- Проработка и согласование user story, use case, макетов интерфейсов и их аналогов
- Тестирование результата

Артефакты – различны!



Какие нужны знания о предметной области?

- Нужны объекты, с которыми работают пользователи
  - В виде словаря понятий, типов документов и сущностей в справочниках
  - В виде полноценной объектной модели
  - Может быть ссылка на отраслевое описание или распространенное приложение
- Нужно ли описание бизнес-процессов?
  - Обязательно, если при внедрении предполагается их изменение – As Is и To Be
  - Если автоматизируются имеющиеся процессы, можно без их описания, ограничиться ролями и функциями пользователей, которые описывать через use case или списком
- Альтернативные способы получения информации:
  - User story и story mapping
  - Event Storming дает событийную модель работы пользователей
  - Workflow документов с ролями пользователей

Передача информации о бизнесе и пользователях

- Есть ли лица, которые готовы заказать или нужно исследования?
- Каков способ коммуникации для получения знаний:
  - Интервью и работа с регламентами и формирование документа-описания?
  - Интерактивные встречи – story mapping или event storming?
- Способ передачи общей информации о бизнесе разработчикам:
  - Участие в общей интерактивной встрече с заазчиком
  - Краткий документ и лекция
  - Метафора системы, если получилось придумать – эффективная практика XP
  - Модель процессов или событий
- Способ передачи детальной информации по конкретной фиче:
  - Описания бизнес-процессов
  - Описания сценариев работы пользователя
  - User story с целями пользователей
  - Контекстная диаграмма в c4 model

Ведение документов

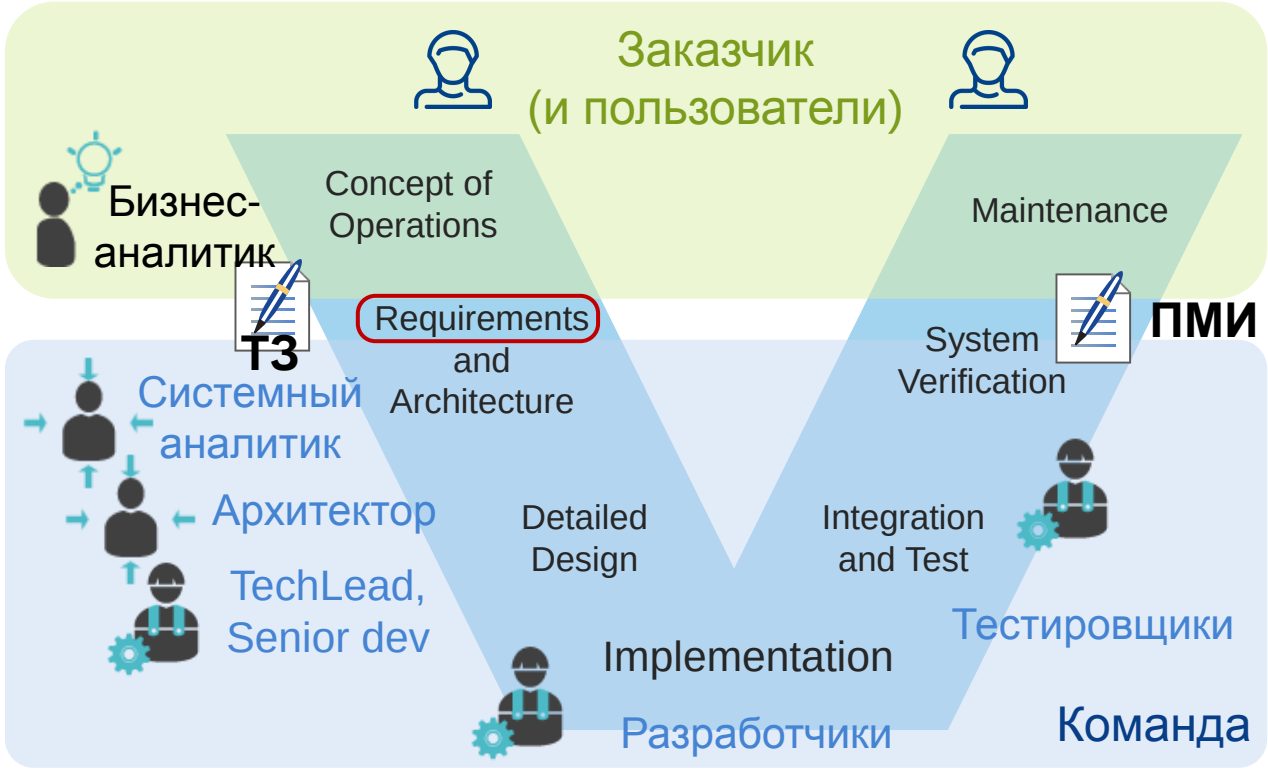
- Документ обеспечивает коммуникацию, распределенную по времени: кому он адресован, эффективна ли коммуникация, не дорог ли документ?
- Важно удобство коллективной работы над документами, есть много средств, выбор – с учетом интересов **всех** участников проекта
- У документа нет автора, важна возможность **и культура** внесения правок

У меня был отдельный доклад «Управление знаниями: какие документы нужны и что в них фиксировать» на Teamlead-2018

Требования – что это такое и нужны ли они вам?

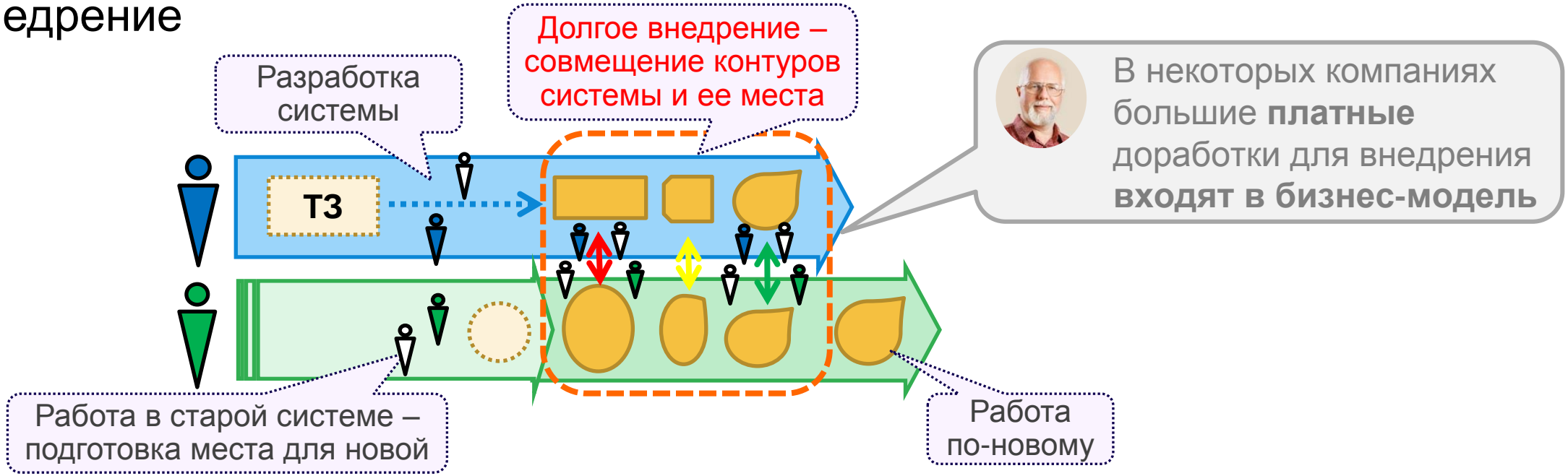
Требования – способ фиксировать контракт

- Заказчик формирует требования – **описание софта как черного ящика**
- Требования могут включать частичные описания архитектуры и дизайна
- Заказчик может активно участвовать и утверждать технические решения



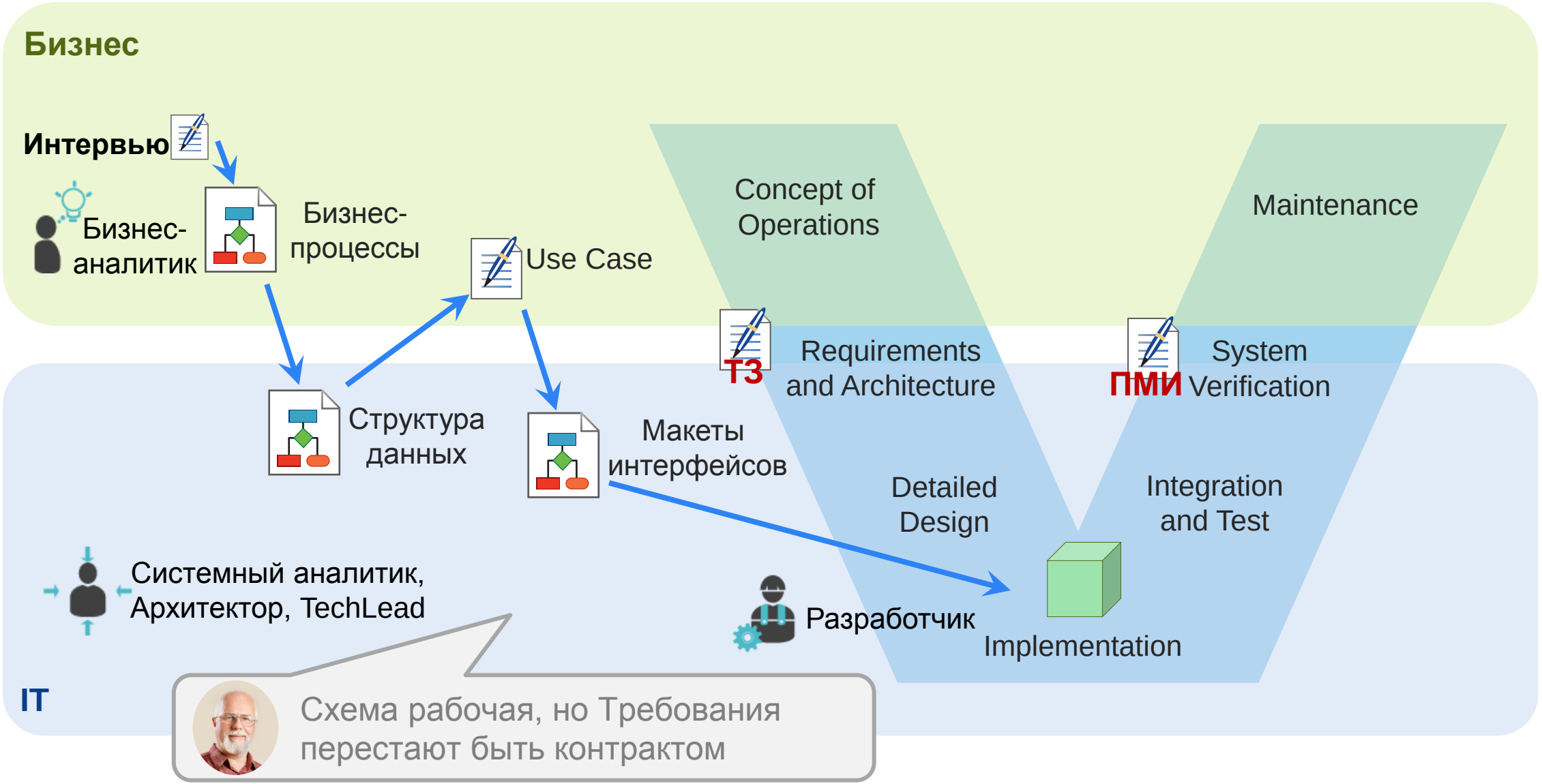
Проблемы контракта «от требований»

- Разработанная система не оправдывает ожиданий, это вызывает долгое внедрение



- Откуда заказчику взять требования на понятном ИТ-языке?
- Методологическая проблема: чтобы сформулировать требования к интерфейсу, надо спроектировать структуры данных

Два прохода: данные и интерфейсы



Структуры данных

Варианты проектирования

- ER-диаграммы хранения базы данных
- Диаграммы классов
- Транспортные объекты, передаваемые при интеграции
- API сервисов

От чего зависит ответ?

- Архитектура: сервисы или отдельное приложение
- Кто делает: аналитик или разработчик
- Согласуется ли с Заказчиком

Работа аналитика в case-средстве, из которого идет генерация кода – реальный вариант. Альтернатива: сравнение модели с реальным кодом

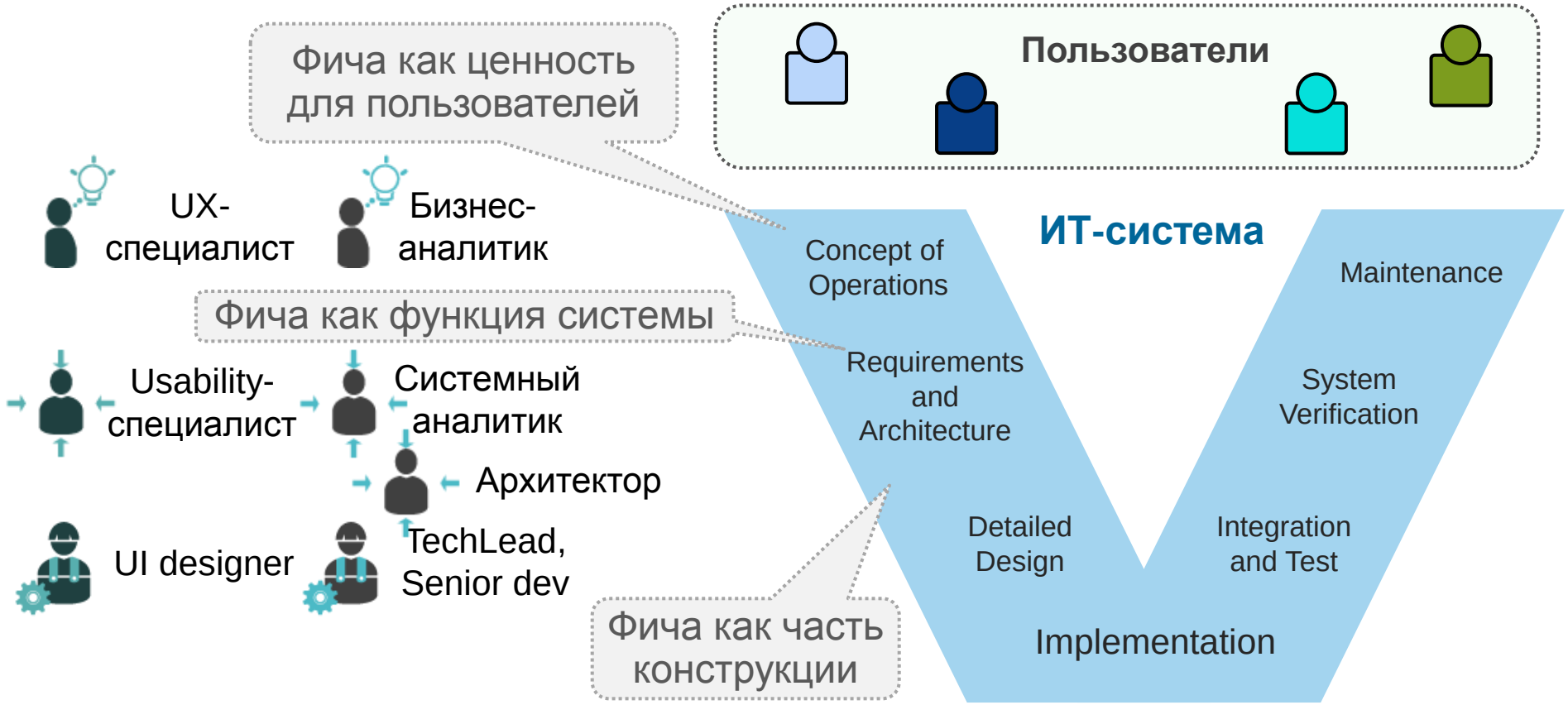
Варианты проектирования интерфейса

- Naked objects: витрины объектов и вызов методов над ними
- Макеты интерфейсов, демонстрируемые и согласуемые с заказчиком
- Проектирование на основе use case или user story, заказчику показываем уже готовые решения

Каково разделение труда между аналитиком, дизайнером и разработчиком?

- Кто и что именно обсуждает с заказчиком? Показываем ли ему текстовые описания, или проекты интерфейсов, или прототипы?
- Как устроен процесс, чтобы обеспечить технологичную доработку?

Аналитики и специалисты по UX-usability



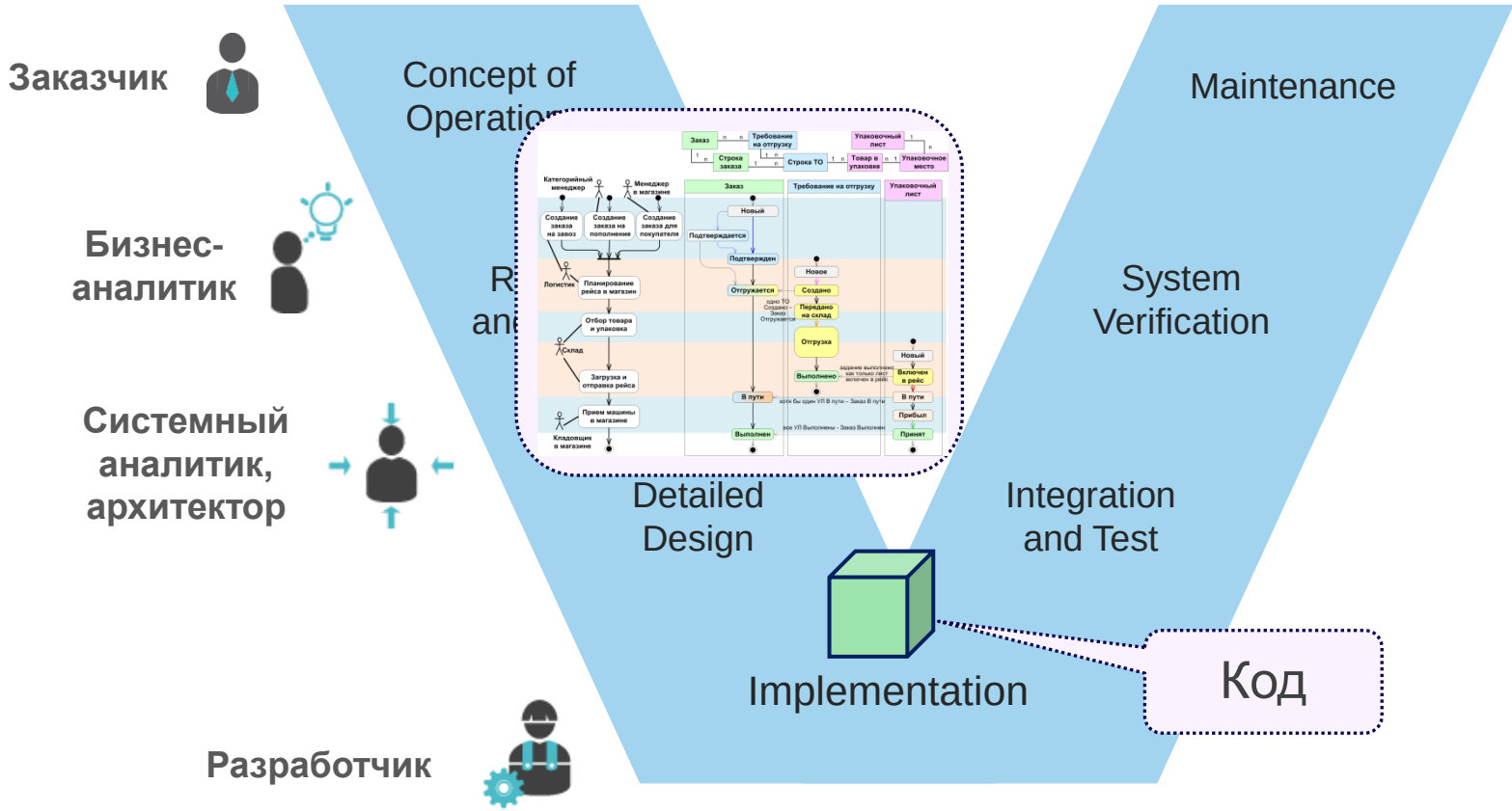
В реальных проектах такого детального разделения труда не бывает, есть аналитик или UX и разработчики, иногда – архитектор

Хороший доклад о разнице UX и аналитиков делал **Юрий Куприянов** на AnalystDays-2015 «Анализ требований с точки зрения UX»

Говорим с бизнесом о системе – DDD

Domain Driven Design

- **Единый язык**
  - На основе терминов предметной области
  - Понятен всем участникам проекта
- **Единая модель**, приложения и его встройки в бизнес
- Прозрачное отражение модели в код



У меня есть много докладов о DDD, последний «DDD: модели вместо требований 9 лет спустя (ЛАФ-2023)»

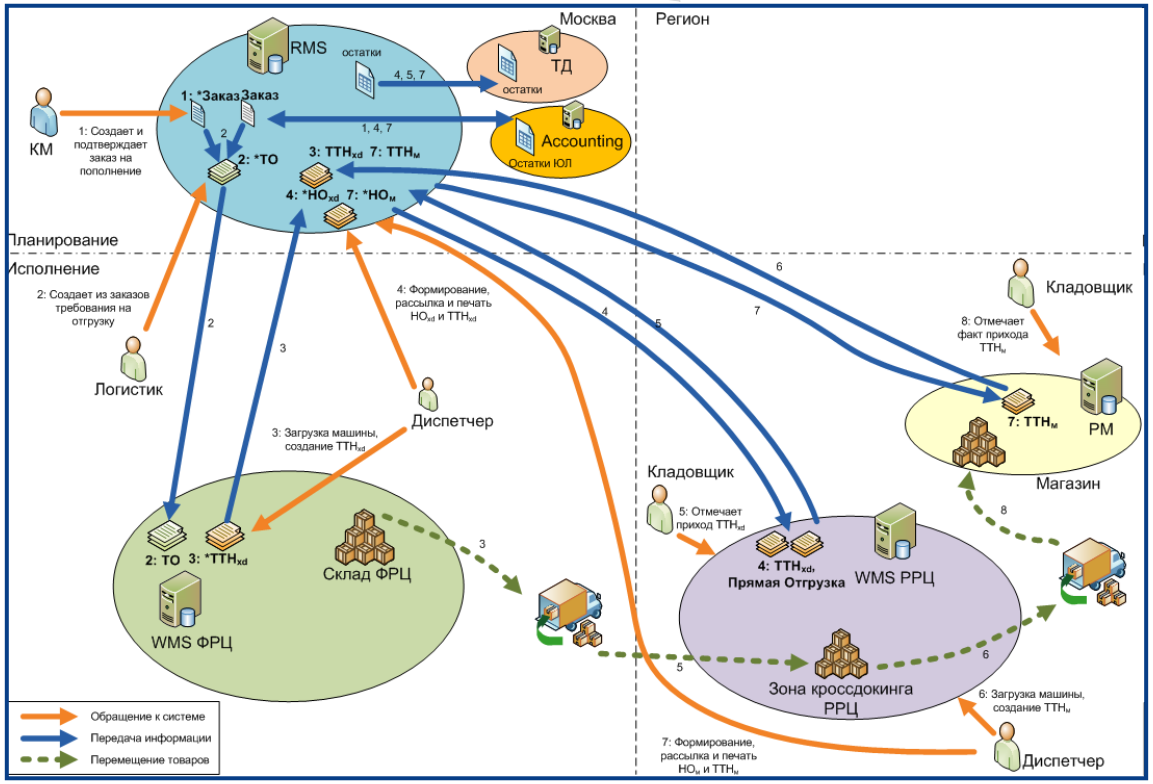
Пример – снабжение магазинов

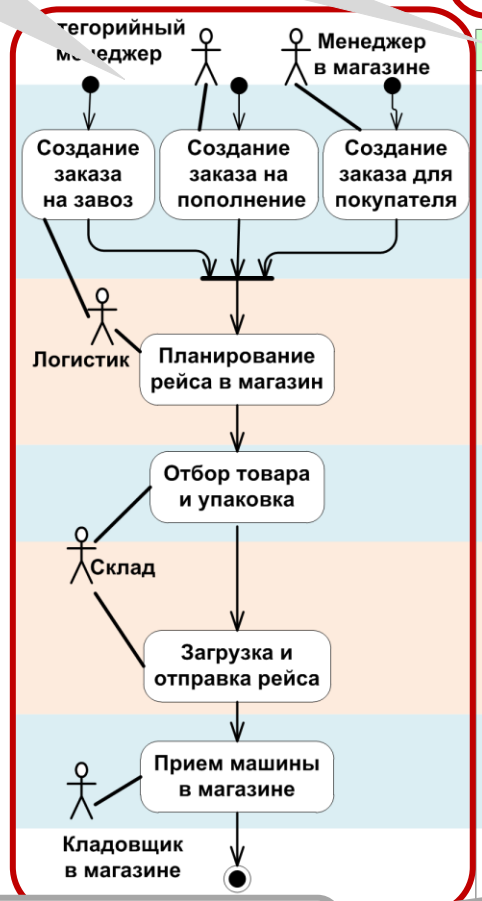
Неформальная схема деятельности и ее отражение в существующих системах

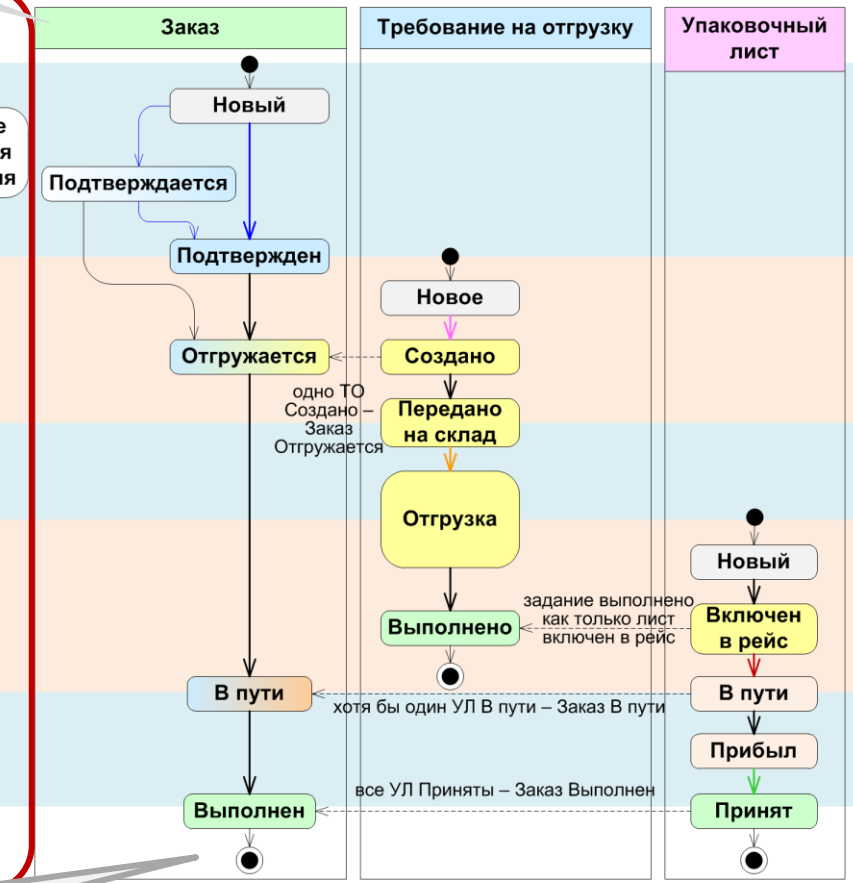
Бизнес-процесс – activity diagram

Состояния документов – state diagram

Объекты – class diagram

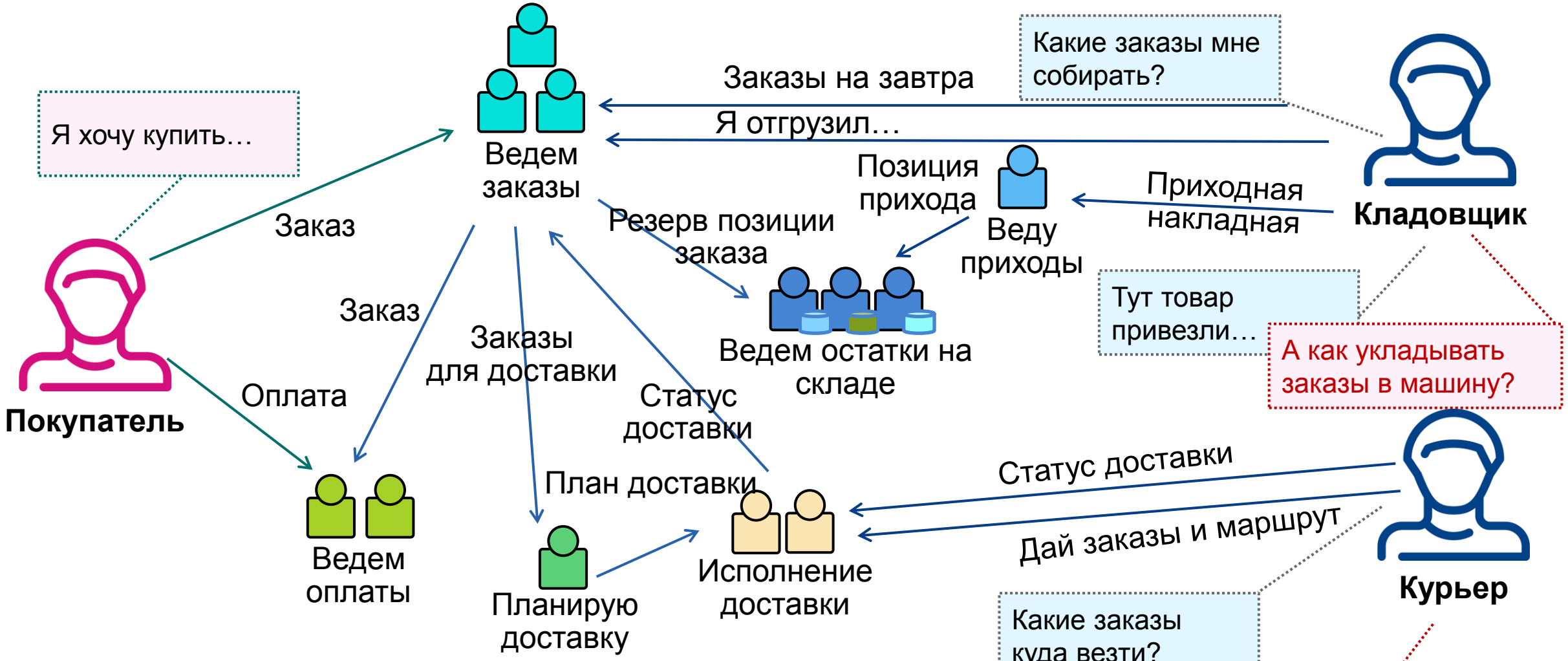






Если workflow документов прозрачно отражает бизнес-процессы, то можно обсуждать их сразу на такой схеме

Модель для сервисной архитектуры



Из доклада [«Визуальное проектирование масштабируемых приложений»]

Представления о бизнесе для фичи

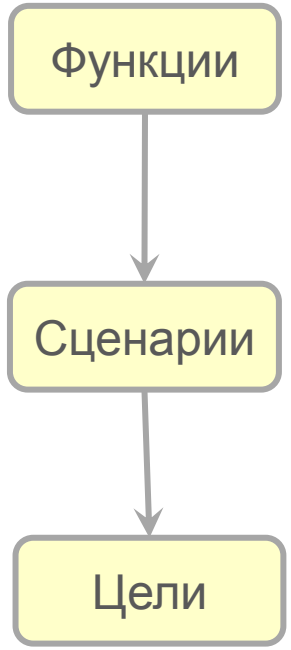
Три категории постановок

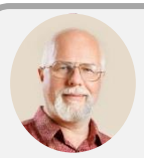
1. Разработка новой системы поверх фрагментарной и малой автоматизации:
   - выясняем модель бизнес-процессов
   - создаем модель системы
   - работаем над ее встройкой в процессы, изменяя их
2. Доработка существующей системы: проектируем изменения модели существующей системы и ее встройки в процессы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

Представить себя на месте пользователя

- Первоначально требования описывали функции системы и систему как целое
- Выяснилось, что сделанные так системы оказываются неудобны или непригодны в работе
- Use case – описание, как пользователь будет применять систему для решения своих задач
- User story – описываем один сценарий работы пользователя, **включая достигаемые им бизнес-цели**
- Контекстная диаграмма в c4 model – аналогична use case для фичи



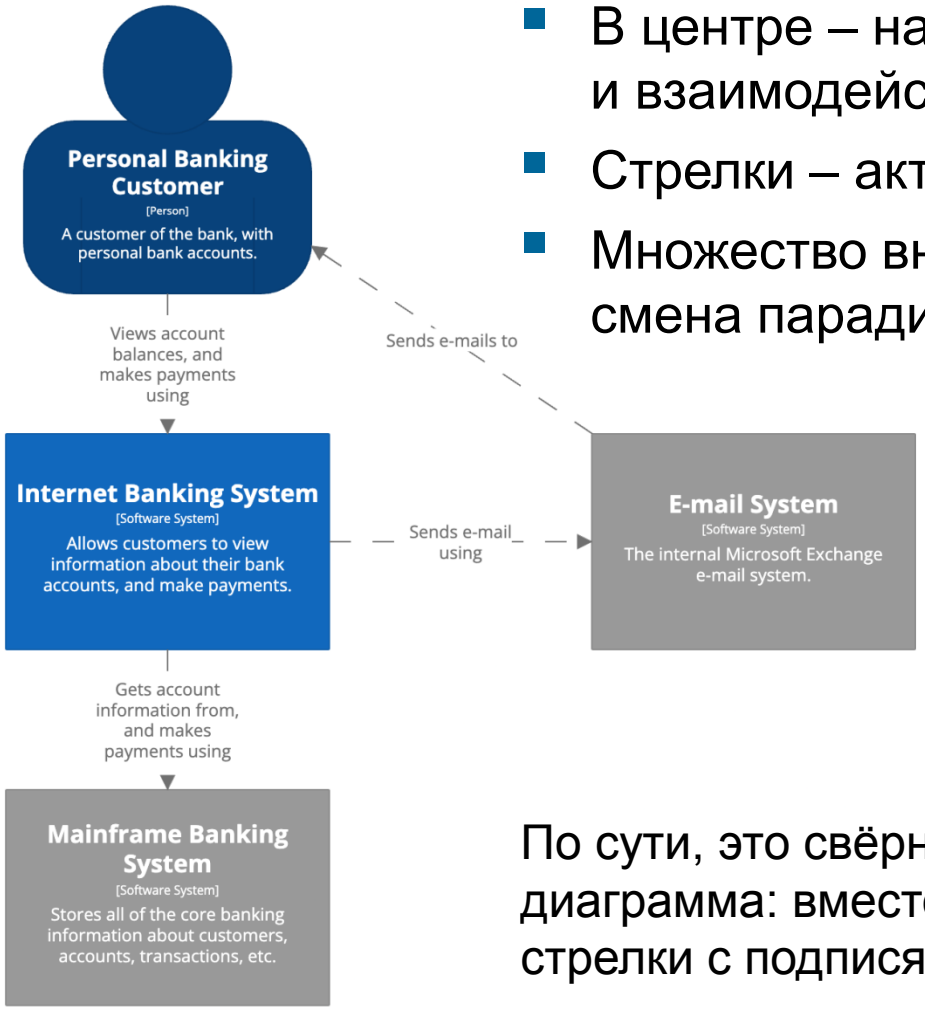


В процессе разработки принимается много решений, требующих представить себя на месте пользователя

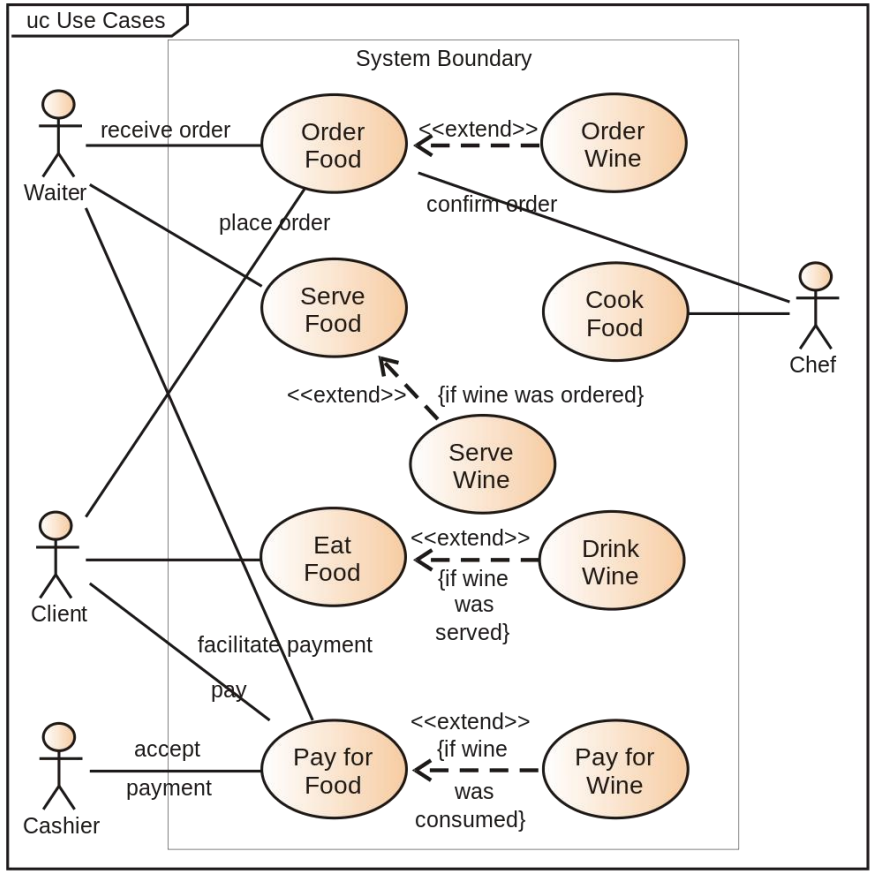
Use case или user story?

- Оба нужны, если мы хотим описать работу пользователя для реализации интерфейсов без контакта с заказчиком до демо
- Оба хорошо описываются на основе бизнес-процессов, на их основе можно делать тест-кейсы
- User story:
  - ☺ Фиксируют цели пользователей, а не только взаимодействие с системой
  - ☺ Хорошая единица работы: можно менять порядок для реализации по важности
  - ☹ Сложные случаи могут потребовать существенных переделок
  - ☹ Однако, сложные истории могут потребовать существенных переделок
- Use case:
  - ☺ Комплексно описывает взаимодействие, это можно заложить в реализацию
  - ☹ Плохая единица работы: в одном use case смешаны частые и редкие сценарии
  - ☺ Можно делить use case на slice – смотри Use Case 2.0 Ивара Якобсона

Контекстная диаграмма C4 model – система в окружении



- В центре – наша система, вокруг – пользователи и взаимодействующие системы
- Стрелки – акты взаимодействия
- Множество внешних систем – смена парадигмы на сервисы



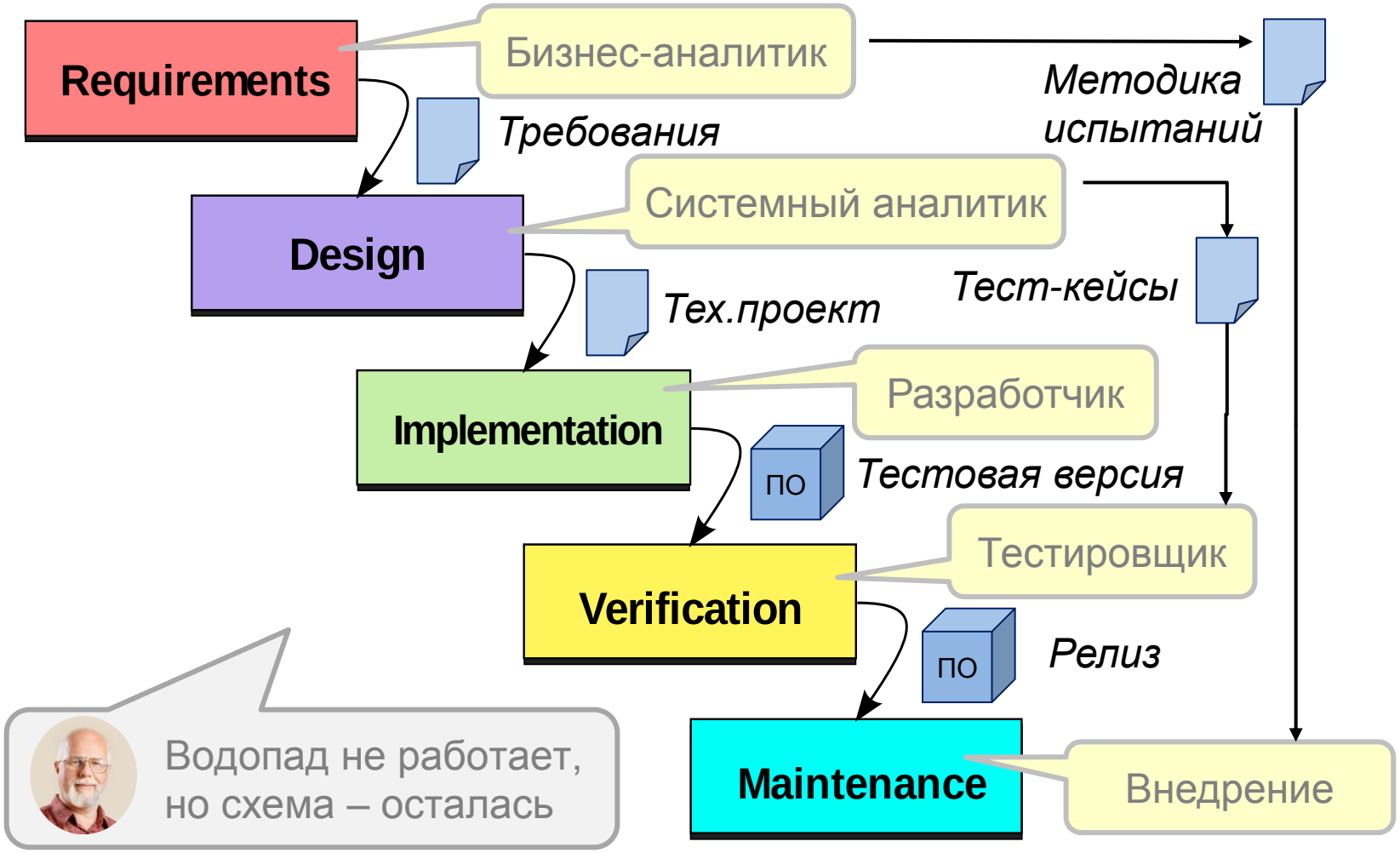
По сути, это свёрнутая use case диаграмма: вместо овалов – стрелки с подписями

Подводя итоги

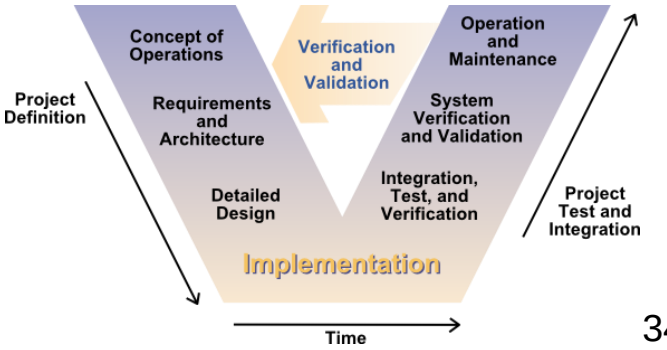
Какие вопросы надо решить?

1. Внешняя граница проекта и степень ее формализации в документе
2. Способ получения и описания информации о бизнесе и пользователях
3. Разделение труда в проектной команде:
   - Есть ли аналитики и архитекторы
   - Каковы границы ответственности и сопряжение с разработчиками и тестировщиками
   - Есть ли специализации аналитиков и разделение ответственности между ними
4. Способ инкрементального проектирования при развитии системы
   - Не слишком забегаем вперед: регулярные демонстрации продукта повлекут изменения в разработанном и в проекте остального
   - Не проектируем локально: хотя сначала реализуются простые случаи, реализация сложных не должны вести к полной переделке
5. Какими артефактами поддерживают проектирование, кто и как их использует?

Водопад – специализация по фазам проекта

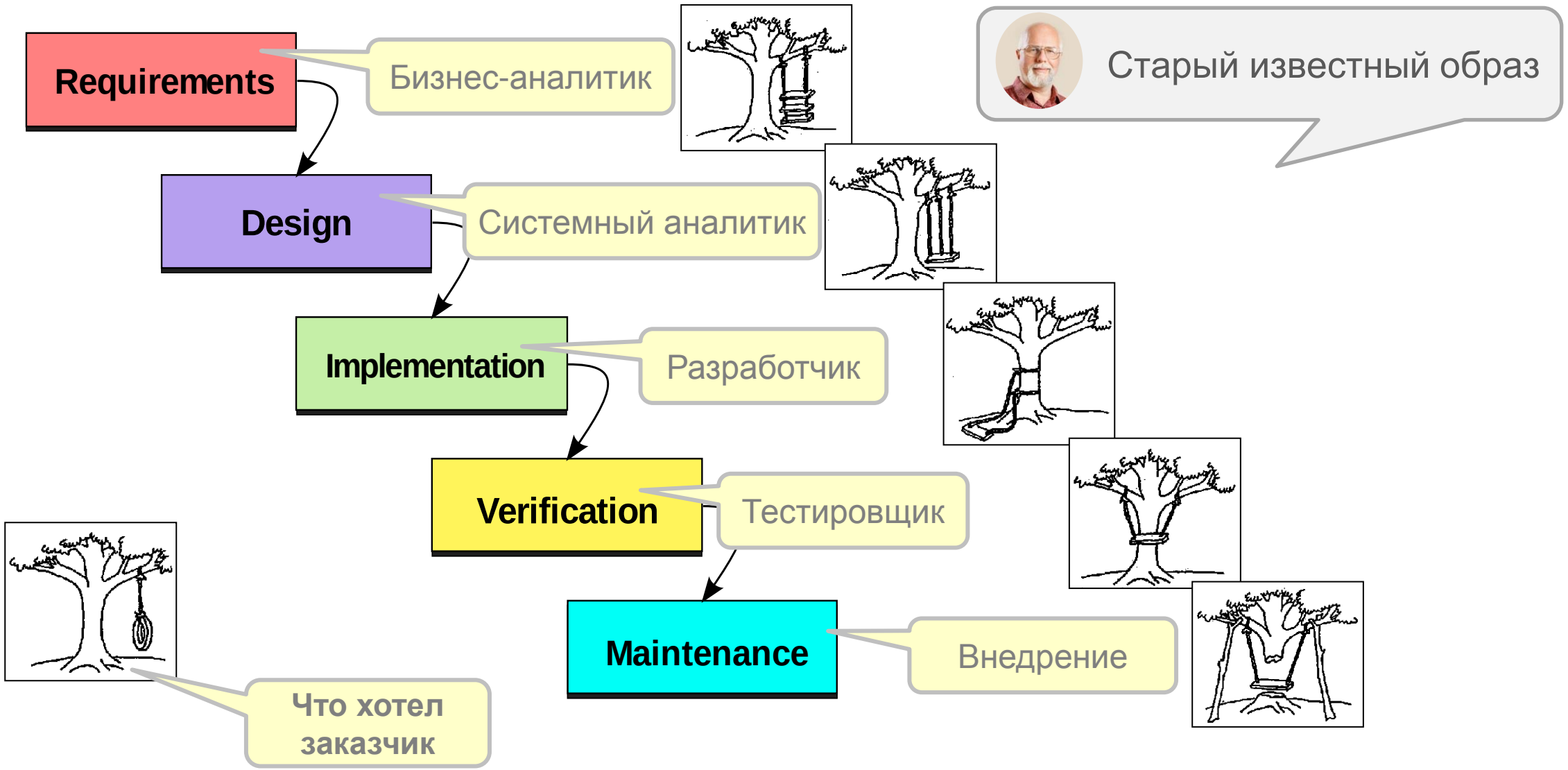


- Каждой фазе – своя роль
- Роли выполняются разными людьми или командами.
- Передача работы – **через артефакты** на отдельных языках.

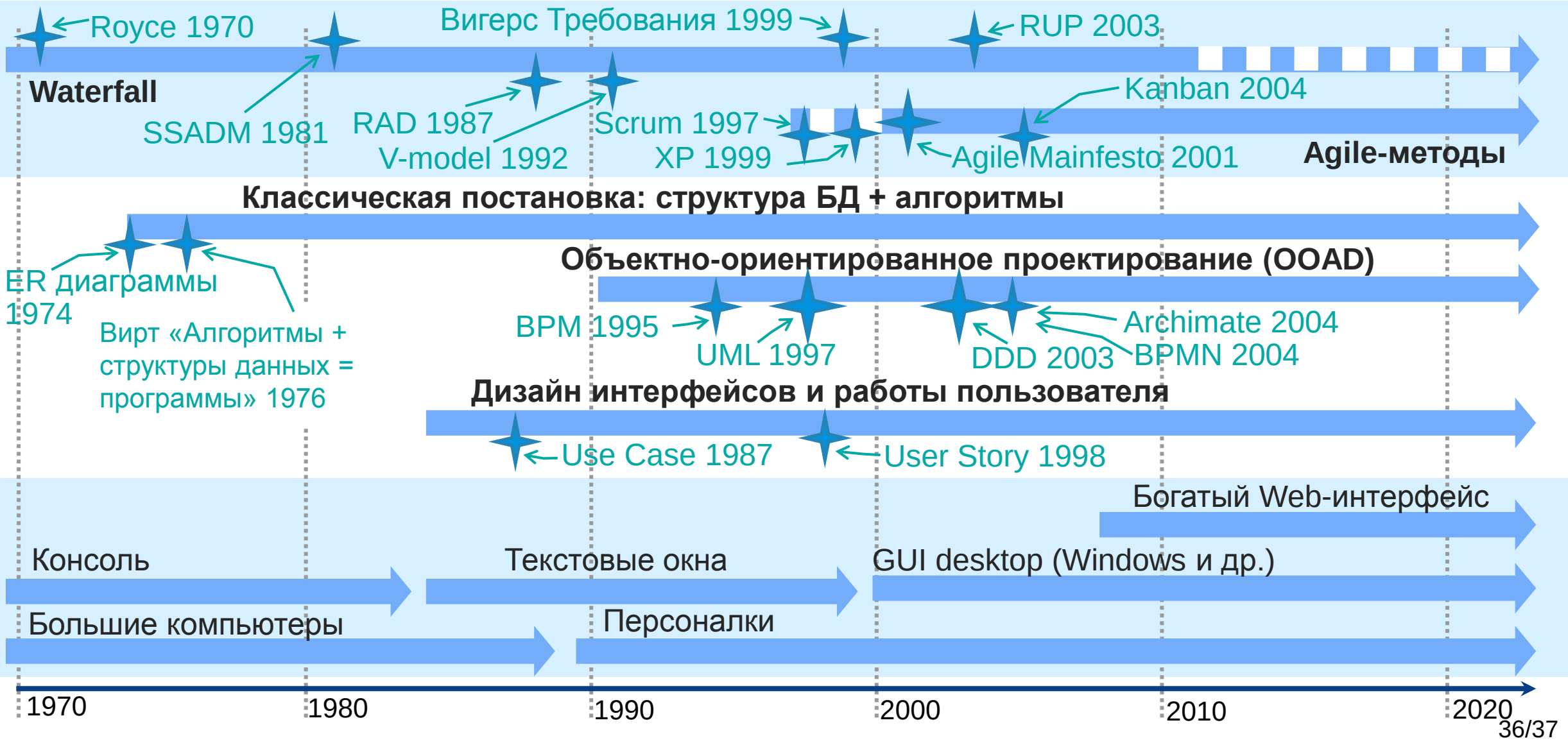


Модель водопада – http://en.wikipedia.org/wiki/Waterfall_model

Каждая передача искажает смысл…



Проектирование и его окружение



Выбирайте метод из ситуации

UIC dev

- Представляйте весь спектр разнообразия методов
- Каждая ИТ-разработка идет в своих условиях и потому способ работы с постановками сам по себе – объект конструирования
- Выбор определяет не только опыт команды, но и характер проекта: контракт с заказчиком, организация процесса, разделение ролей
- 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

Максим Цепков
http://mtsepkov.org
Tg: @MaximTsepkov

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