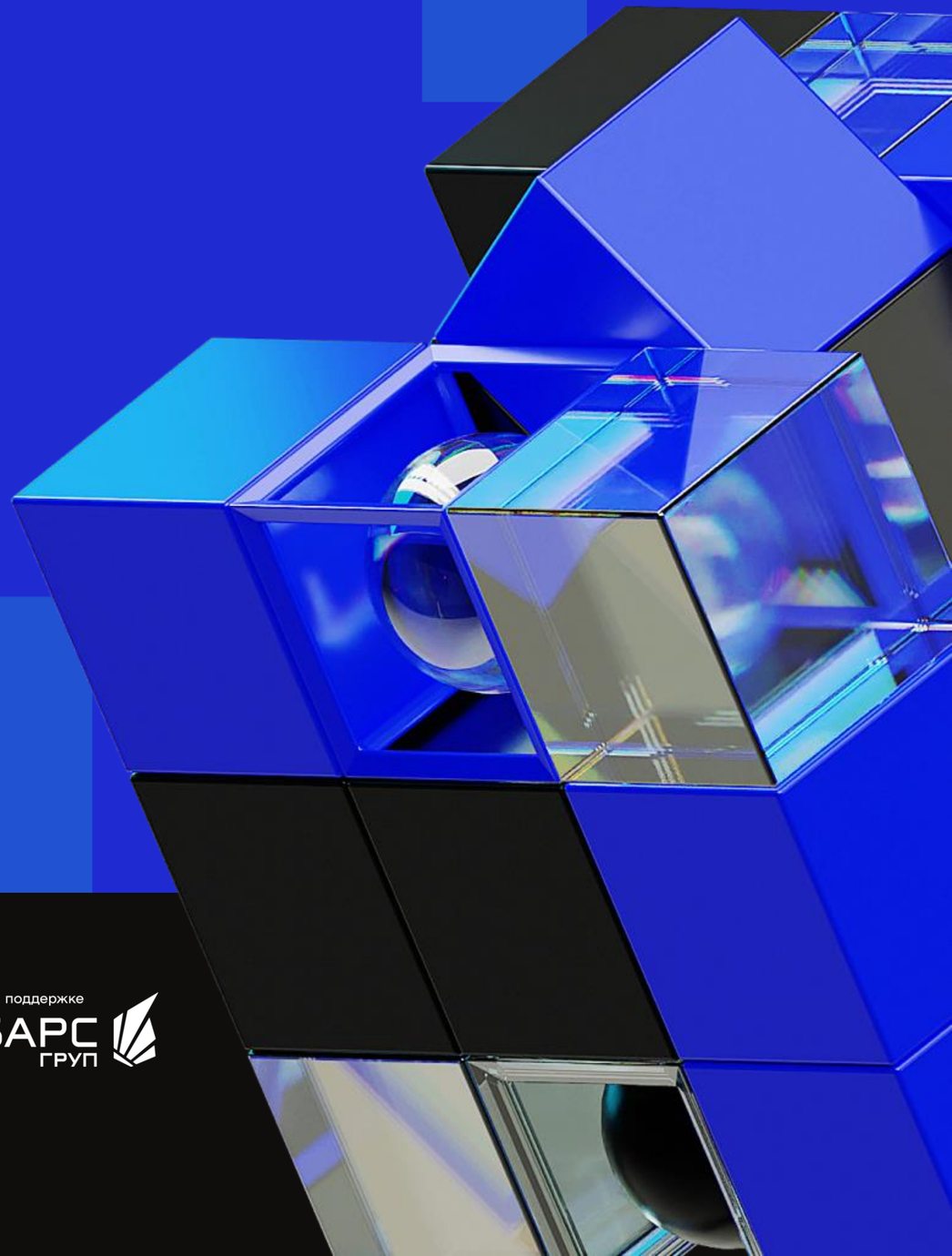
Up!Date

Образовательная
ИТ-конференция

Классика, user story, use case, DDD: как выбрать и начать использовать



Максим Цепков

Архитектор и бизнес-аналитик

KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП

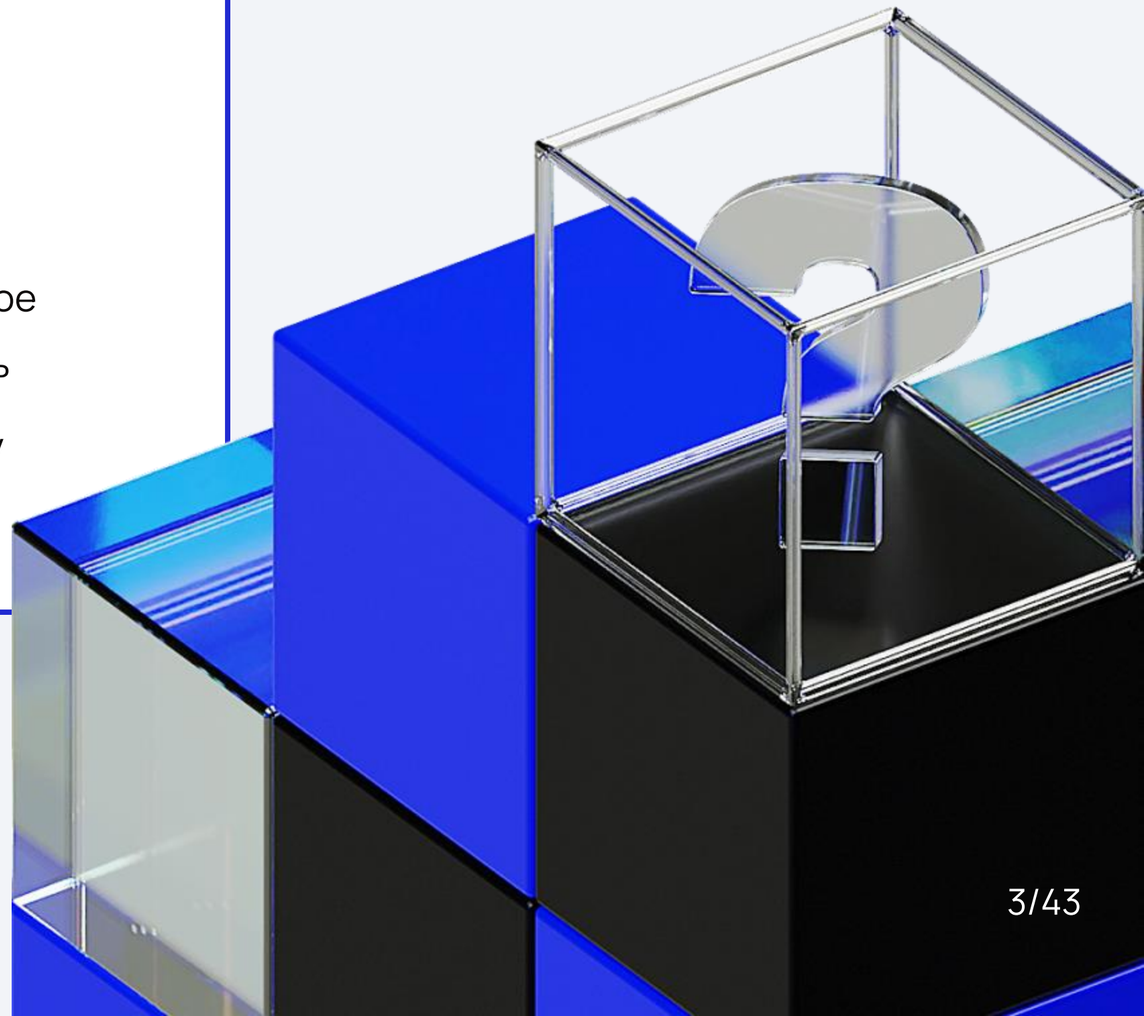
О себе

- Архитектор и бизнес-аналитик
- Навигатор в мире Agile, бирюзовых организаций и спиральной динамики
- mtsepkov.org

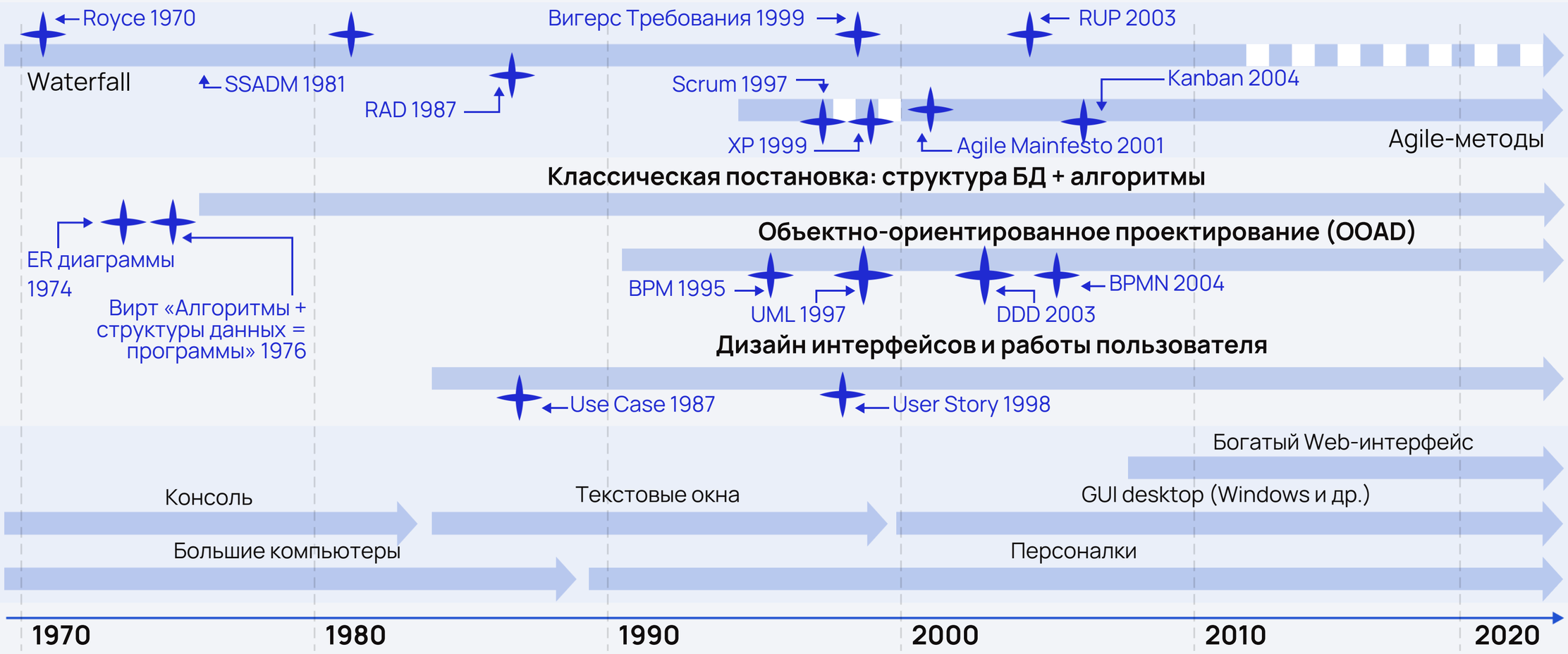
О чем этот доклад

- Развитие ИТ создало много методов проектирования
- Среди нет одного «наиболее правильного» на все случаи:
  - Метод должен соответствовать способу организации работ
  - Метод должен соответствовать технологиям реализации
  - Надо учитывать характер проекта: известные задачи или новое
  - Команда должна уметь применять метод, или ее надо научить
- Выбор по одному из критериев часто противоречит другому
- Просто выбирать привычное – ошибка

Доклад будет о том, как выбрать метод и как начать его применять



Проектирование и его окружение

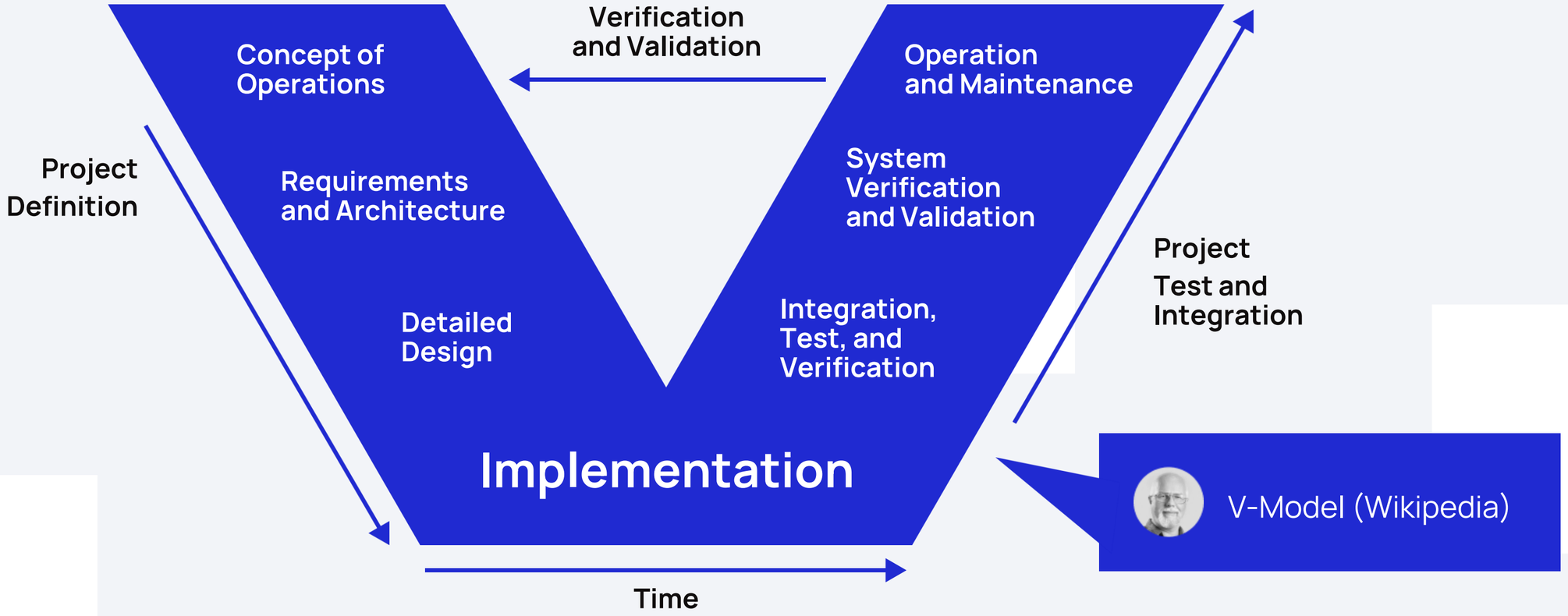


KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП

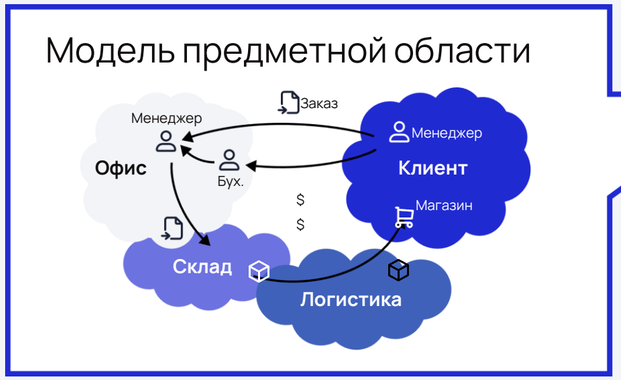
Классические постановки

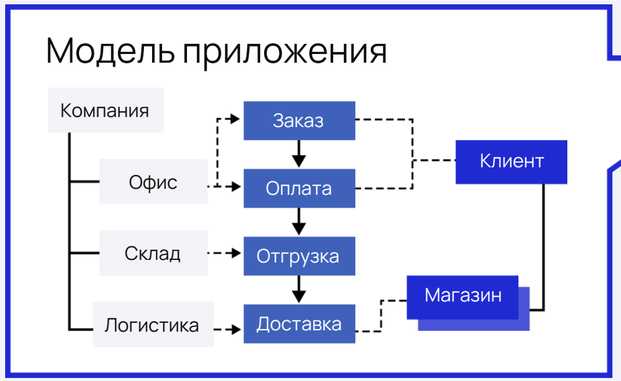
Путь проекта – V-модель

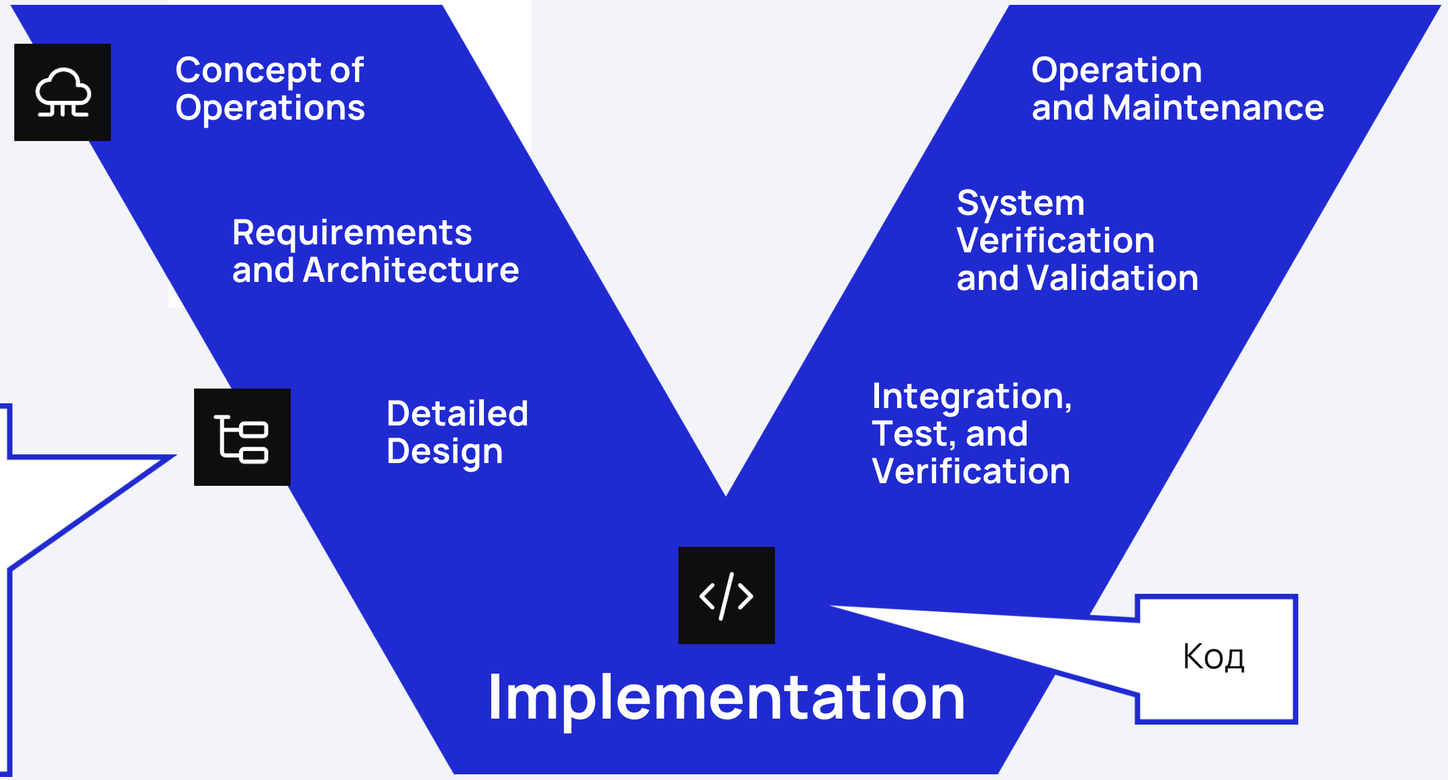


V-Model (Wikipedia)

Классические постановки







Шаги классической постановки

Разобраться в предметной области

01

Выделить сущности и их связи – логическая модель данных

02

Решить, как сущности будут храниться – физическая модель

03

Описать (и реализовать) алгоритмы обработки данных

04

Спроектировать (и сделать) интерфейсы, чаще CRUD-операции

05

Все шаги делает один человек – программист. Он должен разобраться в предметной области: нет проблем, если в одной, а если их много?

Процедурный и объектный подход: пример

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

Процедурный подход: проектируем структуру БД, интерфейсы и алгоритмы обработки. Иногда — процедуры API backend и API RPC

01

Объектный подход: типы и статусы объектов, методы бизнес-логики, интерфейсы в стиле **naked object** — витрины объектов и методы, REST API

02

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

05

Разница в том, где бизнес-логика — в методах объектов или отдельно

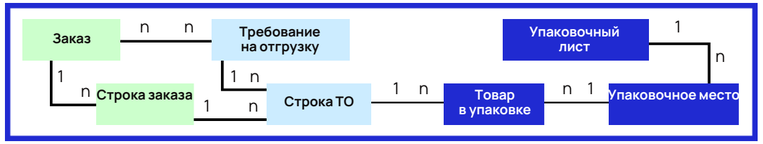
03

В реализации может быть анемичная модель транспортных объектов и соответствующие тем же объектам контролеры для бизнес-логики

04

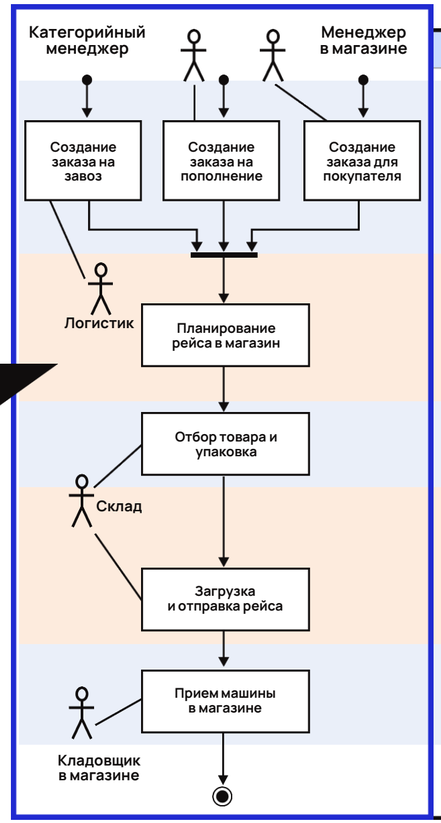
Классическая постановка

Снабжение магазинов



Выделяем объекты-документы

Бизнес-процесс: диаграмма активности

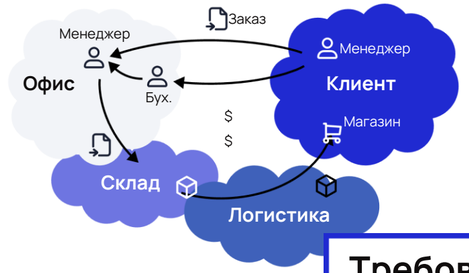


Переходы фиксируют выполнение бизнес-функции, а состояния соответствуют передаче по этапам обработки

Разделение труда: бизнес и ИТ

Интервью

Модель предметной области



Бизнес

Concept of Operations

Maintenance

Бизнес-аналитик

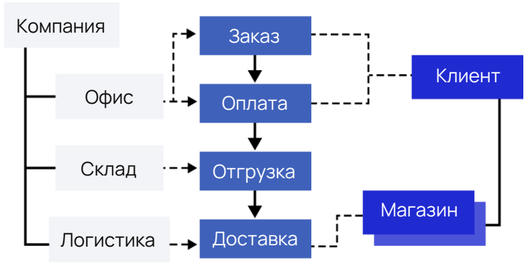
Требования: система как черный ящик

Requirements and Architecture

ТЗ

System Verification

Модель приложения



Системный аналитик, Архитектор, TechLead

Detailed Design

Integration and Test

Код

Implementation

Кодер

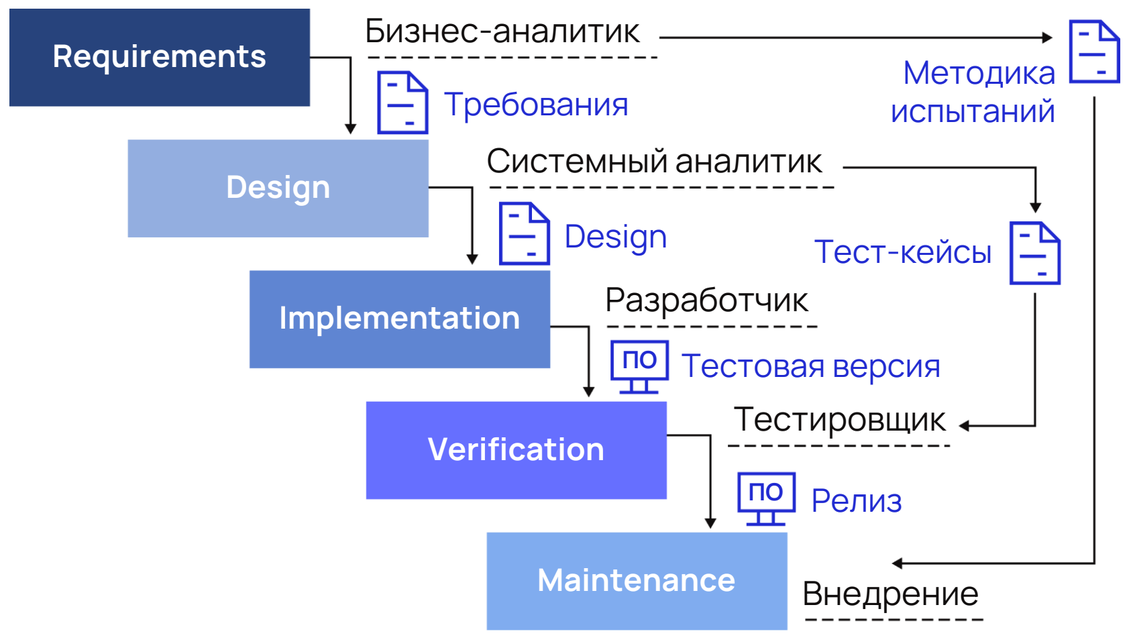
ИТ

KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП

Специализации и разделение труда

Водопад — специализация по фазам проекта



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

- Каждой фазе — своя роль
- Роли выполняются разными людьми или командами
- Передача работы формализована **через артефакты**

Классика + разделение труда

Бизнес-аналитик:

1. Описать работу пользователей через бизнес-процессы
2. Сформировать требования к софту
3. Спроектировать интерфейсы

Системный аналитик, архитектор:

1. Спроектировать структуры данных
2. Описать алгоритмы обработки

На каком языке писать требования, если сущности еще не определены?

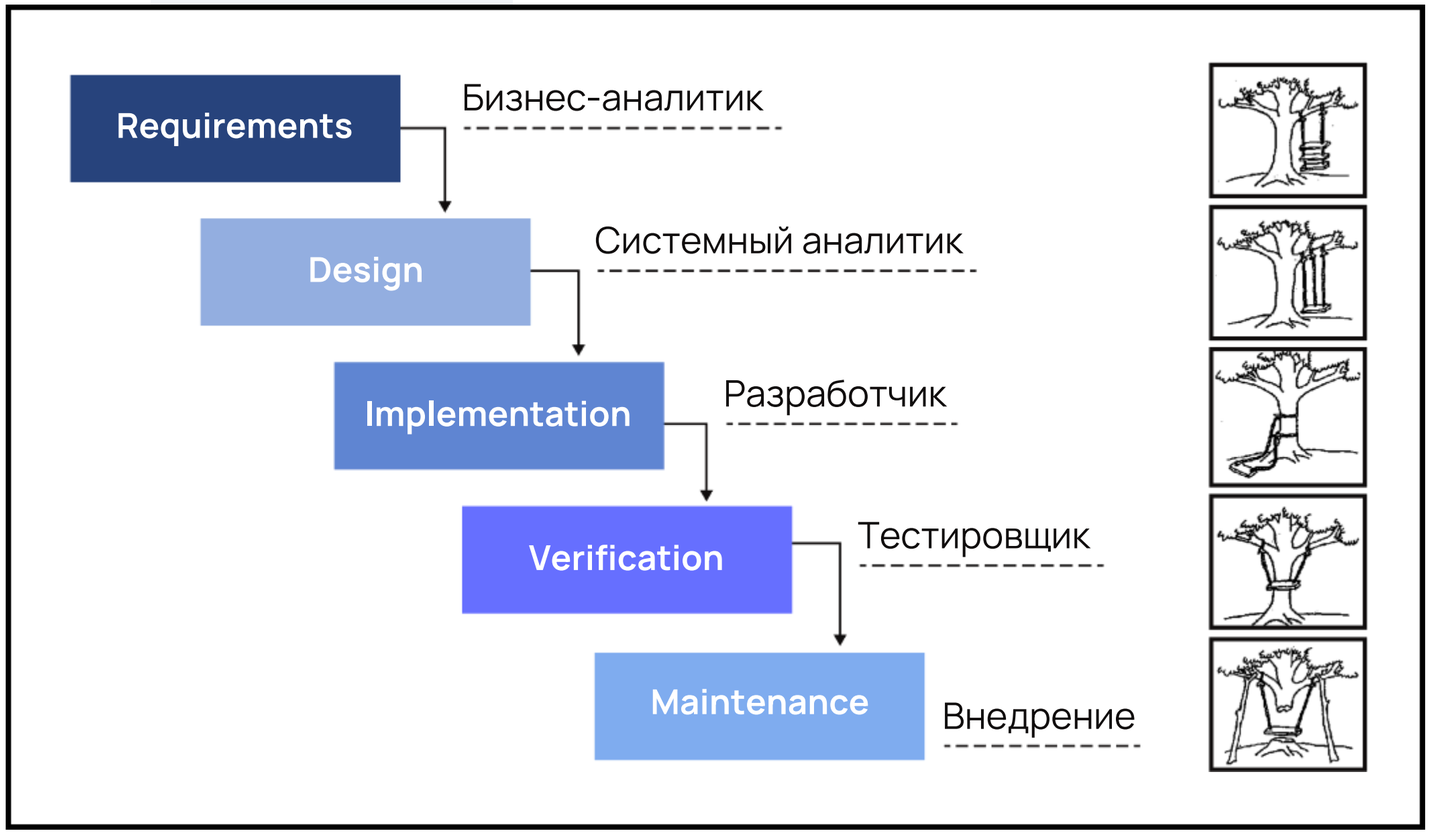
У бизнес-аналитика нет компетенции проектирования структур данных

Как проверить требования – ведь ИТ их реализует формально?

Каждая передача искажает смысл

Что хотел заказчик

Старый известный образ



Решение: требования как описание работы пользователя

Use case – сценарии работы пользователя с системой

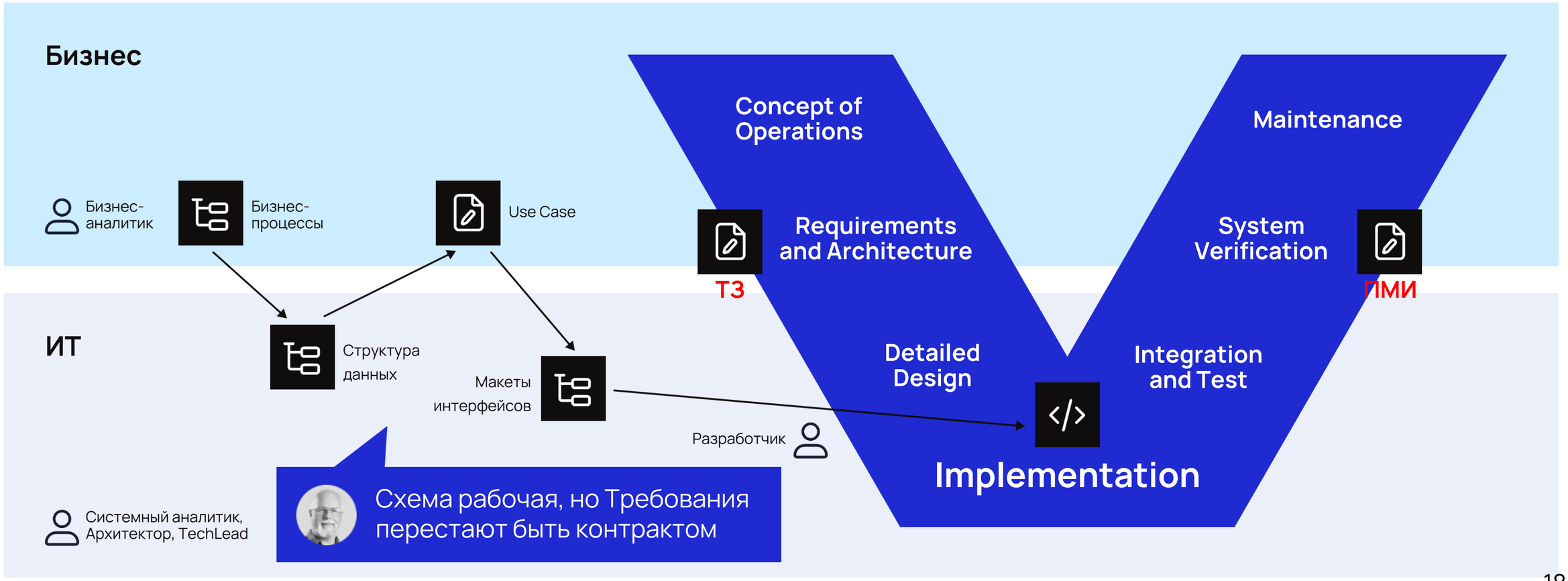
- ☺ Дает представления о потребностях пользователя
- ☹ Сложно сохранять способ описания в виде черного ящика
- ☹ Текстовые сценарий сложно проверяются пользователями

Макеты интерфейсов:

- ☺ Наглядное представление, понятное им проверяемое пользователями
- ☹ Для реалистичного проектирования интерфейсов нужны структуры данных

В обоих вариантах для описания работы пользователей сильно не хватает структуры данных

Два прохода: данные и интерфейсы



KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП

Agile-методы и современные проекты

Персоналки – новые вызовы IT

Кризис разработки: компьютеры стали доступны среднему бизнесу

- Потребовалось кратно больше софта, а выпуск разработчиков сохранился
- Процессы нормированы слабее, чем в корпорациях – переход к запутанной области по Cynefin-framework от сложной, нельзя построить модель
- **Задача меняется во время разработки** – бизнес развивается быстро

Дефицит квалифицированных кадров, конкуренция компаний за специалистов, а не разработчиков – за рабочие места

01

Agile-решение: короткие итерации и обратная связь для быстро изменяющихся задач, отказ от нормирования и фокус на людях

Выполнение автономной кроссфункциональной командой, которая видит ценность в достижении результата

02

Нынешнее быстрое развитие IT-технологий не дает надежды получить компетентный персонал: **технологий быстрого обучения** постоянно обновляющемуся контенту **не существует**

Scrum: три шага реорганизации

Разделение руководителя-предметника и руководителя-менеджера

- Положило конец спору, кто должен быть руководителем проекта
- Резко снизило требования к руководителю по совмещению профессиональных (предметных) и менеджерских (soft skill) компетенций

01

Нормативно закрепил эффективный способ управления – помощь самостоятельным людям в организации вместо микроменеджмента

- К такому способу призывает множество книг по управлению…

02

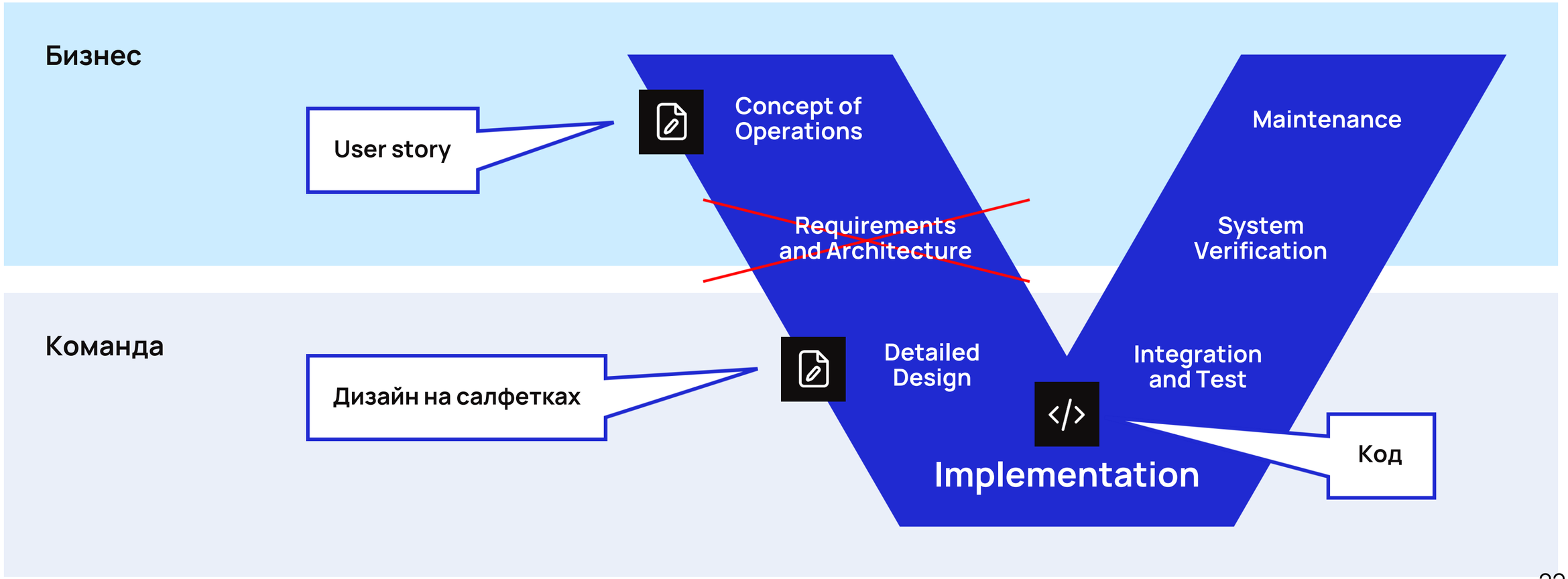
Сделал короткий цикл поставки ценности с обратной связью

- Это позволило практически реализовать принципы организации Agile

03

Scrum сочетает **самореализацию** участника команды с достижением **результативности проекта**

Light-версия требований и дизайна



Короткие задачи для итераций

User story — небольшие истории, **каждая имеет ценность**

Как ‹роль› я хочу сделать ‹действия› для того, чтобы ‹цель›

Часть ‹цель› объясняет смысл действий пользователя, позволяя выбирать из альтернативных решений при реализации

Истории компонуют в ценный функционал, приоритизируют и выбирают в спринты

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

Работа с проектом в целом

Story mapping – скоуп проекта как набор историй

Customer journey map – использование продукта разными группами пользователей

Event storming – способ быстро получить представление о предметной области **без** описания бизнес-процессов

Сценирование большого проекта

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

01

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

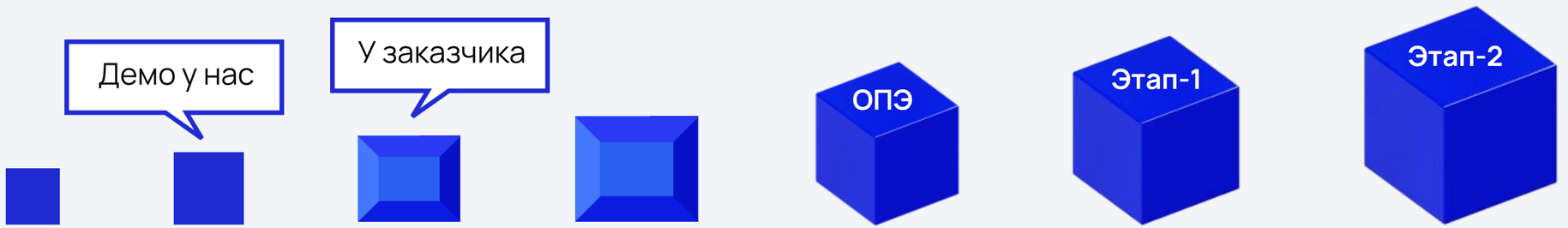
02

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

03

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

04



Новые технологии public web

Кризис технологий:

- Объектные языки не дали хорошего, устойчивого и понятного кода
- Производительность одного сервера недостаточно, распределенную систему не получается сделать базовым софтом, а надо делать в конкретном продукте
- Реляционные базы данных не обеспечивают производительность

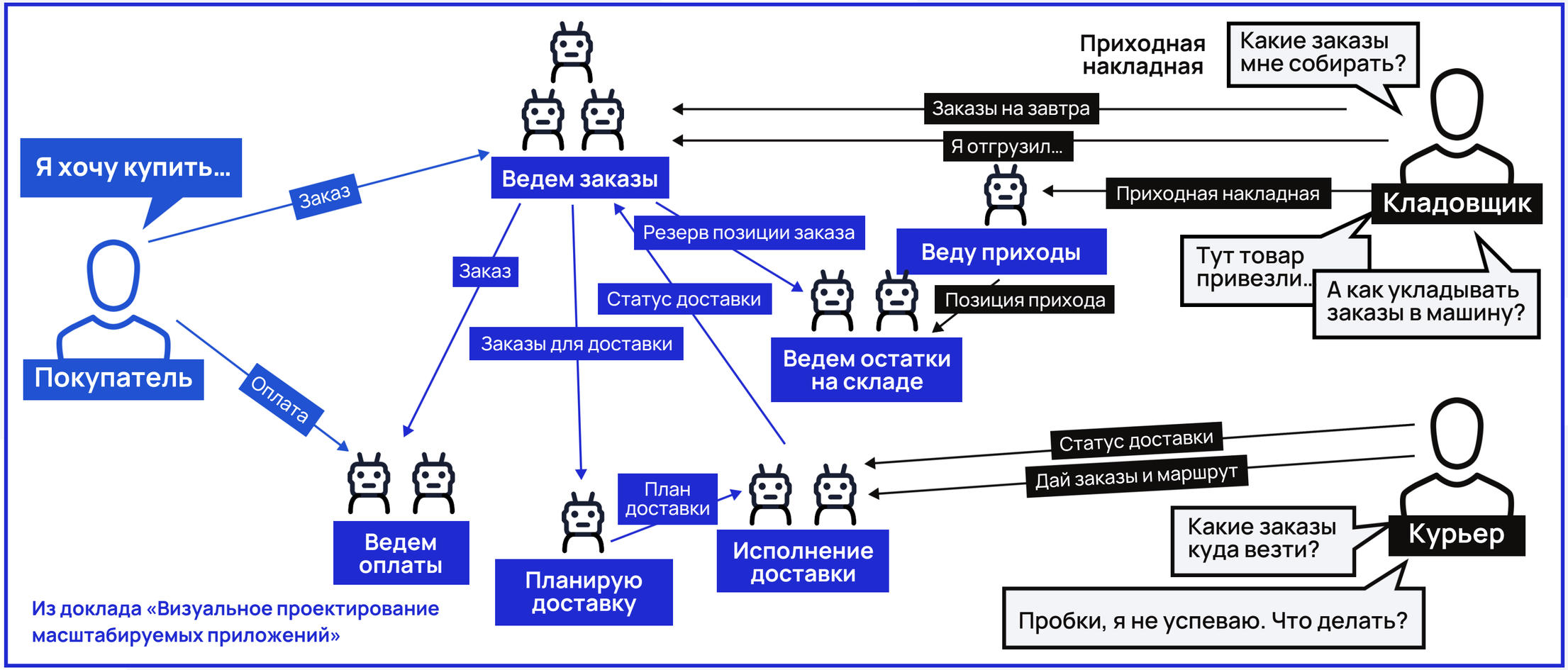
Технологические ответы:

- Переход на микросервисы и акторную модель с оркестрацией и хореографией многих асинхронно действующих агентов
- NoSQL базы данных, множество баз данных для одного приложения без транзакций и консистентности вместо единой реляционной СУБД с SQL
- Мультипарадигмальные языки – объектная, реляционная и функциональная парадигма вместе (с C# в 2008)
- Конвейер непрерывной поставки продукта (devops)

Эти изменения – в практике, объединяющая их теория отсутствует

Новые технологии с оригинальными концептами появляются интенсивно (несколько раз в год) и осваиваются без учебников

Модель для сервисной архитектуры



Проектирование сервисов

KAZAN DIGITAL WEEK при поддержке БАРС ГРУП

Нужны ли нам сервисы и зачем?

- Есть ли задача масштабирования через увеличение числа экземпляров?
- Есть ли задача заменить доработки на переписывание?

01

Ключевые вопросы проектирования

- Декомпозиция с уменьшением числа взаимных связей между сервисами
- Механизмы масштабирования без блокировки ресурсов
- Выбор способа (асинхронного) взаимодействия

02

Проектирование требует компетенций разработчика, а не аналитика

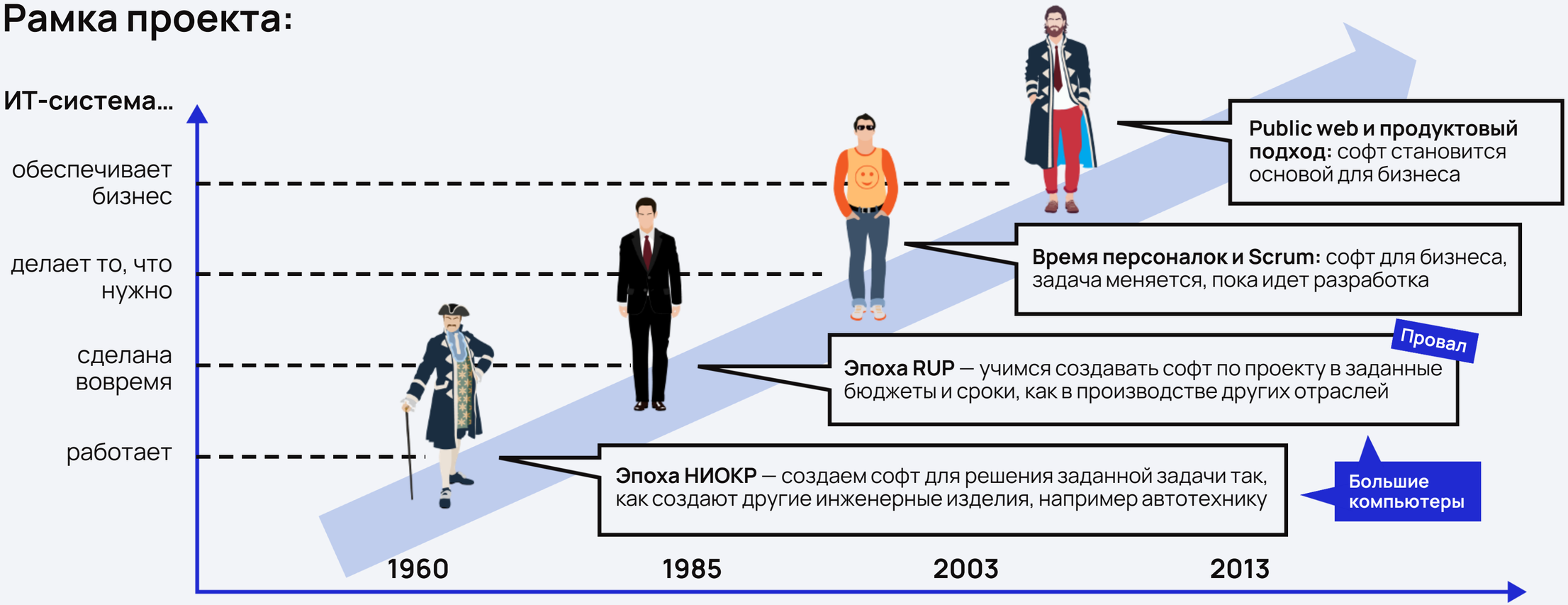
03

После выделения сервисов системный аналитик прорабатывать детальное API, но только если понимает связи по данным

04

Способы организации ИТ-проектов

Рамка проекта:



Пример: шаблоны документов

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

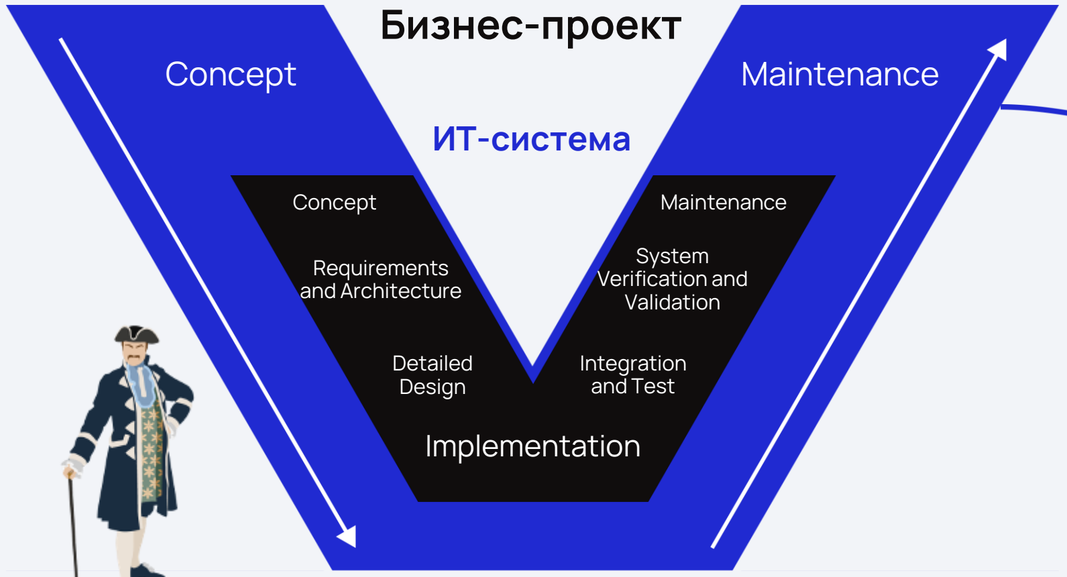
Если постановка неверна и нужна другая система?

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

ИТ- и бизнес-проект объединяются



Есть **только** ИТ-проект

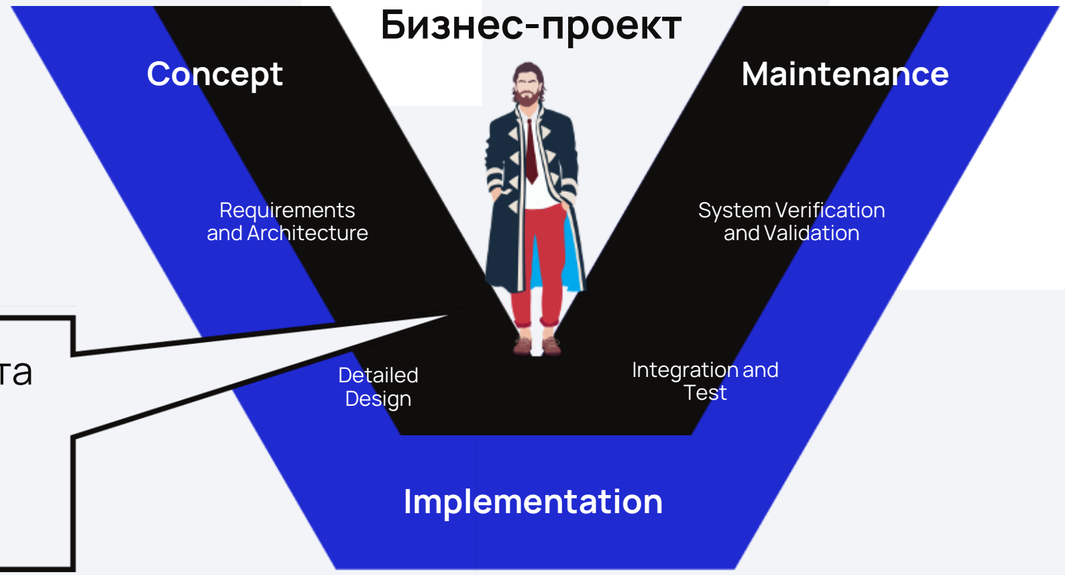
Партнерство

Бизнес должен дать четкие требования к ИТ

Работа по заказу

Сервис

Меняем ИТ в темпе изменений бизнеса



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!**

KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП

Выбор методов

Три категории постановок

KAZAN DIGITAL WEEK при поддержке БАРС ГРУП

01

Разработка новой системы поверх фрагментарной и малой автоматизации:

- выясняем модель бизнес-процессов
- создаем модель системы
- работаем над ее встройкой в процессы, изменяя их

02

Доработка существующей системы: проектируем изменения модели существующей системы и ее встройки в процессы

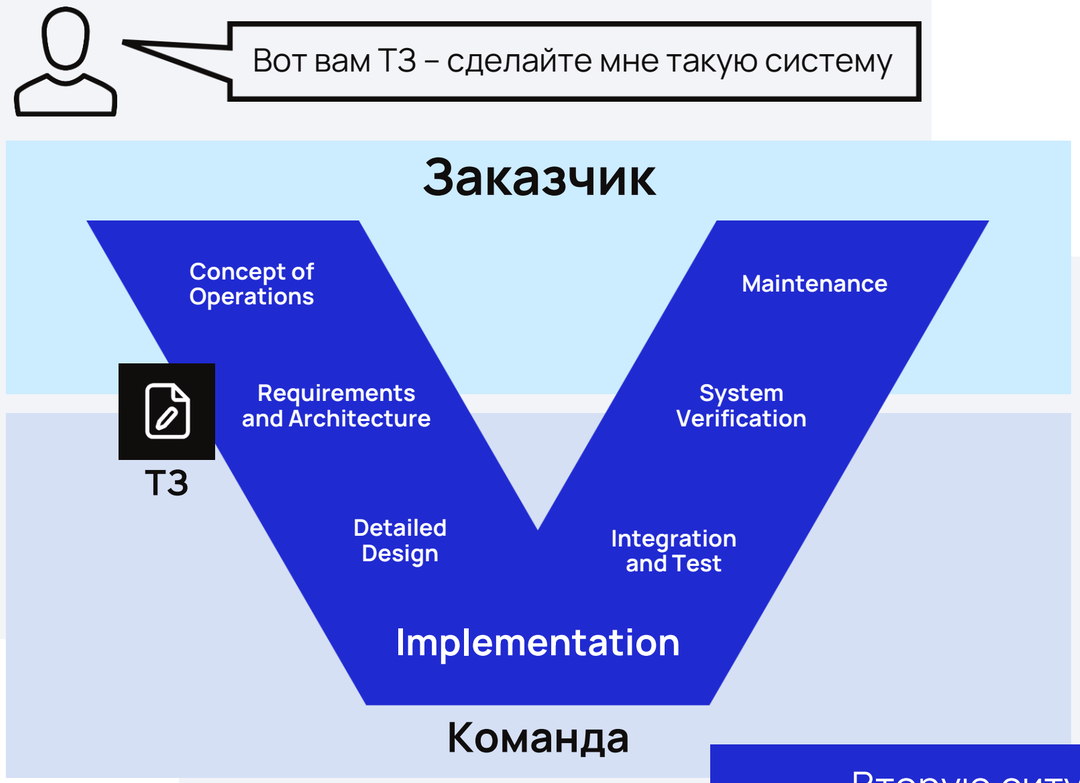
03

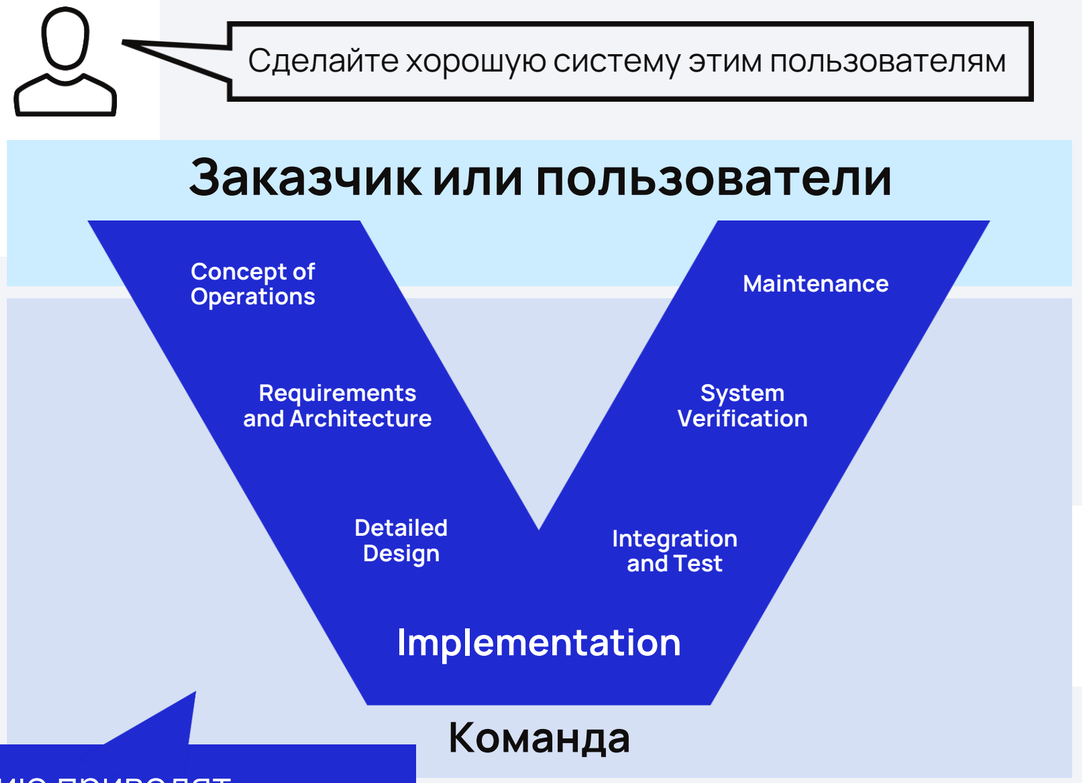
Разработка новой системы, заменяющей существующую:

- 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**

Заказчик и команда – где граница





Вторую ситуацию приводят к первой, выделяя проектирование. Нужно ли это?

Итеративность проектирования

Каким бы тщательным не было проектирование, готовый продукт будет не соответствовать потребностям пользователей – это одна из ключевых причин, породивших Agile-методы

Как действуем:

- 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

Инкрементальное проектирование возможно в любом подходе

Знания о предметной области

KAZAN DIGITAL WEEK при поддержке БАРС ГРУП

01

Нужно ли описание бизнес-процессов

- Обязательно, если при внедрении предполагается их изменение, AsIs и ToBe
- Если автоматизируются имеющиеся процессы, можно без их описания, ограничиться ролями и функциями пользователей
- Функции пользователей можно описывать через user story или use case, или просто списком

02

Альтернативные способы:

- Event Storming
- Workflow документов с ролями пользователей

03

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

Как фиксируется контракт?

KAZAN DIGITAL WEEK при поддержке БАРС ГРУП

Классика: требования в виде объектов и функций и ПМИ

- Мина: при внедрении окажется, что софт не подходит пользователям

Требования в виде макеты интерфейсов и сценариев работы в виде user story или use case

- Макеты интерфейсов согласовывать легче, чем требования
- Мина: окажется, что для функций в цепочке не будет хватать входных данных

DDD: согласованная модель приложения

- Мина: проектирование – существенная часть проекта, сложно оценить заранее
- Мина: заказчик не всегда может разбираться в модели
- Модель – очень хорошая основа для развития системы

Уровень бизнеса: обеспечение возможностей и решение проблем, при ограничениях на время, стоимость и сроки

- Мина: что все-таки будет решено – неясно, итерации этот риск уменьшают

Проектирование интерфейса

01

Naked objects: витрины объектов и вызов методов над ними

02

Макеты интерфейсов, демонстрируемые и согласуемые с заказчиком

03

Проектирование на основе use case или user story, заказчику показываем уже готовые решения

Кто это делает: аналитик, дизайнер или разработчик?

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

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

Выбирайте метод из ситуации

KAZAN DIGITAL WEEK при поддержке БАРС ГРУП

Представляйте весь спектр разнообразия методов проектирования

01

Каждая ИТ-разработка идет в своих условиях и потому способ работы с постановками сам по себе – объект конструирования

02

Выбор определяет не только опыт команды, но и характер проекта: контракт с заказчиком, организация процесса, разделение ролей

03

Помните: соответствие требованиям не гарантия успех внедрения

04

Если систему надо не только создать, но и долго развивать – лучше использовать DDD

05

Максим Цепков

http://mtsepkov.org

@MaximTsepkov

На сайте много материалов по Agile, бирюзовым организациям и спиральной динамике, мои доклады, статьи и конспекты книг

Up!Date

Образовательная
ИТ-конференция

Спасибо за внимание

KAZAN DIGITAL WEEK

при поддержке БАРС ГРУП