TeamLead Conf FOUNDATION 2022

CUSTIS® ИТ-РЕШЕНИЯ ДЛЯ РАЗВИТИЯ

Почему проектный подход не работает в ИТ

Максим Цепков

Главный архитектор решений CUSTIS
Навигатор в мире Agile, бирюзовых организаций и Спиральной динамики

Что такое проектный подход?

CUSTIS

Проектный подход — это классический способ делать проекты, описанный в PMBOK

Проекты можно делать и иными способами:

- С помощью Scrum, Kanban или других Agile-методов
- С помощью гибридных технологий, собираемых из различных практик

У способа из PMBOK есть репутация надежного, но тяжелого метода

- Это влечет сожаления о том, что обстоятельства мешают нам его реализовать
- А иногда приводит к попыткам реализации

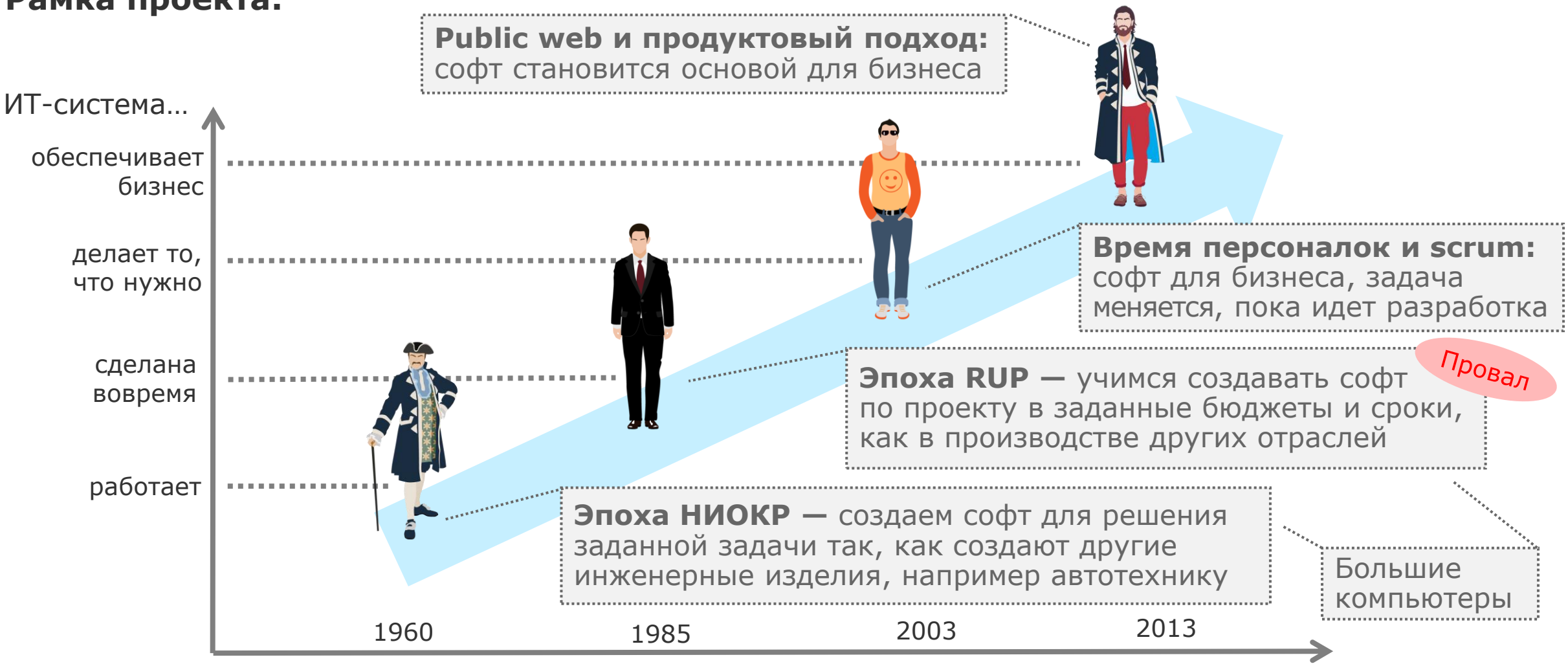
Однако эта репутация не заслужена. В проектный подход заложены ограничения, которые делают его неприменимым к ИТ-проектам, особенно современным

Я разберу ограничения проектного подхода и полезные практики, которые стоит из него взять, включая современные light-версии

История культур ИТ-проектов

CUSTIS

Рамка проекта:



Эпоха НИОКР: когда компьютеры были большими

CUSTIS

Прикладные задачи:

- физические расчеты, управление ракетами и т. п.
- автоматизация для корпораций со **стабильными** процессами

Учимся создавать и развивать большие и сложные системы

Системный софт: операционные системы, языки, компиляторы, библиотеки программ, чтобы упростить разработку и выполнение софта

Использование математических моделей как основы для теории

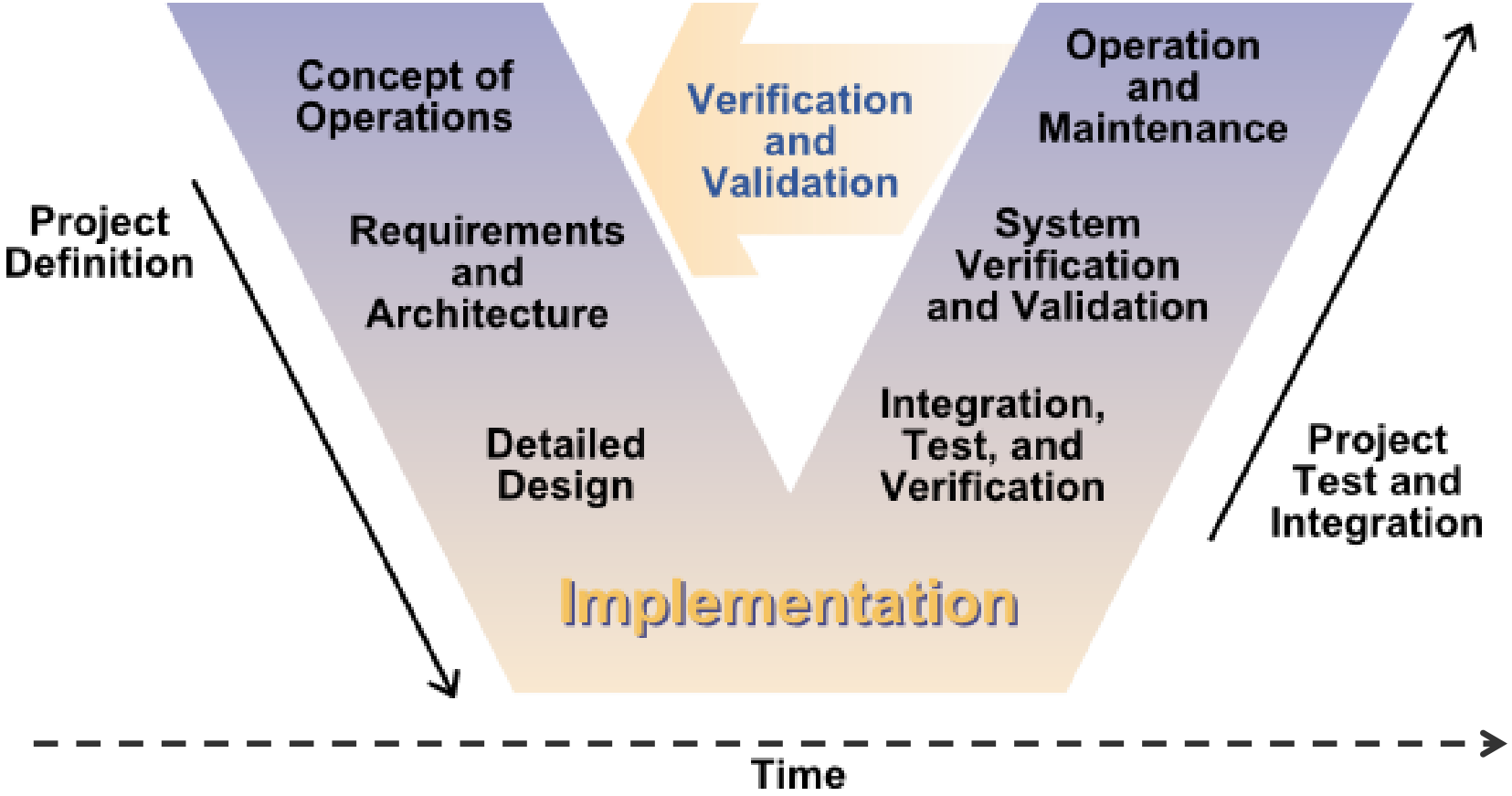
Алгоритмика: Кнут «Искусство программирования» (1968–1973)

Структурное программирование, чтобы программа была понятной

SQL и **ER-диаграммы** — проектирование и работа с большими данными

Вирт «Алгоритмы + Структуры данных = программы» (1976)

Проект представляет V-модель



V-Model (Wikipedia)

Эпоха RUP

Эпоха RUP: укладываемся в бюджет и сроки

CUSTIS

PMBOK-3 (2004)
RUP (2003)

Потребность в софте возрастает, разработка — часть бизнес-проекта

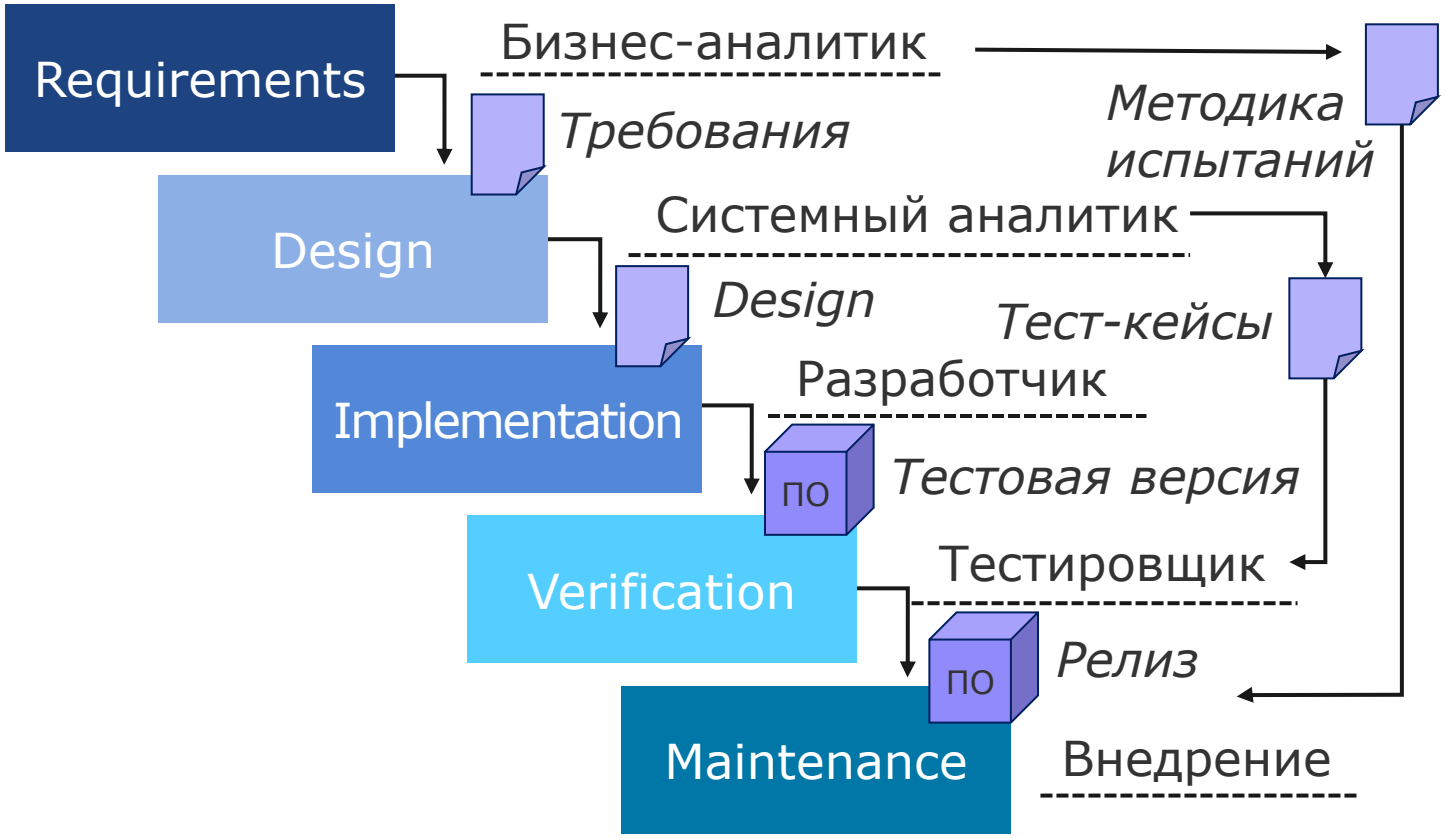
Применим к ИТ-разработке принципы **промышленного производства**

- Разделим задачу на этапы: проектирование, разработка, внедрение
- Наладим процессы, разделим роли и зоны ответственности
- Определим требования к компетентности сотрудников
- Технические средства: базовый софт (СУБД, фреймворки) и **ООП**

Оценка качества: удалось ли выполнить проект в срок, уложиться в бюджет и достичь ожидаемых результатов

Не получилось.
Успех около 30%

Водопад — специализация по фазам проекта



Каждой фазе — своя роль

Роли выполняются разными людьми или командами

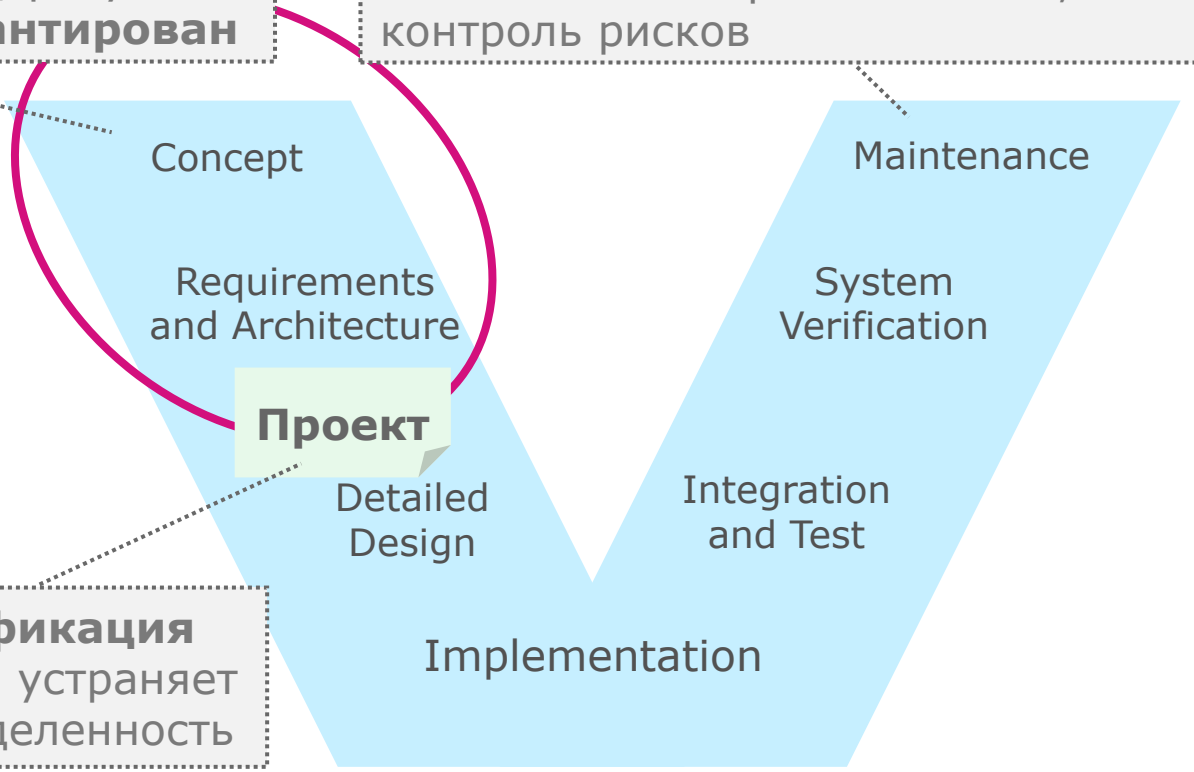
Передача работы — формализована **через артефакты**

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

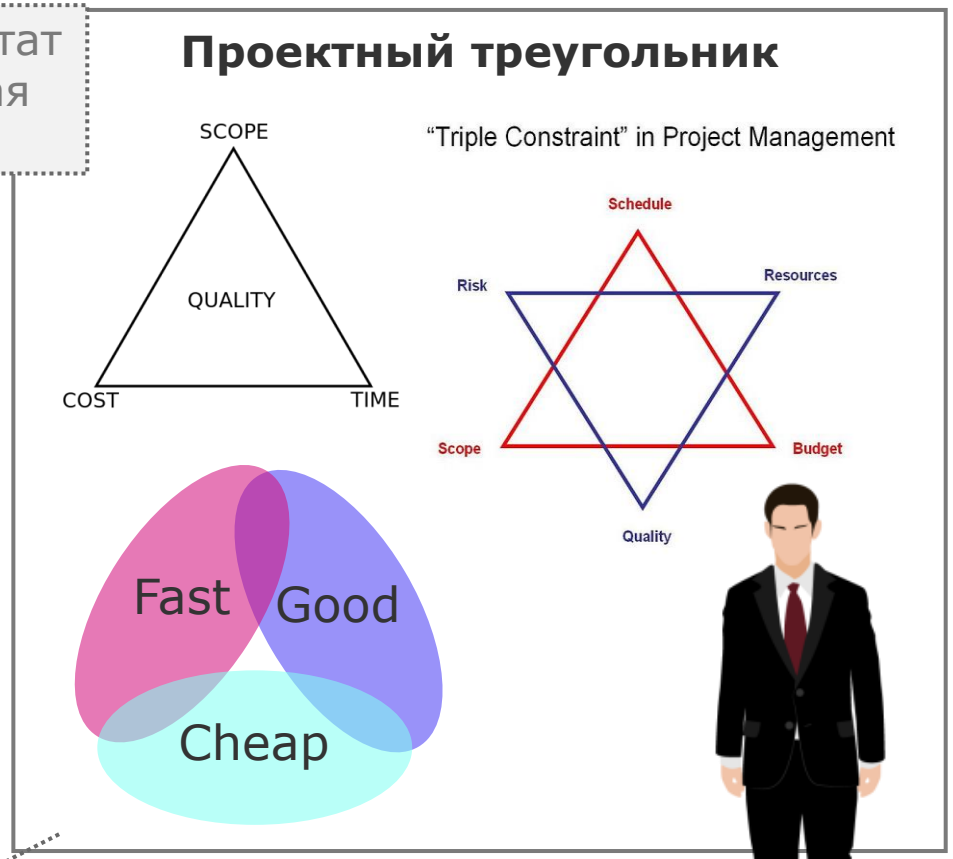
Классическая V-модель переносит неопределенность в проектирование!

Проектирование — **НИОКР**, результат **не гарантирован**

Исполнение — **производство**, результат обеспечивается регламентами, включая контроль рисков



Спецификация проекта устраняет неопределенность



PM **контролирует** результат проектирования, а затем управляет исполнением в проектном треугольнике

CUSTIS

Персоналки и public web

Кризис появления персоналок

CUSTIS

Компьютеры стали доступны среднему и мелкому бизнесу:

- потребовалось кратно больше софта, а выпуск разработчиков сохранился
- процессы в этом бизнесе слабее нормированы, чем в крупном, предметная область является **запутанной, а не сложной** (Cynefin), нельзя построить модель
- бизнес меняет задачу во время разработки – меняется рынок и способ организации
- нет научных методов разработки софта – разработчики экспериментируют с решениями

Решением стали Agile-методы ведения ИТ-проектов:

- признать, что разработка — это **эксперимент** и возможны провалы
- ограничить размер эксперимента-итерации, двигаться малыми шагами
- доводить эксперимент до проверки на конечном потребителе
- проблемы коммуникации решить, снизив (или устранив) специализации

Время public web — технологии

CUSTIS

Кризис технологий:

- объектные языки не дали хорошего, устойчивого и понятного кода
- производительности 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
- базы данных NoSQL, множество БД 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, CI/CD)

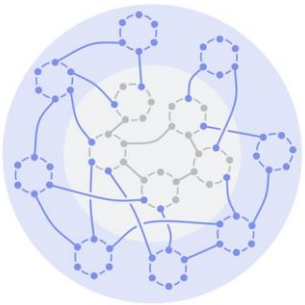
В новых технологиях надо принципиально **иначе описывать устройство** системы, если не ограничиваться описанием экранов или внешнего поведения системы через use case или user story

Время public web — продуктовое мышление

CUSTIS

Софт не поддерживает создание бизнес-продукта, а образует этот продукт, другие активности становятся поддержкой, как в OZON, Uber и др.

При успехе MVP, создаваемого одной командой, число разработчиков увеличивается, получается много полуавтономных команд в сетевой структуре с оркестрацией между ними по графику и продуктам работ



Конструкция **фрактальна**:

- каждая команда порождает гипотезы о перспективных фичах, проверяет их на пользователях, развивает
- успех гипотез приводит к росту команды и ее разделению по направлениям, а не по специализациям
- на уровне компании также порождаются направления и создаются команды под них

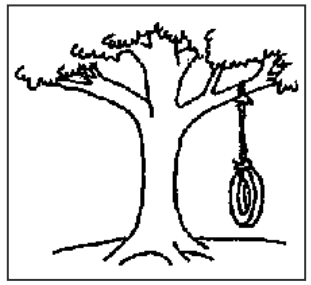
Рамка работы — развитие социотехнической системы, образуемой продуктом (схема OMG Essence)

Проблемы проектного подхода

Каждая передача искажает смысл

CUSTIS

Что хотел заказчик



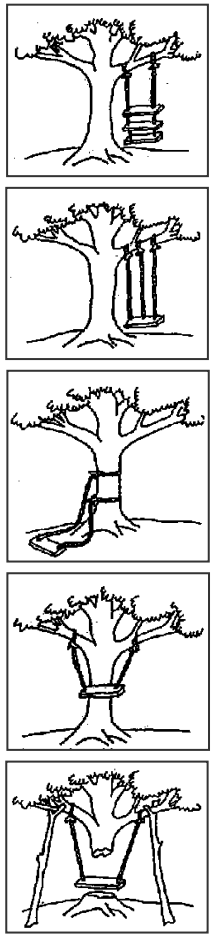
Requirements → Бизнес-аналитик

Design → Системный аналитик

Implementation → Разработчик

Verification → Тестировщик

Maintenance → Внедрение



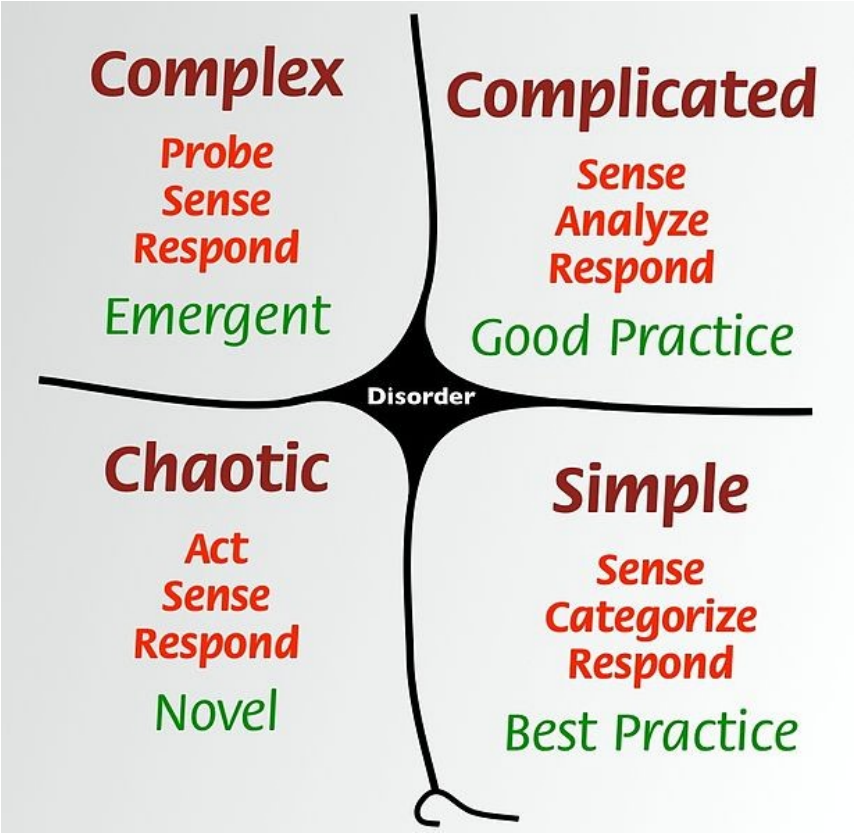
Старый известный образ

Предсказуемость результата — Cynefin framework

CUSTIS

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)

Разработка кода — часть проектирования

CUSTIS

Обычные НИОКР

ИТ-разработка

Jack W. Reeves. What is software design, 1992 (перевод)

Ловушки новых технологий

CUSTIS

Новые технологии имеют большой потенциал, но проработаны недостаточно

Стейкхолдеры знают этот потенциал и часто думают, что их использование — как путешествие по хорошей дороге, а по факту это форсирование болота

Пара примеров разного уровня:

- чат-боты вместо операторов в службах поддержки, вы все с ними встречались
- делаем мобильное место продавца вместо десктоп-приложения — оказывается, что для печати на ближайшем принтере нет готовой технологии, и в браузере тоже нет. Получается исследовательская задача

Об оценке задач

CUSTIS

Проблема статистики: оценка задач имеет распределение с длинным хвостом, поэтому для уверенного попадания необходим резерв в 2–3 раза к математическому ожиданию. Подробности — Андрей Бибичев «Пуассоново горение сроков»

Проблема технологий: при использовании незрелых технологий реализация частной задачи может потребовать доработки этой технологии, которую не предполагали

Проблема квалификации: когда в команде существенная доля людей осваивают технологии и учатся, невозможно уверенно оценить их скорость решения задач

В чем проблема человеческого фактора?

CUSTIS

У разных сотрудников производительность отличается на порядок, однако это сильно зависит от типов задач

Зная производительность на одних типах задач, нельзя предсказать производительность на других — невозможно построить грейды

Опыт сотрудника не является гарантией быстрого освоения новых технологий и фреймворков, решения новых типов задач

Способность быстро осваивать новое сложно измерить

Нет технологий обучения, как быстро осваивать новое

- Design Thinking не оправдал надежд, хотя продвижение было
- Школа А. Левенчука (курсы онтологики, системного мышления и др.) работает над решением этой задачи

Почему RUP не достиг успеха

CUSTIS

Разработка кода — проектирование низкого уровня, а не производство

Технологии только развиваются, незрелые — TRL 4-5. При использовании доводим технологию внутри разработки: нельзя предсказать трудозатраты и сроки, для предсказания нужны TRL не ниже 8 (опыт Boeing)

ООП не дал методов разработки, которые бы позволили технологично и понятно писать код (хотя улучшил ситуацию)

Закон Конвея: архитектура софта воспроизводит структуру коммуникаций в команде разработки — организация неотделима, спроектированная конструкция по факту не воплощается

В разработке критичен человеческий фактор, это поняли в 1980-х: Фредерик Брукс («Мифический человеко-месяц», 1975) и Том ДеМарко («Человеческий фактор», 1987). Преодолеть не получилось

CUSTIS

Делаем проекты — решения Agile-методов

Решения от Agile

CUSTIS

Планирование релизов и вех проекта: поставка ценного функционала в ходе проекта, а не в конце, и запуск его в эксплуатацию

Оценка: отказ от детального планирования, конкретизация задач по мере движения в проекте и переход от грубых начальных оценок к уточненным перед стартом выполнения

Наблюдение за ходом проекта и **работа с бюджетом:** progress bar по бэклогу и оценка перспектив, резерв в бюджете и оценка его исчерпания

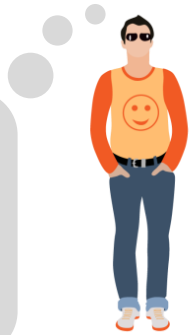
Неопределенность требований: организация обратной связи от стейкхолдеров и пользователей через регулярные демонстрации или другим образом

Некомпетентная команда: ежедневная фиксация проблем и коллективный поиск решений

Работа с рисками: регулярная обратная связь, малый квант потерь, вынос проблемных задач в начало работы

Если требования неверны 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

Сценирование заказного проекта

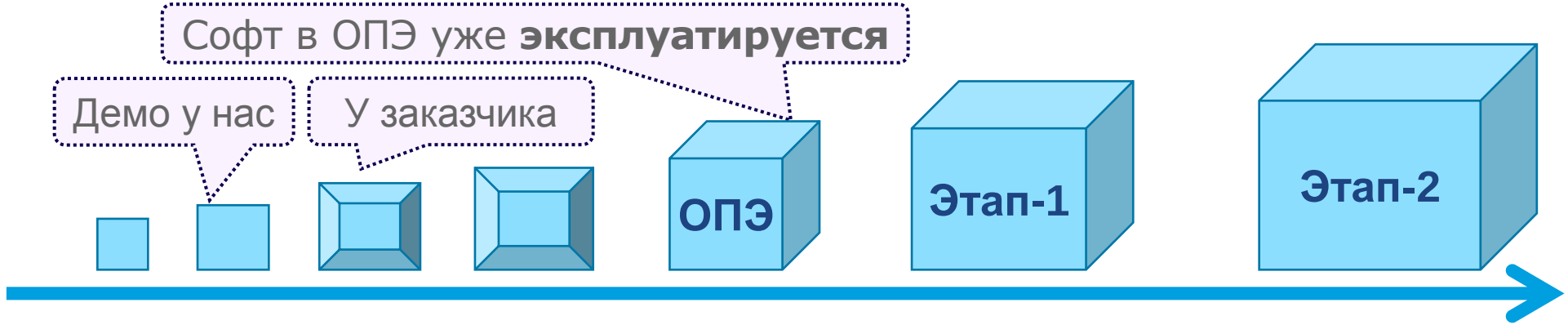
CUSTIS

Выделяем из скоупа проекта MVP, пригодный для запуска в эксплуатацию

Функционал ОПЭ разбиваем его **на 2–4 демо**, представляя **интересный** бизнес-заказчику **целостный** функционал (тоже работает принцип MVP)

Первые демо проводим на своей территории — **знакомим заказчика с командой**

Далее разворачиваем тестовую среду у заказчика, переносим туда демо-испытания



Из моего доклада «Планирование проекта от демонстраций для разворачивания партнерства с заказчиком» на Saint Teamlead 2019

Проектный подход и Agile-методы

CUSTIS

ХАРАКТЕРИСТИКА	ПРОЕКТНЫЙ МЕНЕДЖМЕНТ	AGILE-ПОДХОДЫ
Устройство мира по Cynefin-фреймворку	Сложная (complexity) область: можно построить модель с учетом динамики изменений, она ляжет в основу плана	Запутанная (complex) область: построить модель нельзя (или долго и дорого), изменения непредсказуемы
Способ действия	Разработка проекта полного создания продукта, чтобы учесть все риски, затем его воплощение. Убираем неопределенность в проектировании	Придумываем MVP как гипотезу о минимальном ценном результате, делаем, оцениваем и далее развиваем через эксперименты
Технологии	Только апробированные	Можно использовать новые, незрелые
Команда	Обученные, компетентные специалисты, каждый отвечает за свою часть	Вместе работают на цели проекта, умеют обучаться новому в ходе проекта
Успех проекта	Выполнить планы в срок и бюджет, возврат инвестиций — вне проекта	Решить проблему, создать ценность, результат должен превысить затраты
При неудаче	Если план утвержден и выполнен — виноват заказчик, иначе — руководитель	Ошибки возможны, надо совместно искать выход и избегать повторения

Различия — в мышлении, Agile создавали в другой картине мира

Подробнее — моя статья «Agile-методы и проектный подход — в чем разница?»

Понимает ли разработчик смысл приложения?

Не понимает

Понимает

Создает

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

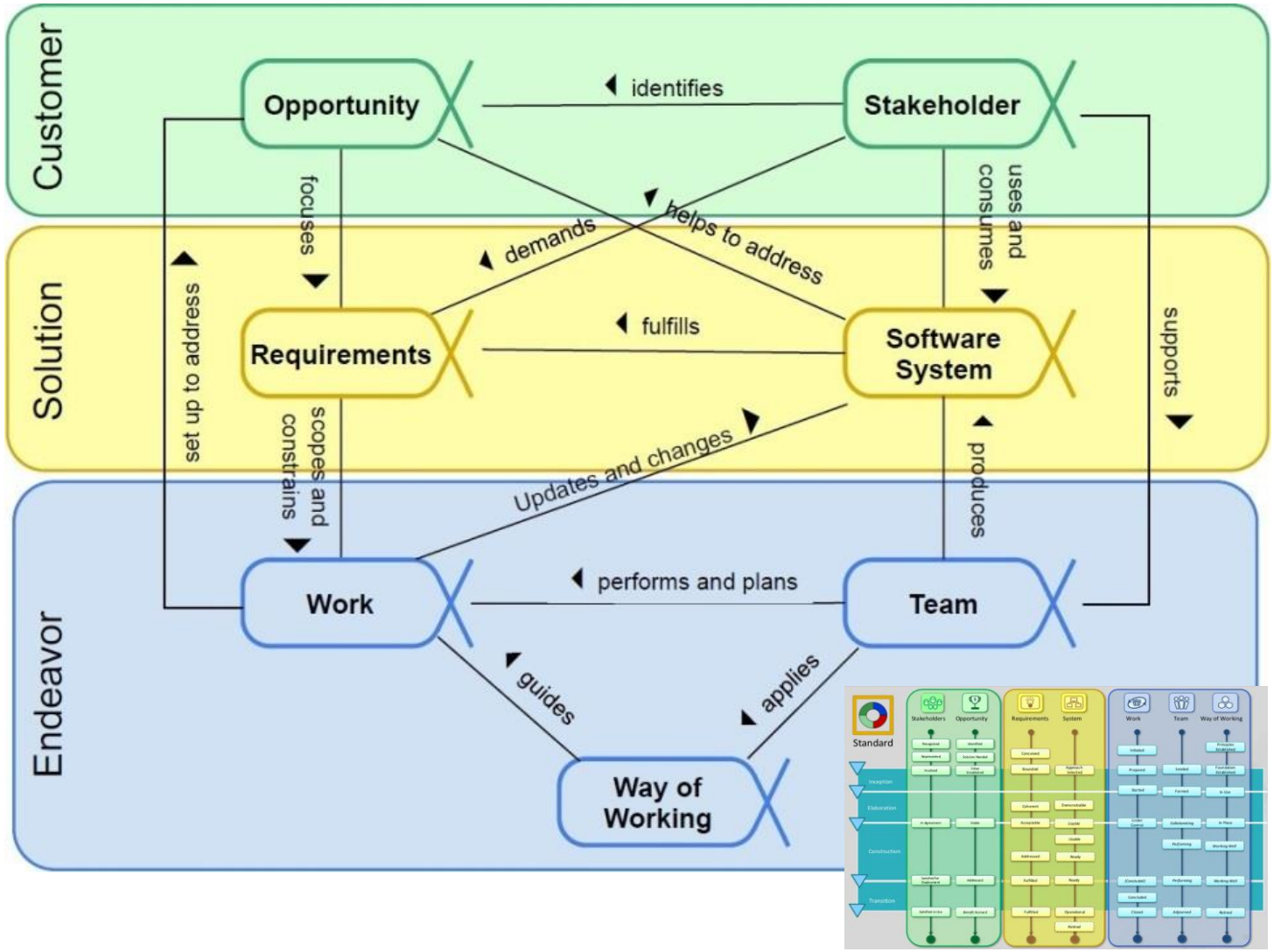
Разработчик понимает цели софта и предлагает технические решения и идеи

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

Разработчик в команде проекта активно участвует в формировании целей софта

OMG Essence — делаем гибриды

CUSTIS



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

Заключение: как делать проекты

CUSTIS

Есть много гибридных вариантов (Стратоплан, Анатолий Левенчук, Павел Алферов и др.)

- Работаем с людьми: лидер проекта, драйв команды, соучастие стейкхолдеров
- Работаем с неопределенностью: демонстрации, обратная связь, проверка гипотез
- Планирование и оценка: поэтапная конкретизация, прогноз продвижения, статусы

Важно! Для каждого проекта нужен свой метод, учитывающий особенности, собираем его так же, как технологии — из разных библиотек и фреймворков, учитывая их совместимость

http://mtsepkov.org

@MaximTsepkov

На сайте **страница доклада со всеми ссылками**, 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