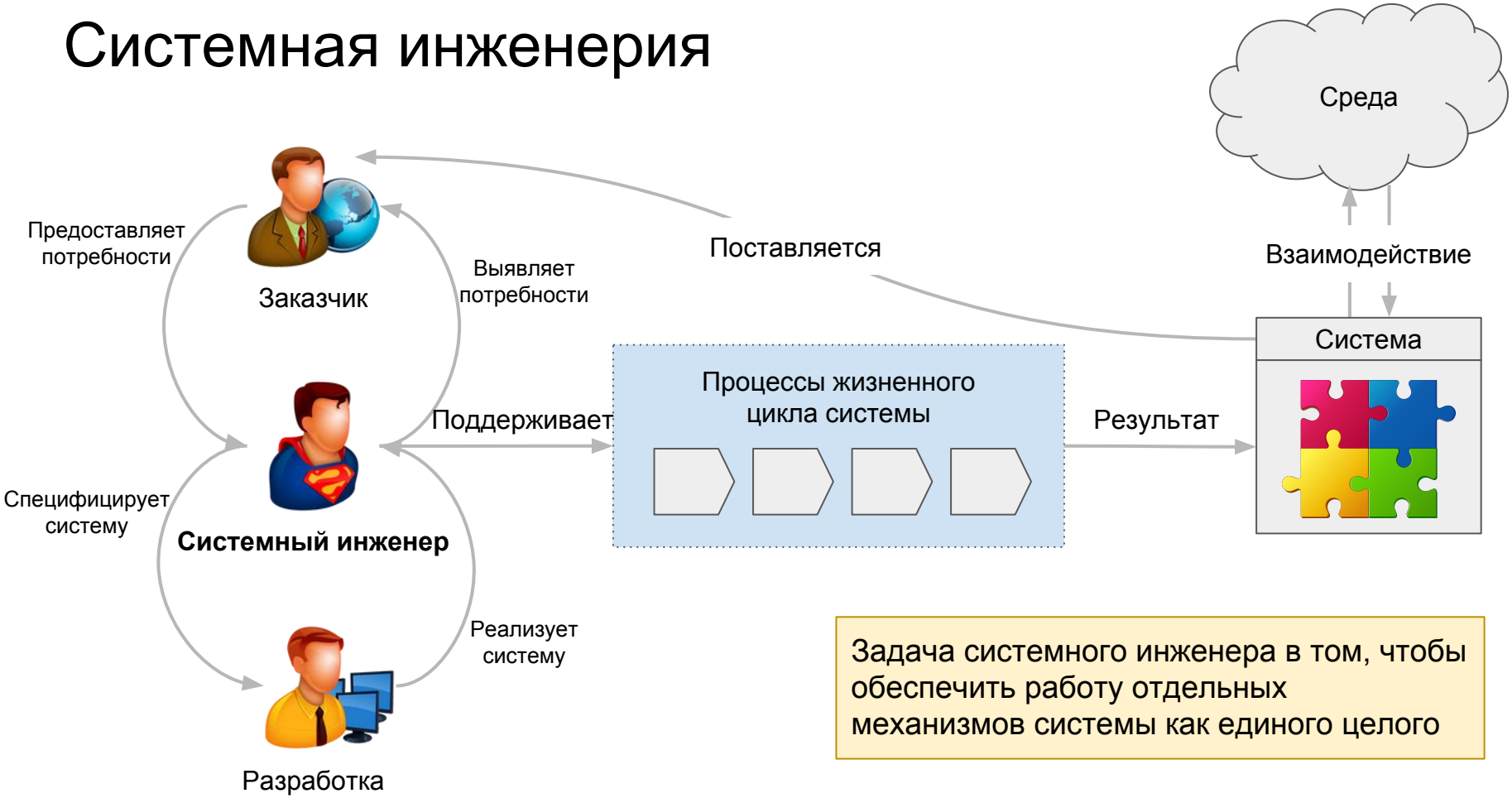
Бизнес-анализ

Тема 2

Системы

Системная инженерия

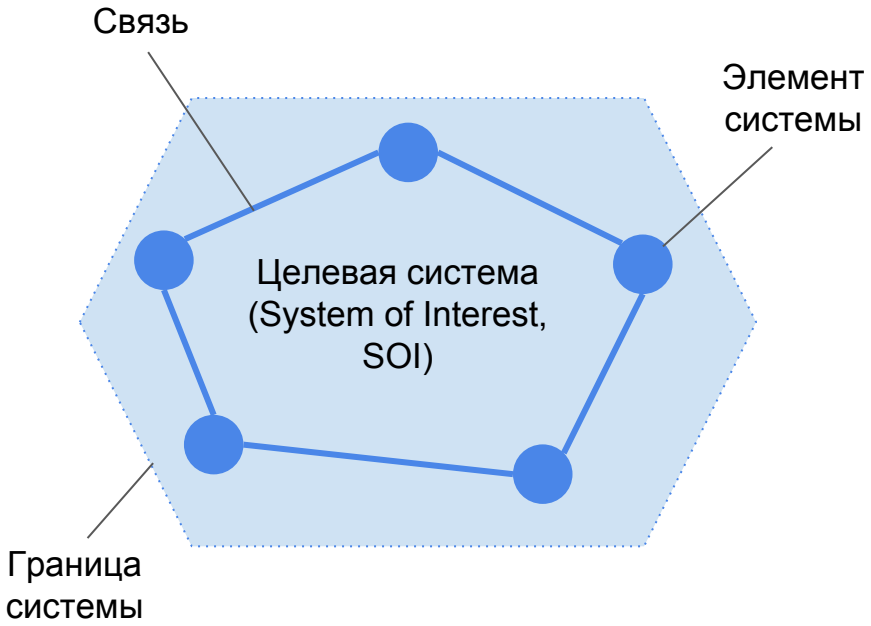


Задача системного инженера в том, чтобы обеспечить работу отдельных механизмов системы как единого целого

Системный аналитик и бизнес-аналитик
не являются системными инженерами,
но им крайне необходимы навыки
системного мышления

Система в системной инженерии

- Система -- это набор элементов, которые взаимодействуют для достижения заданной цели
- Цель системы:
  - формулируется бизнесом и стейкхолдерами
  - является отправной точкой для проектирования
  - дает возможность понять, все ли было правильно сделано
- Элементы, связи и внешняя граница являются результатом проектирования
- Система должна быть способна достигать своей цели автономно,
  т.е. ею можно управлять независимо от других систем

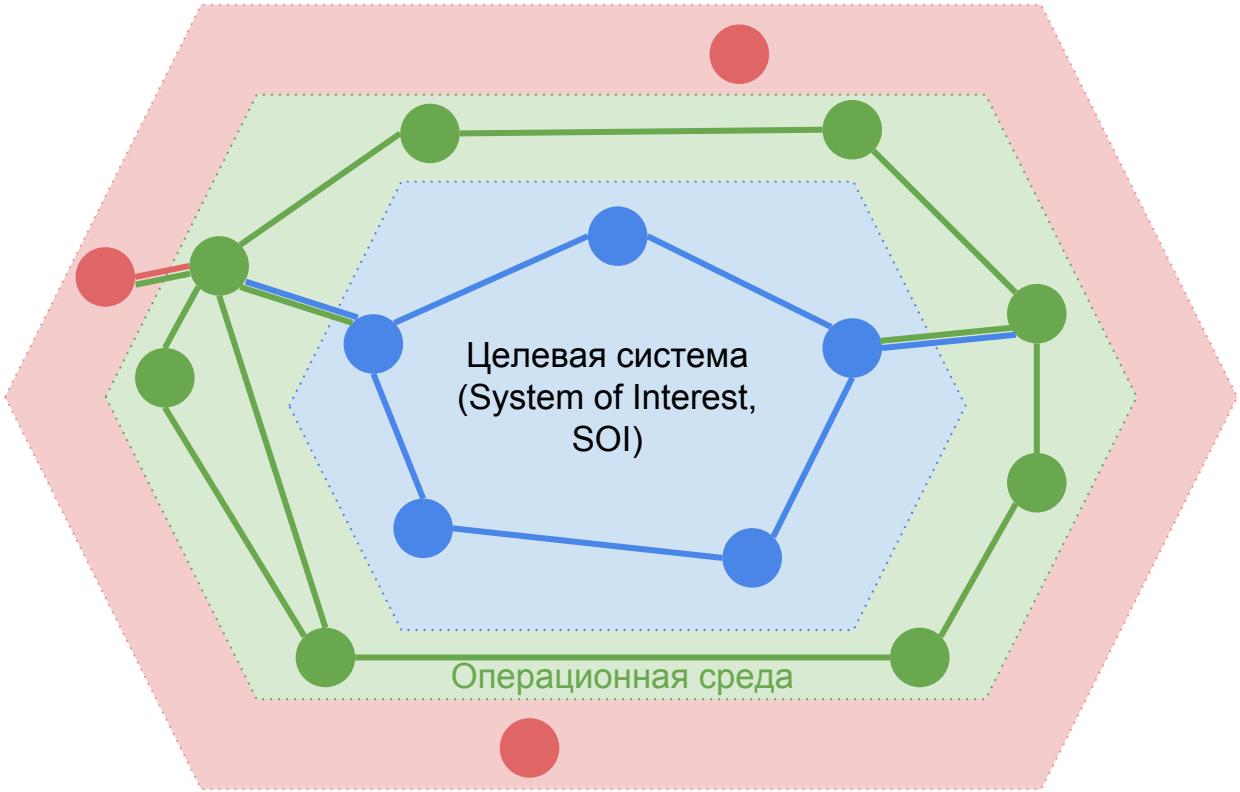


Некоторые важные классы систем

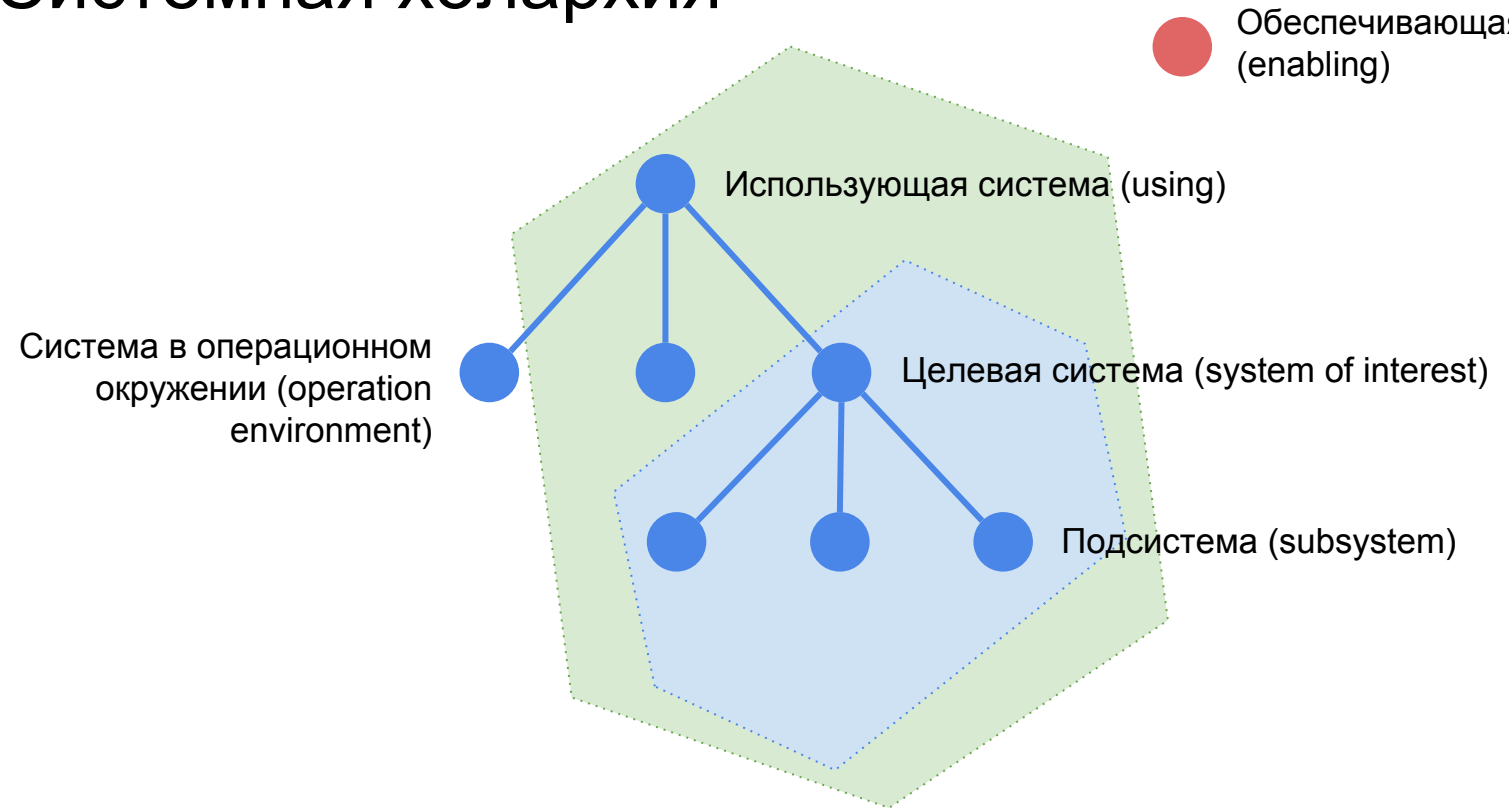
Открытые, закрытые
Естественные, **искусственные**, **модифицированные человеком**
Абстрактные, **конкретные**
Прецедентные, беспрецедентные
Статические, динамические, гомеостатические
Гомогенные, гетерогенные
Мягкие, жесткие
Централизованные, децентрализованные

Область системной инженерии

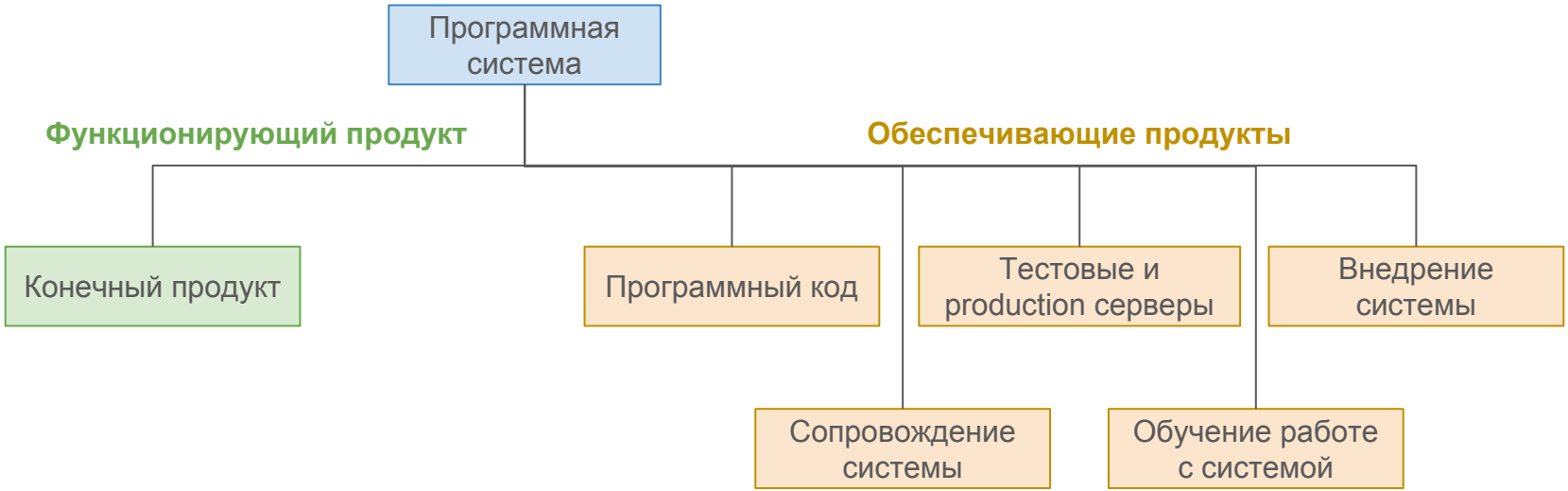
Система и ее внешняя среда



Системная холархия



Система как продукт



Система как возможность достижения цели

Самолет
(основное оборудование)

Система как возможность достижения цели

Персонал
(пилоты, диспетчеры, персонал аэропорта)

Самолет
(основное оборудование)

Данные
(расписание, технические схемы, спецификации оборудования)

Службы аэропорта
(заправка топливом, багаж, ремонт)

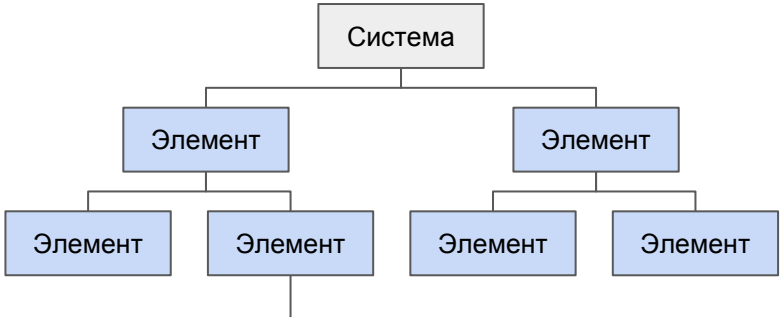
Обучение персонала
(обучение пилотов и членов экипажа, диспетчеров, работников аэропорта)

Сооружения
(терминалы, ВПП, вышки, парковка)

Правила и процедуры
(регламенты, правила провоза багажа, правила безопасности)

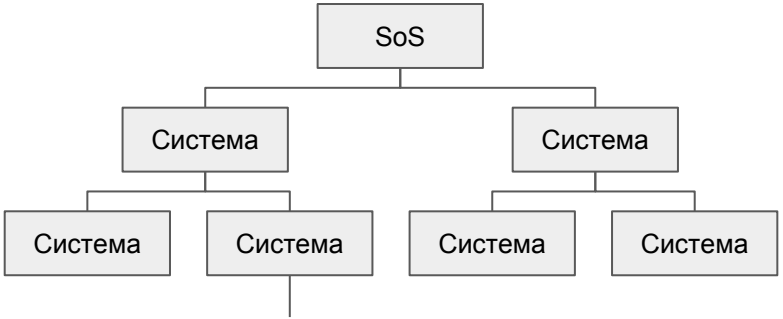
Организационная структура
(подразделения, руководство)

Система систем (System of Systems, SoS)



Оптимизирован под цели системы

Система оптимизирована

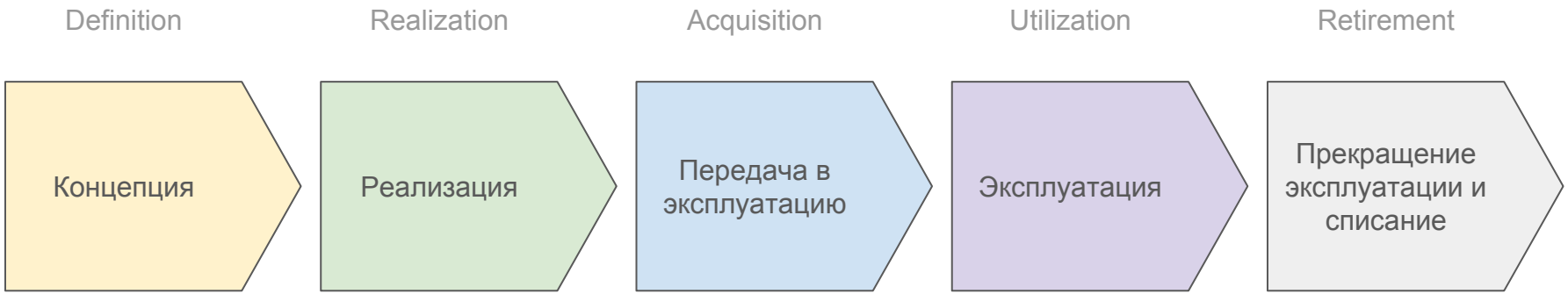


Оптимизирована под свои цели

SoS не оптимизирована

Жизненный цикл системы (продукта)

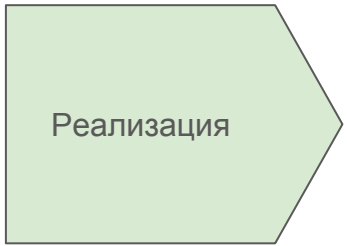
Жизненный цикл системы



Жизненный цикл системы

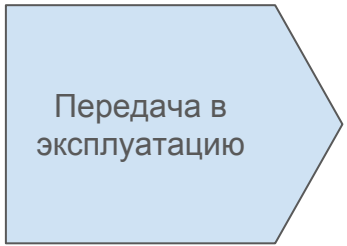
- Этап концептуального дизайна -- это переход из мира бизнеса в мир инженерии
- Этот этап включает в себя разработку требований к системе, что требует вовлечения бизнес-руководства и стейкхолдеров верхнего уровня
- Этап включает в себя последовательную разработку трех видов требований:
  - бизнес-требований
  - требований стейкхолдеров
  - спецификации системы

Жизненный цикл системы



- Реализация -- это этап когда непосредственно разрабатываются подсистемы, агрегаты и компоненты системы
- У любой системы есть разработчик (supplier) и потребитель (acquirer)
- Разработчик может быть частью принимающей организации, но довольно часто он является внешним контрактором (подрядчиком)
- Если контрактор не может осуществить разработку полностью, то он нанимает субподрядчиков

Жизненный цикл системы

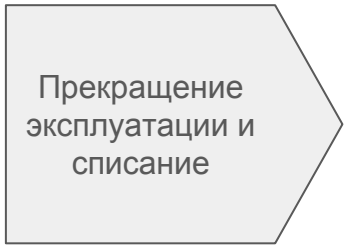


- Этап разработки завершается внедрением системы на мощностях потребителя
- При передаче системы происходит ее проверка (acceptance test) на соответствие требованиям принимающей стороны
- В современных проектах такая проверка производится в конце каждой итерации

Жизненный цикл системы

- Система продолжает подвергаться модификациям даже в фазе эксплуатации
- Основные причины:
  - Устранение ошибок
  - Соответствие изменившимся внутренним и внешним требованиям
  - Улучшение производительности
- Если внесение изменений в систему становится нецелесообразным, то система переводится в состояние поддержания жизнедеятельности (maintenance)

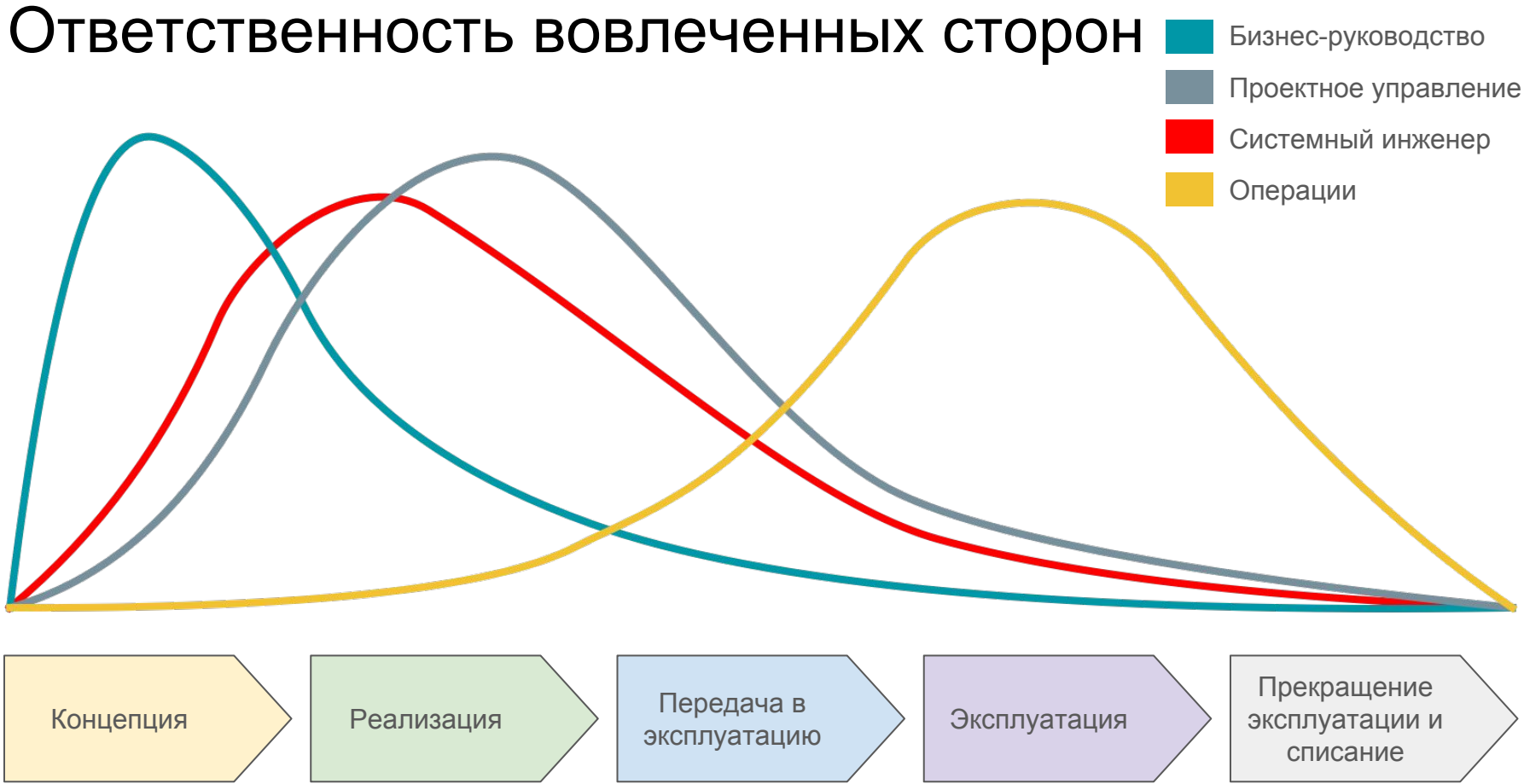
Жизненный цикл системы



- Система используется до тех пор пока *:
  - У бизнеса есть в ней потребность
  - Система предоставляет необходимые функции
  - Ее содержание оправданно по финансовым или политическим соображениям
- В противном случае система терминируется, ее жизненный цикл завершается

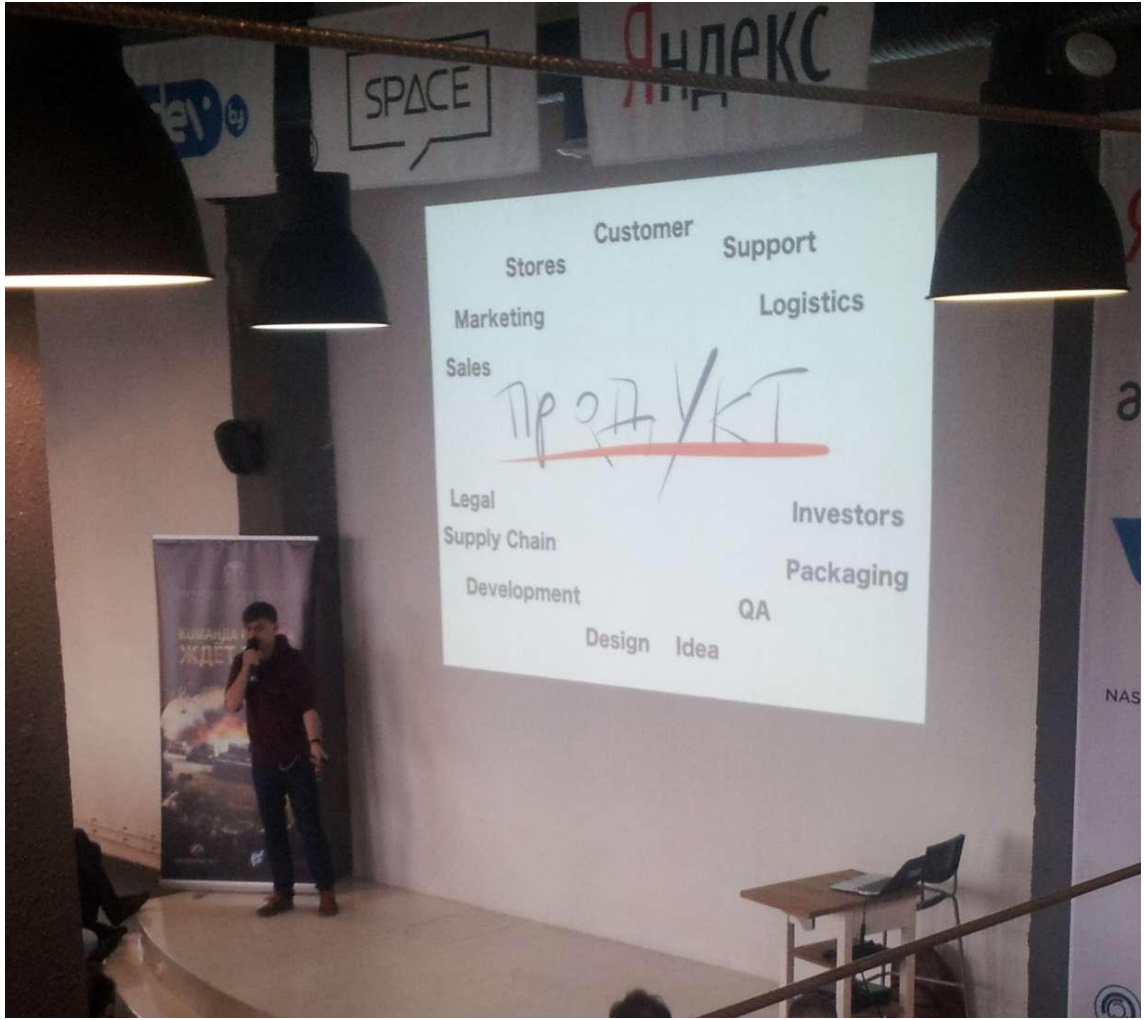
* Даже ИТ-системы могут эксплуатироваться очень длительный срок

Ответственность вовлеченных сторон

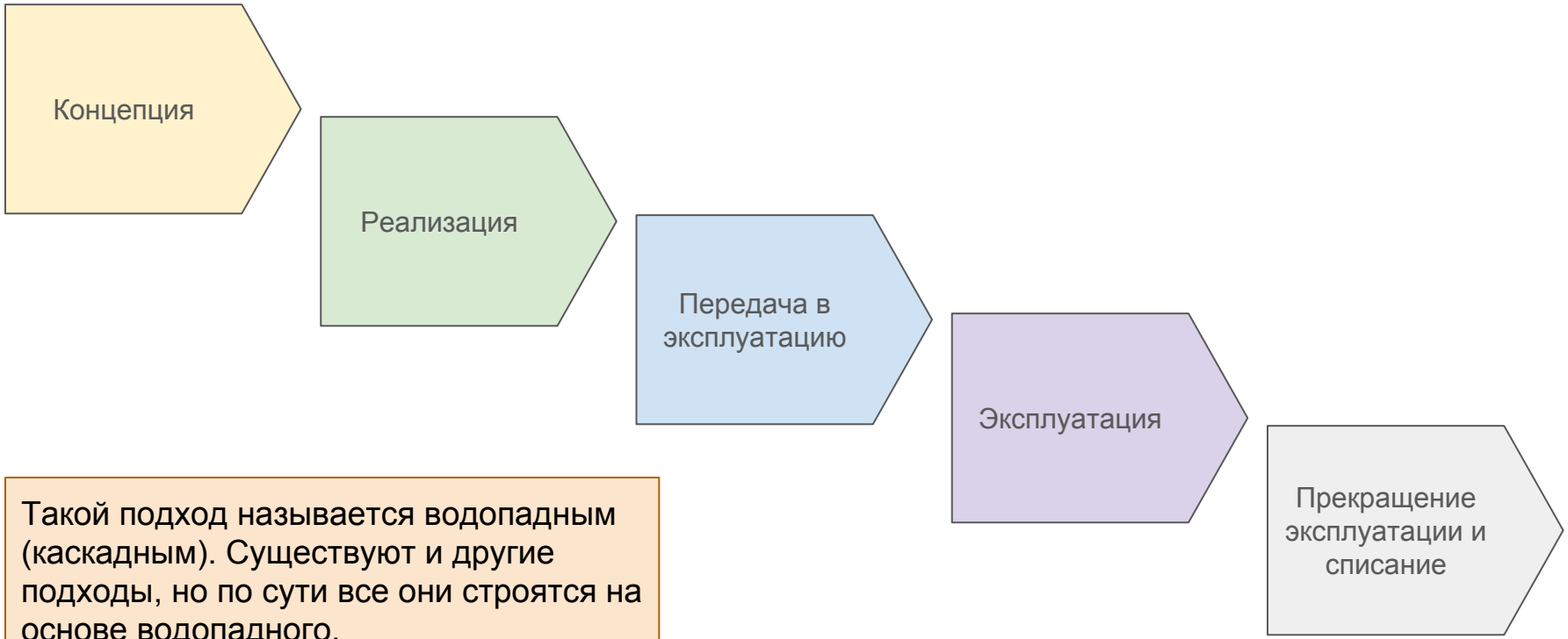


Жизненный цикл продукта

Потребность	Потребность ⇨ Запрос предложения (Request for Proposal, RFP) ⇨ Тендер ⇨ Предложение ⇨ Контракт
Проектирование	Инициирование ⇨ Планирование ⇨ Разработка требований
Разработка	Проектирование архитектурного решения ⇨ Разработка ⇨ Тестирование
Развертывание	Развертывание ⇨ Пользовательское тестирование (User Acceptance Testing, UAT)
Функционирование	Функционирование продукта
Поддержка	Поддержка ⇨ Закрытие продукта ⇨ ☠


SPACE
Яндекс
Customer
Support
Stores
Logistics
Marketing
Sales
Продукт
Legal
Investors
Supply Chain
Packaging
Development
QA
Design
Idea

Водопад (Waterfall)



Такой подход называется водопадным (каскадным). Существуют и другие подходы, но по сути все они строятся на основе водопадного.

Системная инженерия

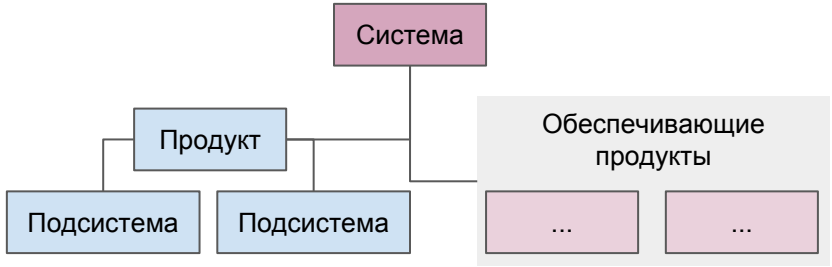
Догматы системной инженерии

Подход “сверху-вниз”
Инженерия требований
Фокус на полном ЖЦ
Оптимизация и баланс системы
Интеграция различных дисциплин
Управление

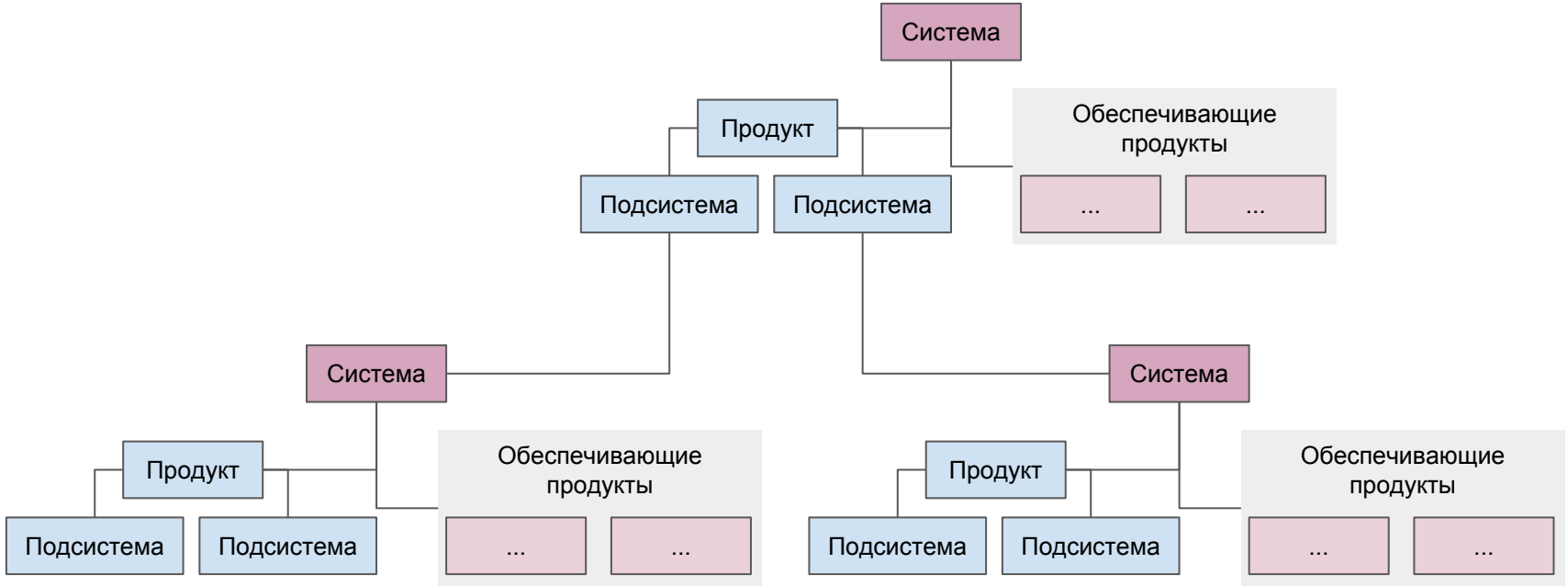
Подход “сверху-вниз”

- Традиционная инженерия работает “снизу-вверх”, создавая компоненты и интегрируя их пока не получится желаемая система
- Это подходит для хорошо сформулированных проблем и простых систем
- Но не подходит для сложным систем с большим количеством связей
- Системный инженер начинает с рассмотрения системы в целом и разрабатывает требования на этом уровне
- Далее рассматриваются подсистемы для которых разрабатываются свои требования и так далее
- Процесс останавливается, когда получено полное представление о системе

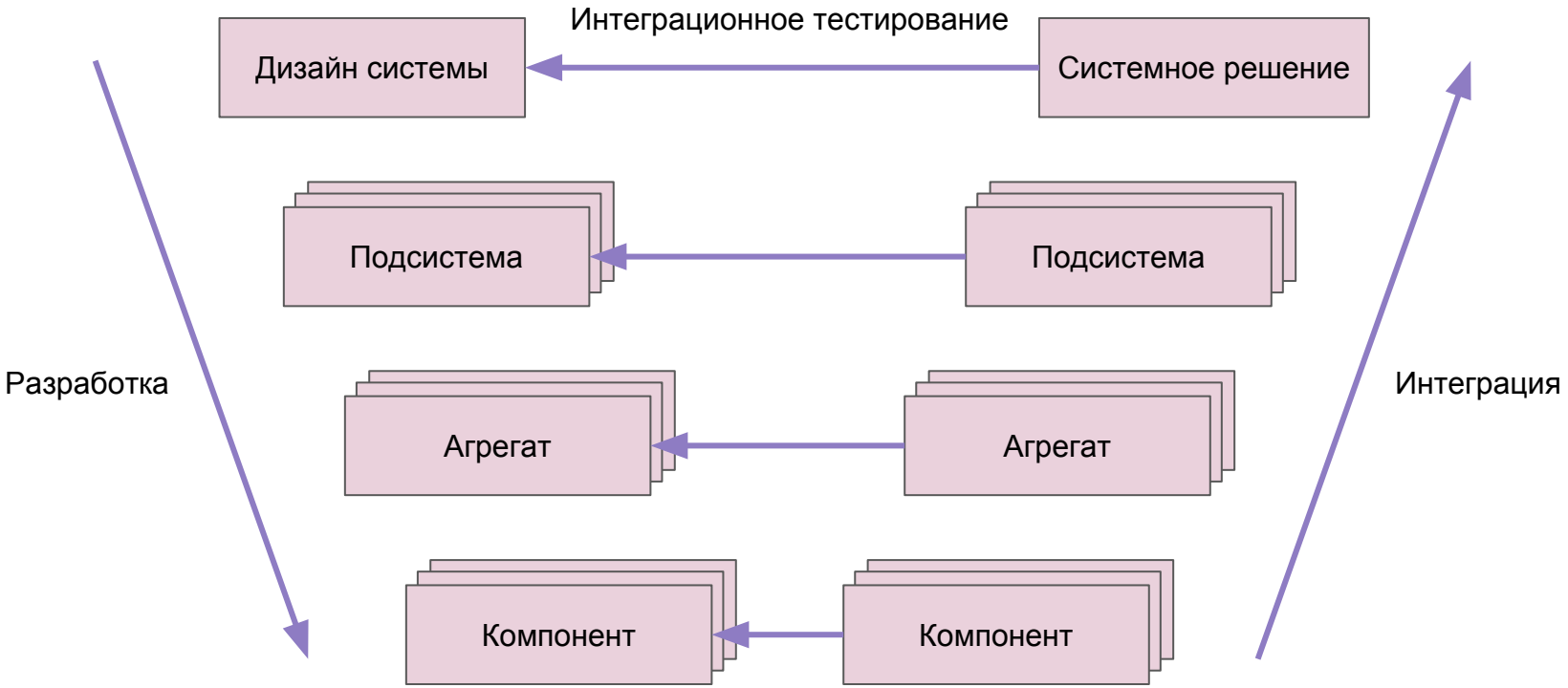
Подход “сверху-вниз”



Подход “сверху-вниз”



Подход “сверху-вниз”



Инженерия требований

- Полные и точные требования являются фундаментом для успеха системы
- После получения от бизнеса первоначальных требований происходит их детализация до уровня самого маленького компонента.
- ВАЖНО: Плохие требования никогда не могут быть исправлены хорошим проектированием.
- Важным свойством требований является их трассируемость, то есть способность проследить:
  - Как требования верхнего уровня реализуются на более низких уровнях
  - Как низкоуровневые требования связаны с верхнеуровневыми
- Трассируемость нужна как разработчику, так и клиенту, так как позволяет отследить, что в проект включено все, что требовалось и только оно

Фокус на полном ЖЦ

- Системный инженер фокусируется не каком-то определенном этапе, а учитывает весь ЖЦ системы
- Ошибочным является стремление удешевить этап реализации без рассмотрения того, как это скажется на затратах во время следующих этапов
- Например, дешевая машина с высоким уровнем затрат при использовании (расход бензина, запчасти) является худшим выбором по сравнению с дорогой машиной с низкими накладными расходами

Оптимизация и баланс системы

- Системный инженер ориентирован на оптимизацию на системном уровне, так как оптимально спроектированные подсистемы не обязательно формируют оптимальную систему.
- Примером может служить установка двигателя от болида F1 на машину семейного типа
- Разрабатывая систему надо иметь в виду социальные, этические, культурные, психологические факторы

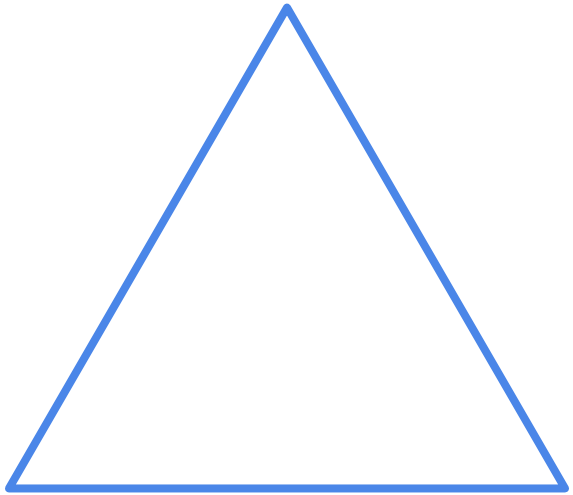
Интеграция различных дисциплин

- Трудно представить систему, разработка которой требует специалистов только одной дисциплины
- Например, создание системы “самолет” потребует работы инженеров разных дисциплин -- аэронавтика, металлургия, безопасность, электроника, логистика, тестирование, поддержка
- Также потребуются специалисты неинженерных дисциплин -- маркетинг, юристы, финансы
- Интеграция всех этих дисциплин является одной из важнейших частей системной инженерии ввиду сложности системы, контрактных связей между сторонами, их географической разделенности

Управление

- Системный инженер -- это не только техническая, но и управленческая роль
- Проектный менеджмент должен поставить продукт **вовремя с нужной функциональностью и не выходя за рамки бюджета**
- Системный инженер, инженер по требованиям и проектный менеджер работают в связке для того, чтобы так и случилось

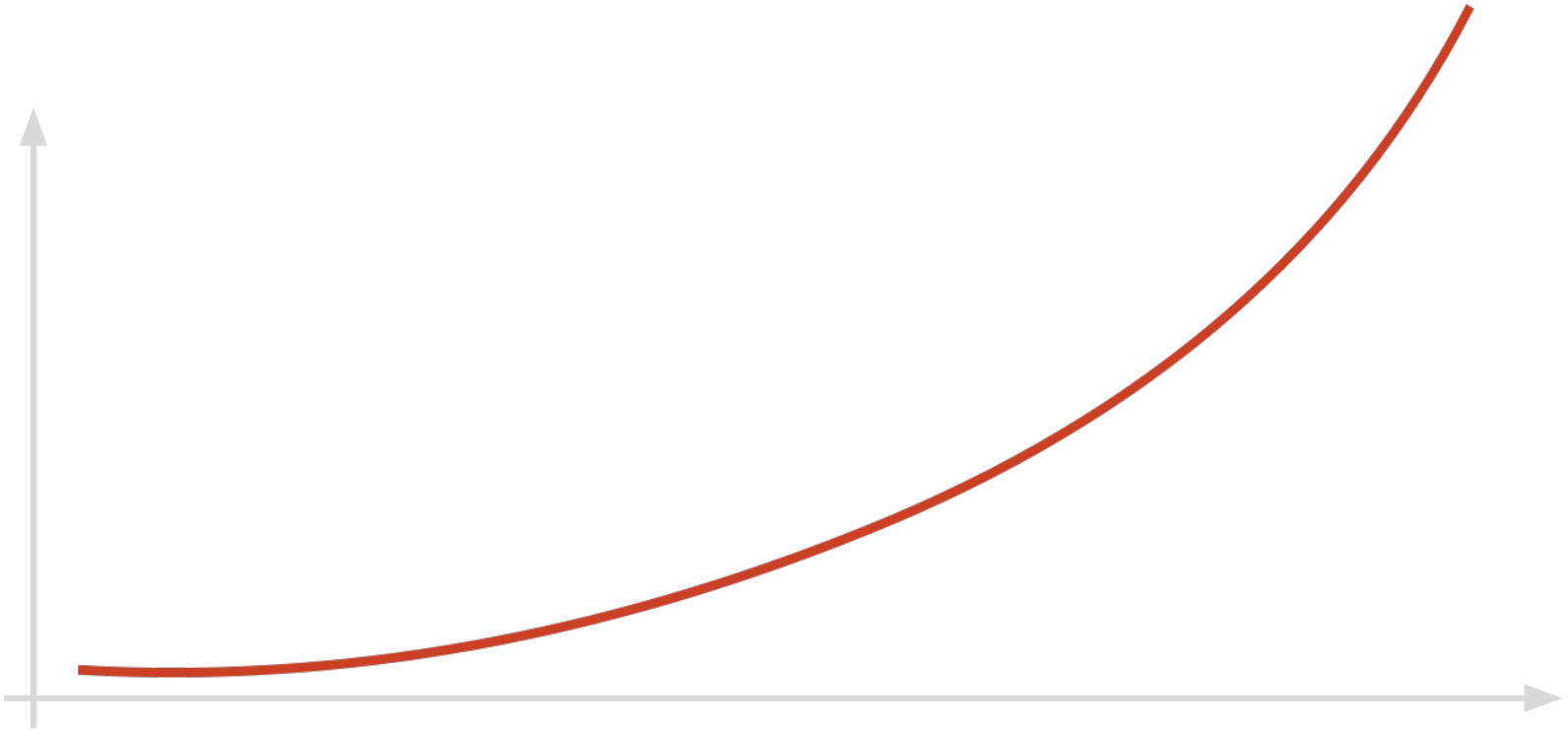
Функциональность



Бюджет

Сроки

Рост стоимости изменений



Требования ⇨ Проектирование ⇨ Код ⇨ QA ⇨ Внедрение ⇨ Эксплуатация

Команда разработки

Команда по Бруксу (1975)

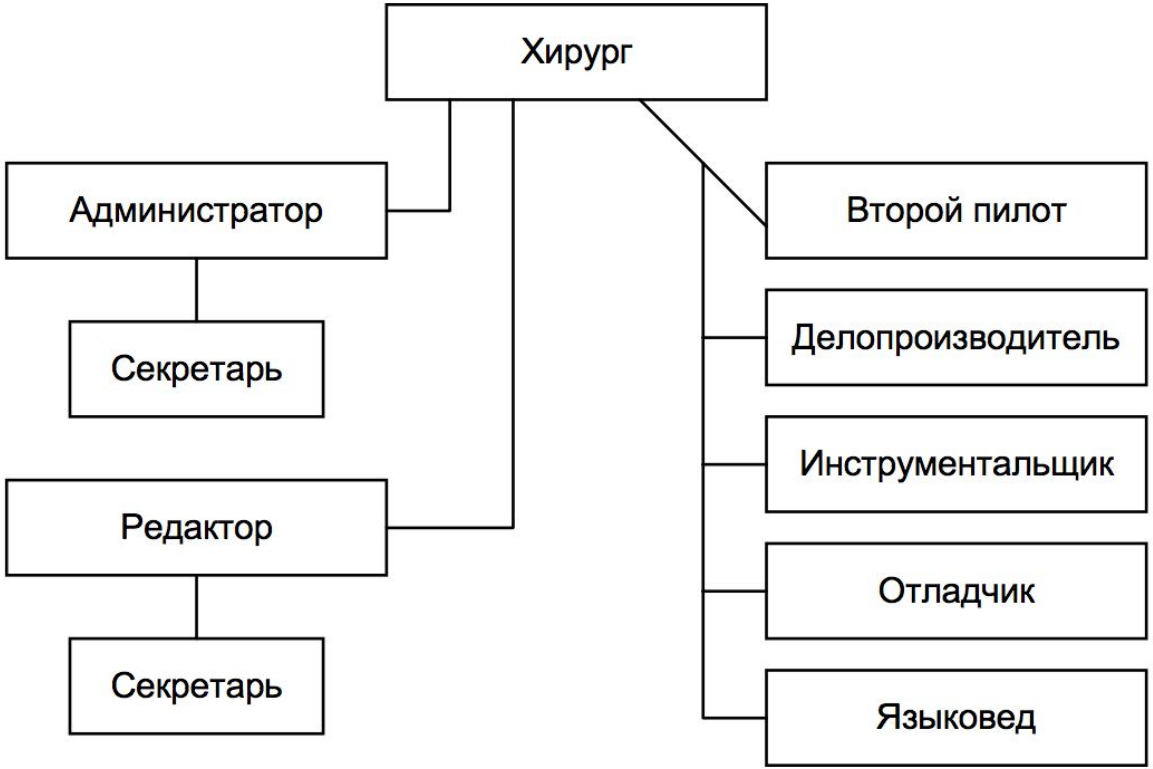
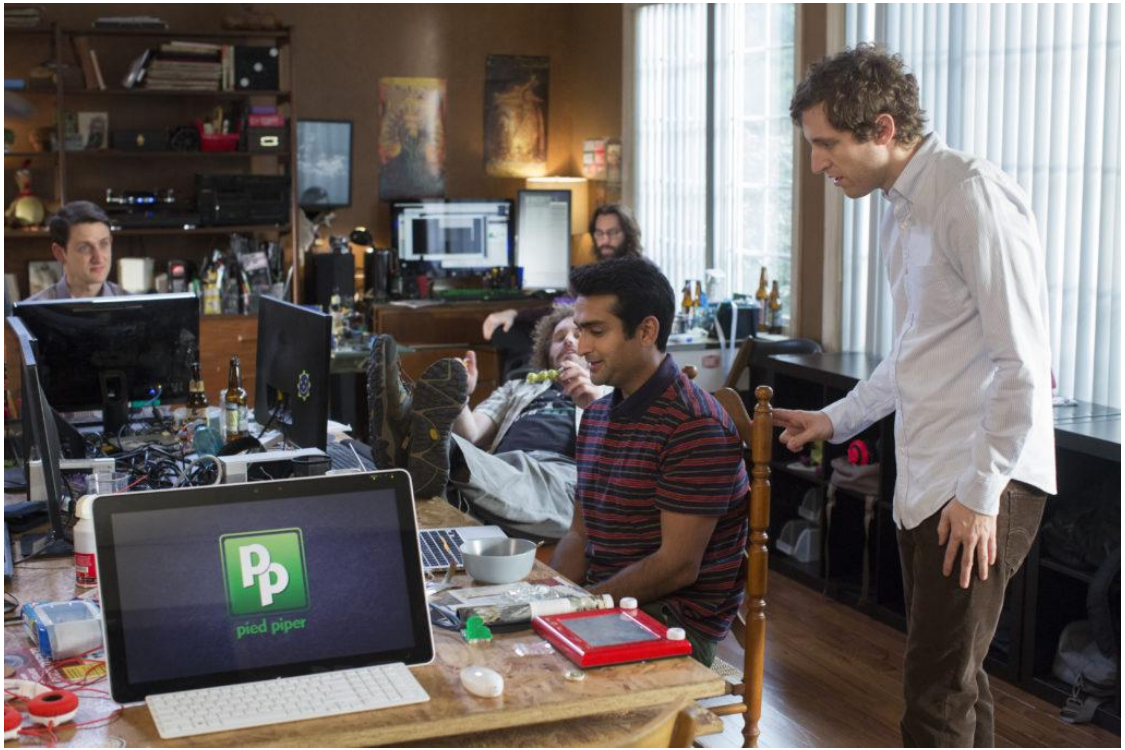


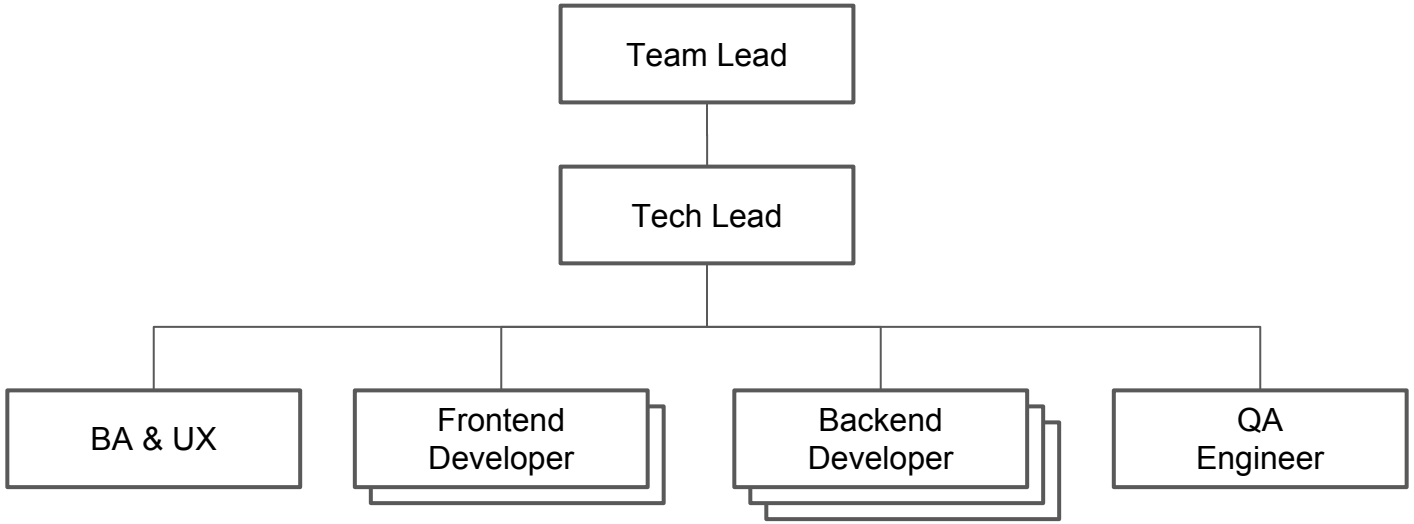
Схема контактов между сотрудниками в бригаде из 10 человек

Эволюция команды: стартап

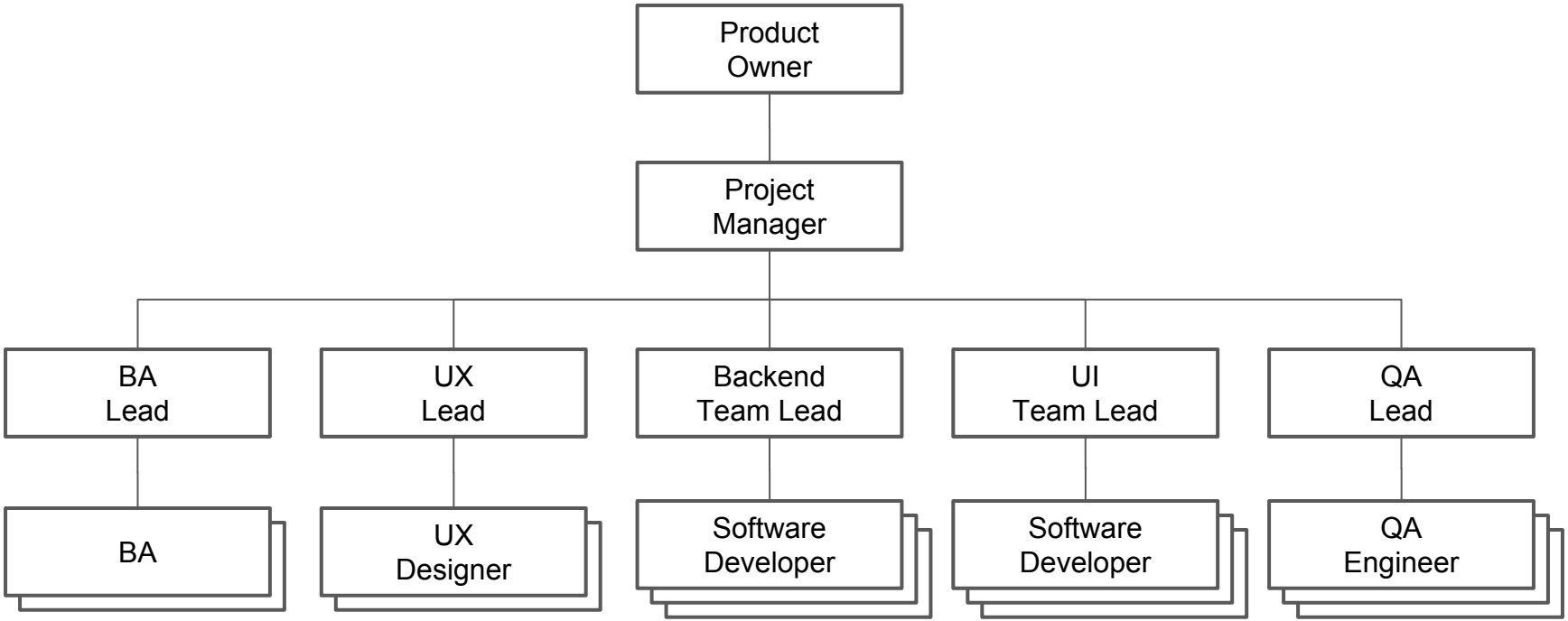


В сериале ”Кремниевая долина (Silicon Valley)” стартап “Pied Piper” -- это: трое программистов (включая основателя), финансовый директор (он же биздев) и инвестор (он же хозяин дома)

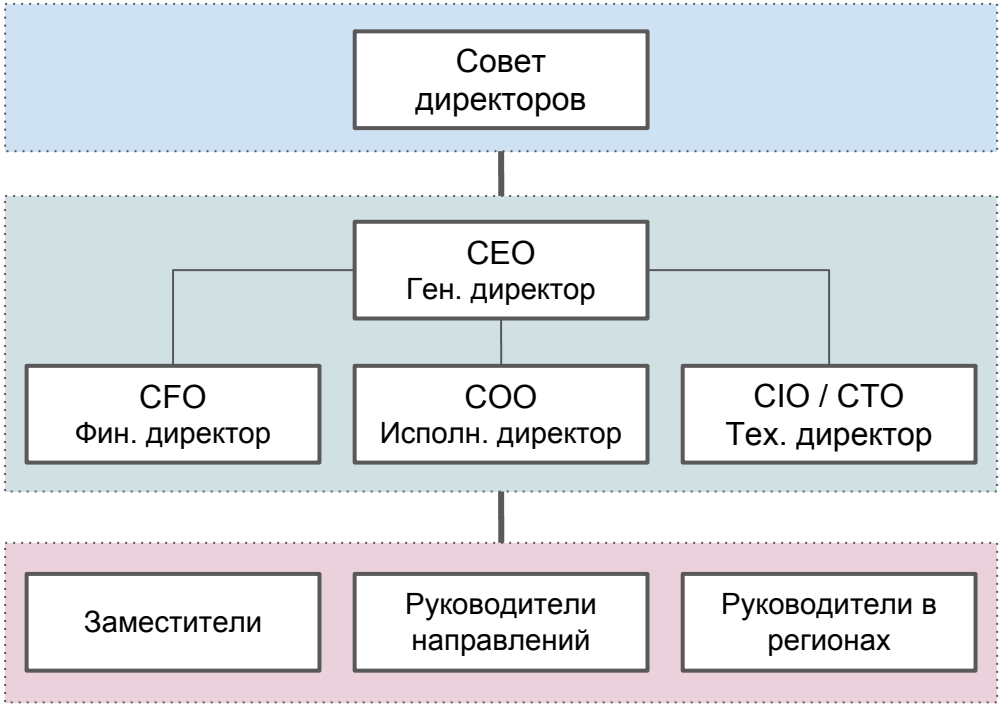
Эволюция команды: небольшой проект



Эволюция команды: один из проектов компании



Эволюция команды: структура верхнего уровня



Amazon

Apple

Facebook

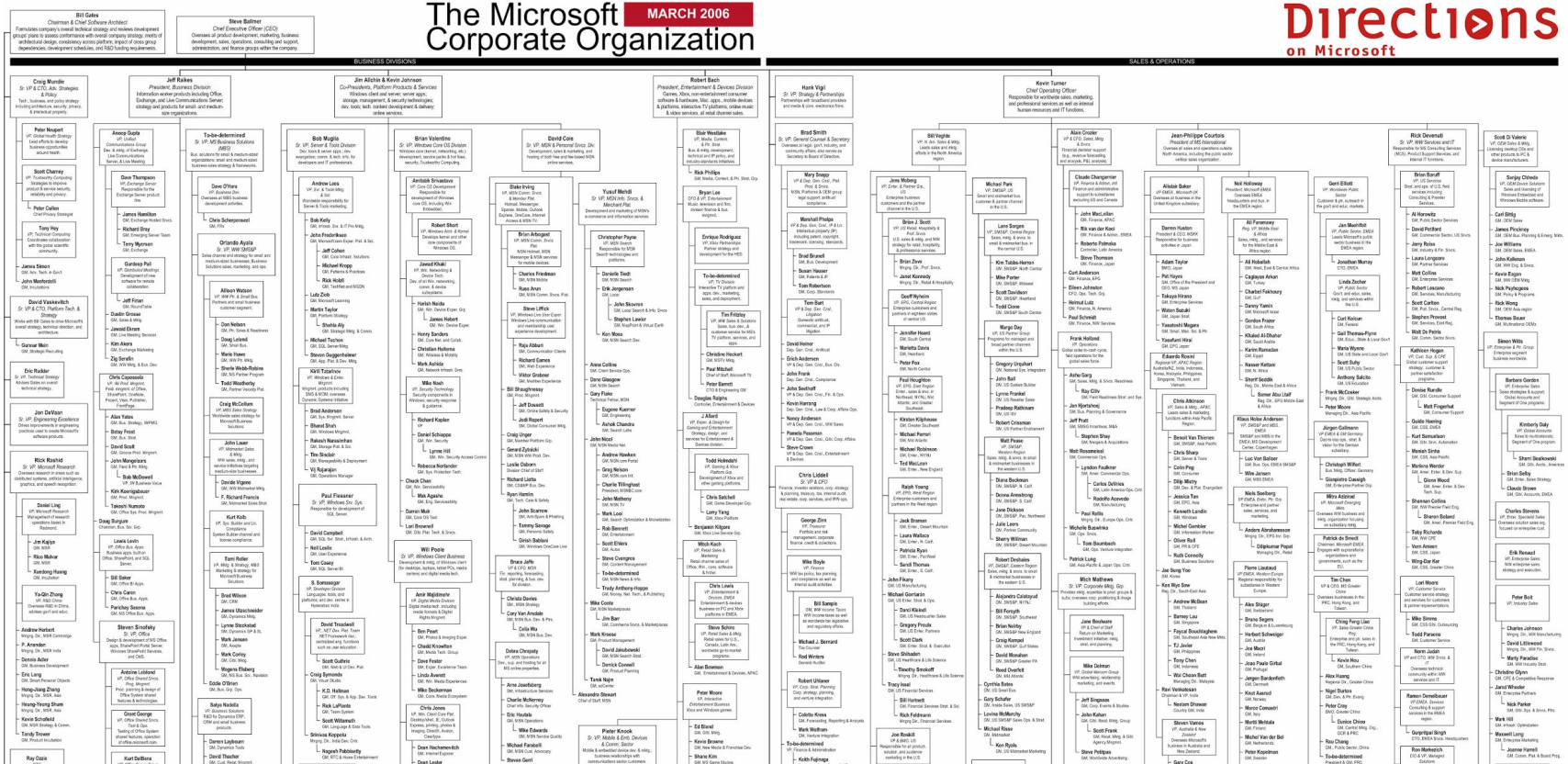
Mail.ru

Microsoft

Яндекс

Yahoo

Оргструктура может быть громадной



Роли, необходимые для создания продукта

Планирование и контроль

Product Owner	Бизнес-цели, стратегия
Product Manager	Руководство продуктом, ведение roadmap
Project Manager	Планирование релизов, контроль исполнения
Tech Lead	Оценка осуществимости, применяемые технологии

Непосредственная работа

Business Analyst	Выявление, анализ и описание требований, мокапы
UX Designer	Пользовательское взаимодействие, прототипы
Architect	Архитектура
Backend Developer	Разработка серверной части
Frontend Developer	Разработка фронтэнда
QA	Тестирование
IT Engineer	Техническая поддержка

Сферы деятельности в ИТ

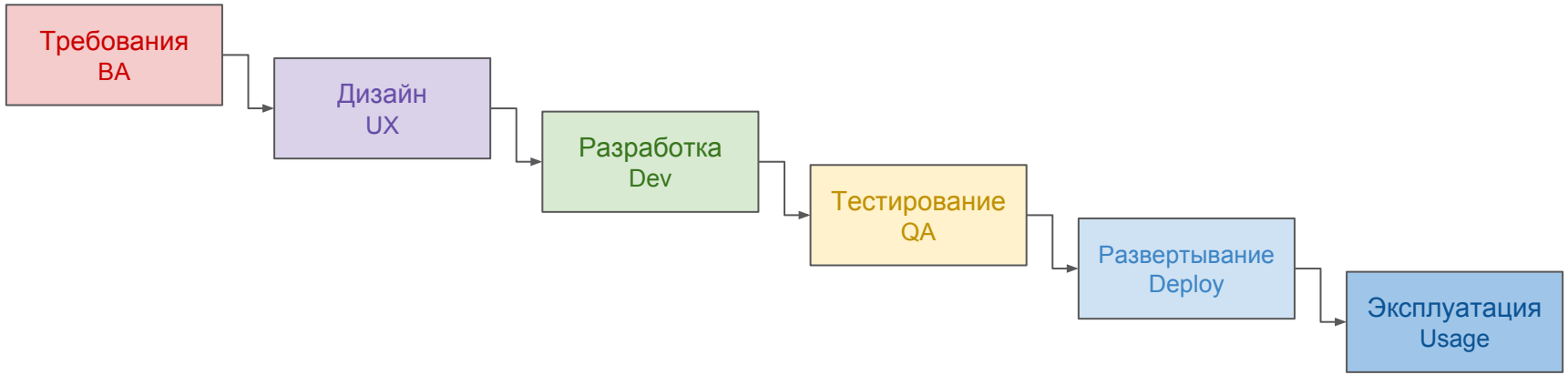
- Безопасность компьютеров и сетей
- Разработка и администрирование баз данных
- Обслуживание и поддержка
- Внедрение корпоративного ПО и консультирование
- Управление проектами
- Администрирование серверов и сетей
- Архитектура ПО и сетей
- Разработка ПО
- Системный анализ
- Опыт взаимодействия и информационная архитектура
- Дизайн интерфейсов и опыта взаимодействия

Методологии разработки

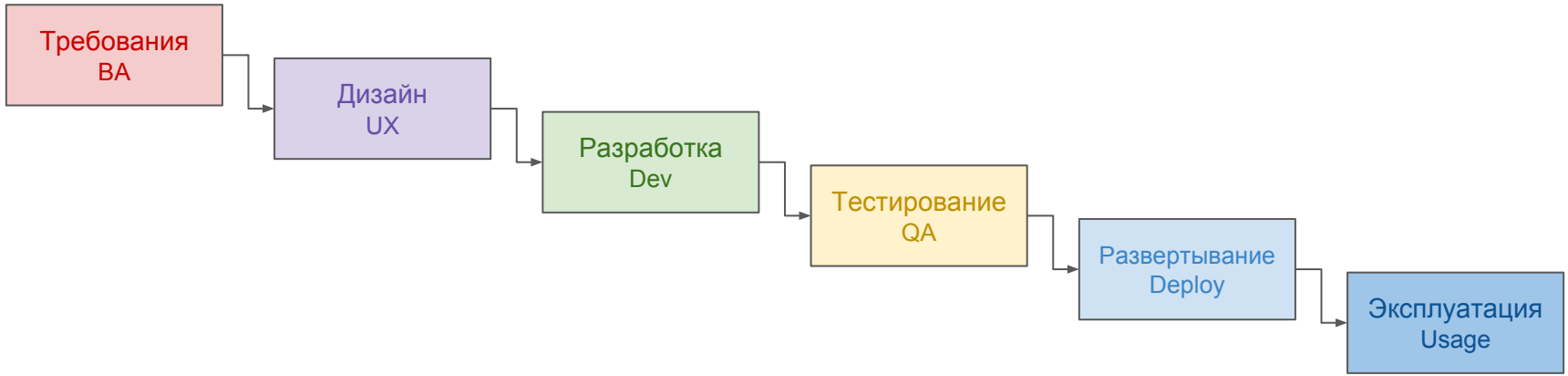
Некоторые методологии разработки продуктов

Водопадная (каскадная) модель	Переход от одной фазы к другой происходит только после полного и успешного завершения предыдущей
Инкрементная разработка	Выпуск базовой версии продукта и его доработка в последующих версиях
Итеративная модель	Разновидность инкремента, создание рабочего прототипа будущего продукта, его итеративное улучшение
Спиральная модель	Разновидность инкремента с акцентом на анализ рисков на каждом этапе
Быстрая разработка (RAD)	Разновидность инкремента, на каждой интерации работа ведется параллельно с последующей интеграцией в единый прототип
Agile	Итеративная разработка короткими итерациями

Водопад, каскад (Waterfall)



Водопад, каскад (Waterfall)



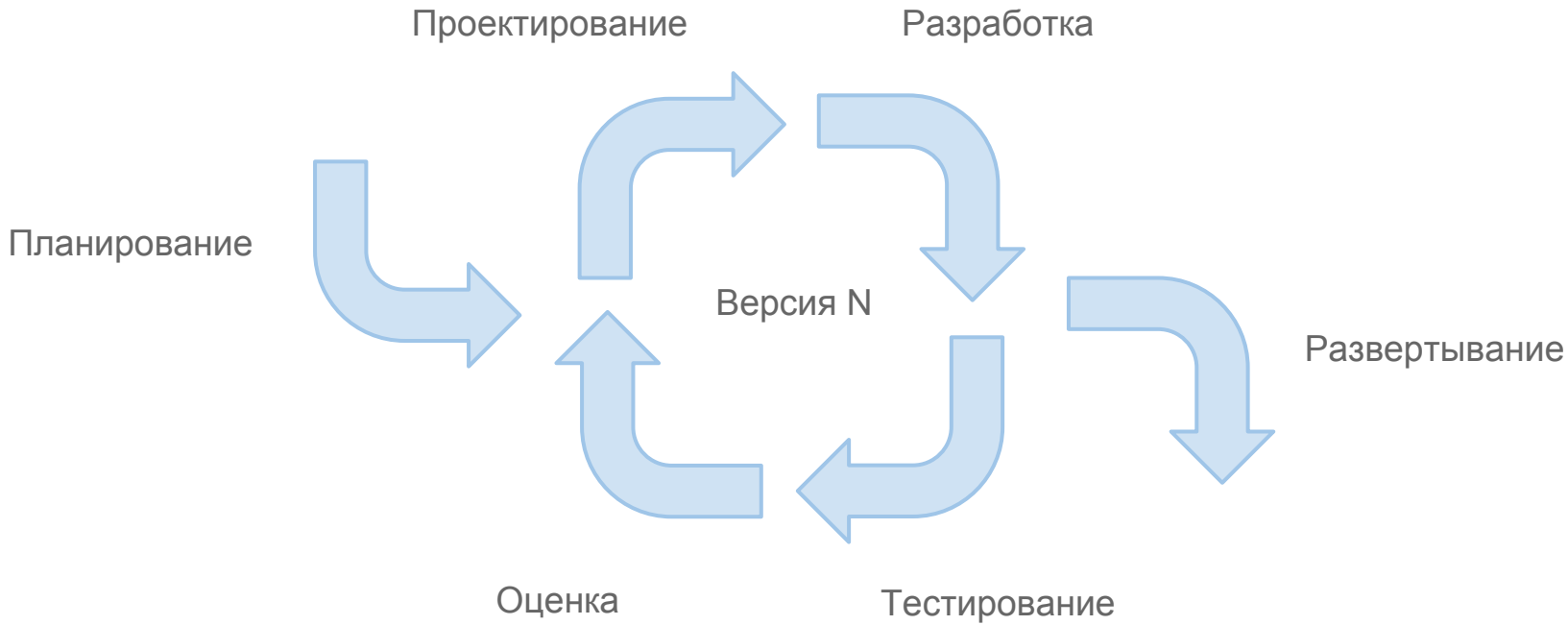
Преимущества

- Фиксированная стоимость разработки
- Простое управление проектом

Недостатки

- Нужна полная и четкая спецификация
- Нет возможности что-либо изменить в уже начатом проекте
- Стоимость внесения изменений высока

Инкрементная VS итеративная модель



Инкрементная VS итеративная модель

Промежуточные продукты могут не иметь ценности

Инкрементная разработка

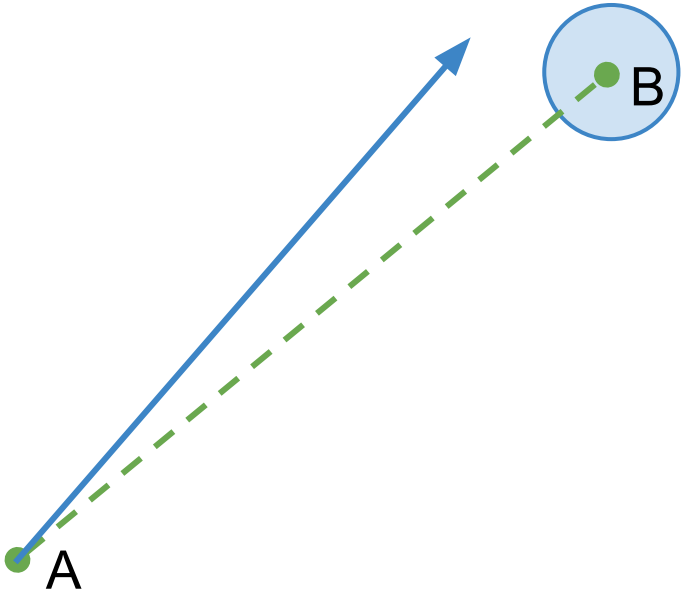
Конечный продукт может быть не оптимален

Готовый продукт после каждой итерации

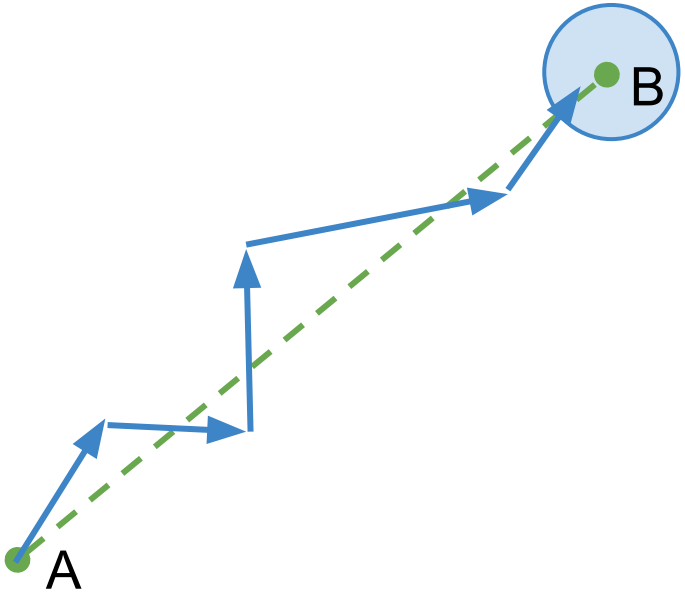
Итеративная разработка

Продукт дорабатывается под нужды заказчика

Водопад VS итеративный подход

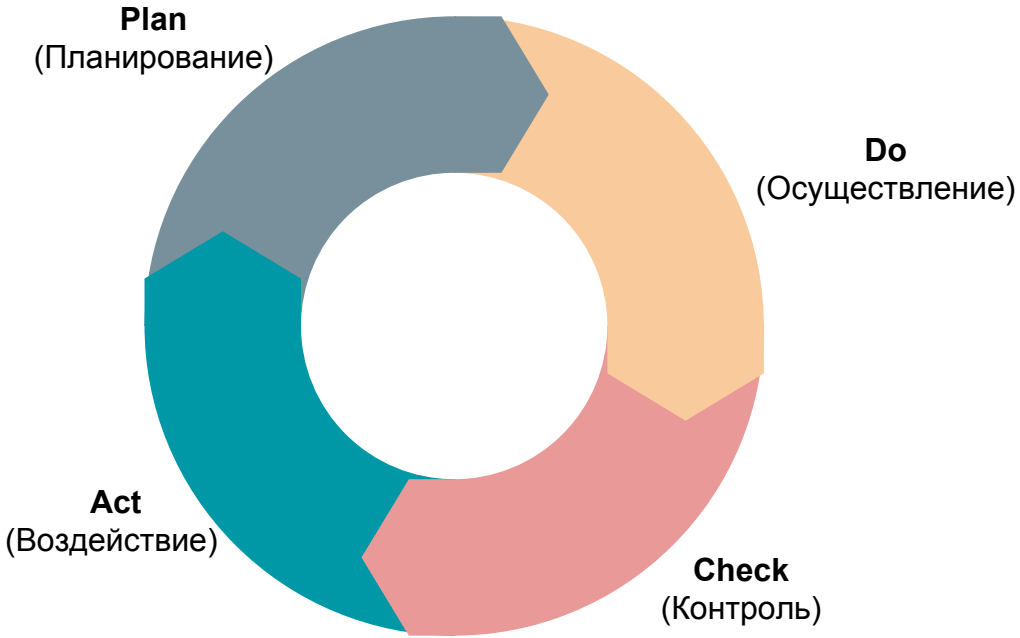


Водопад



Итеративный подход

Цикл PDCA (Деминг, Шухарт)



Принципы Agile

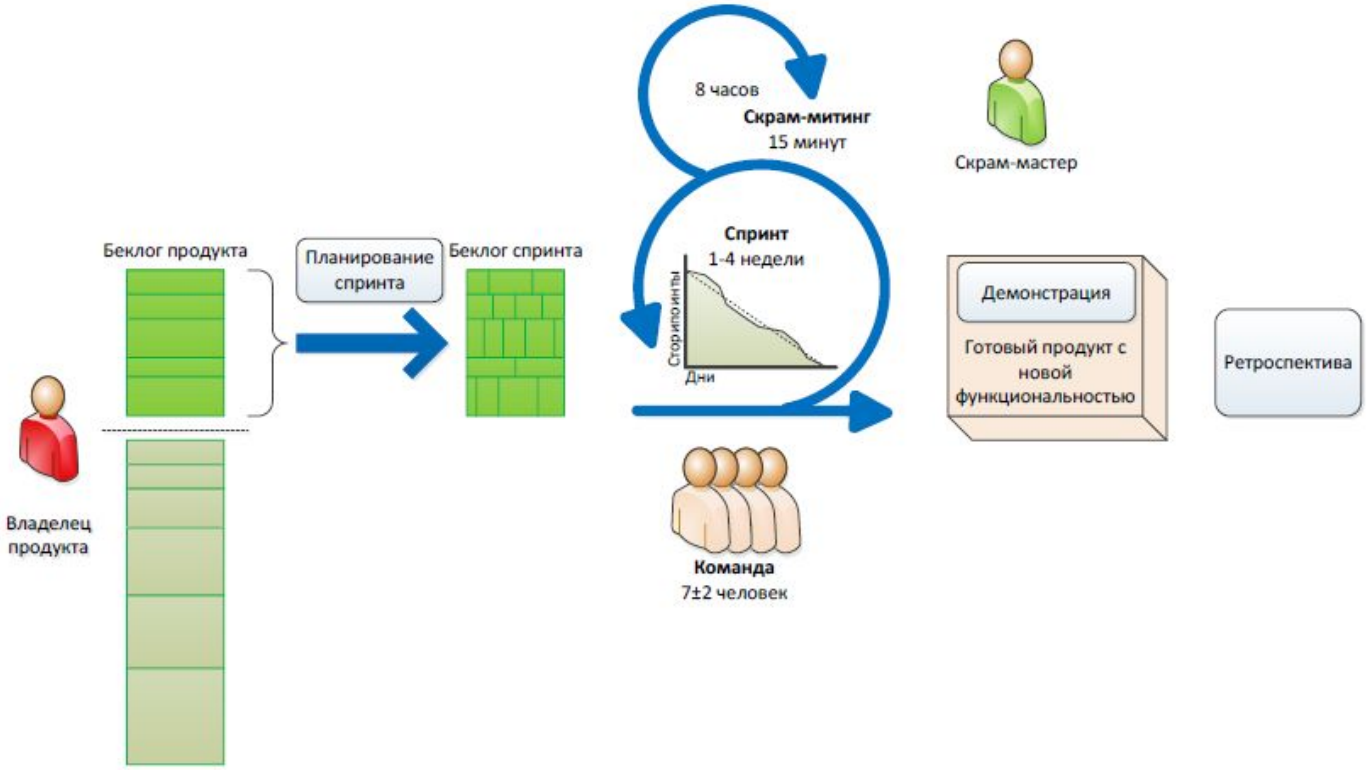
Ценности

- Люди и взаимодействие между ними
- Работающий продукт важнее документации
- Сотрудничество и выстраивание партнерских отношений с заказчиком важнее контракта
- Готовность к изменениям

Принципы

- Основная задача -- удовлетворить заказчика
- Требования могут быть изменены
- Постоянная обратная связь
- Лучший способ коммуникации -- лицом к лицу
- Постоянно работающий продукт
- Частые итерации
- PDCA-цикл
- KISS-подход
- Качественная команда = качественый код = качественный продукт
- Самоорганизация, а не управление

Скрам


8 часов
Скрам-митинг
15 минут
Скрам-мастер
Беклог продукта
Планирование спринта
Беклог спринта
Спринт
1-4 недели
Сторипоинты
Дни
Демонстрация
Готовый продукт с новой функциональностью
Ретроспектива
Владелец продукта
Команда
7±2 человек

Водопад VS итеративный подход VS agile

	Водопад	**Итерации**	**Agile**
Цели	Определяются для каждой фазы (входы/выходы)	Завершенные требования передаются в разработку и тестирование	В рамках спринта создается работающая фича
Проектирование	Объем работ проекта определяется PO, *все* фичи должны быть проработаны до их программирования	Объем каждой итерации определяется PM и согласуется с PO, в итерацию попадают только проработанные фичи	Команда определяет объем исходя из приоритетов и ресурсов, в спринт попадает то, что не будет изменено и может быть сделано за время спринта
Оценка затрат	PM рассчитывает оценку для всего проекта и согласует ее с PO	PM дает оценку для каждой итерации	Оценка дается командой на митинге, проводимом скрам-мастером, оценка затем может быть пересмотрена
График работ	Па фазам	По итерациям	По спринтам

Водопад VS итеративный подход VS agile

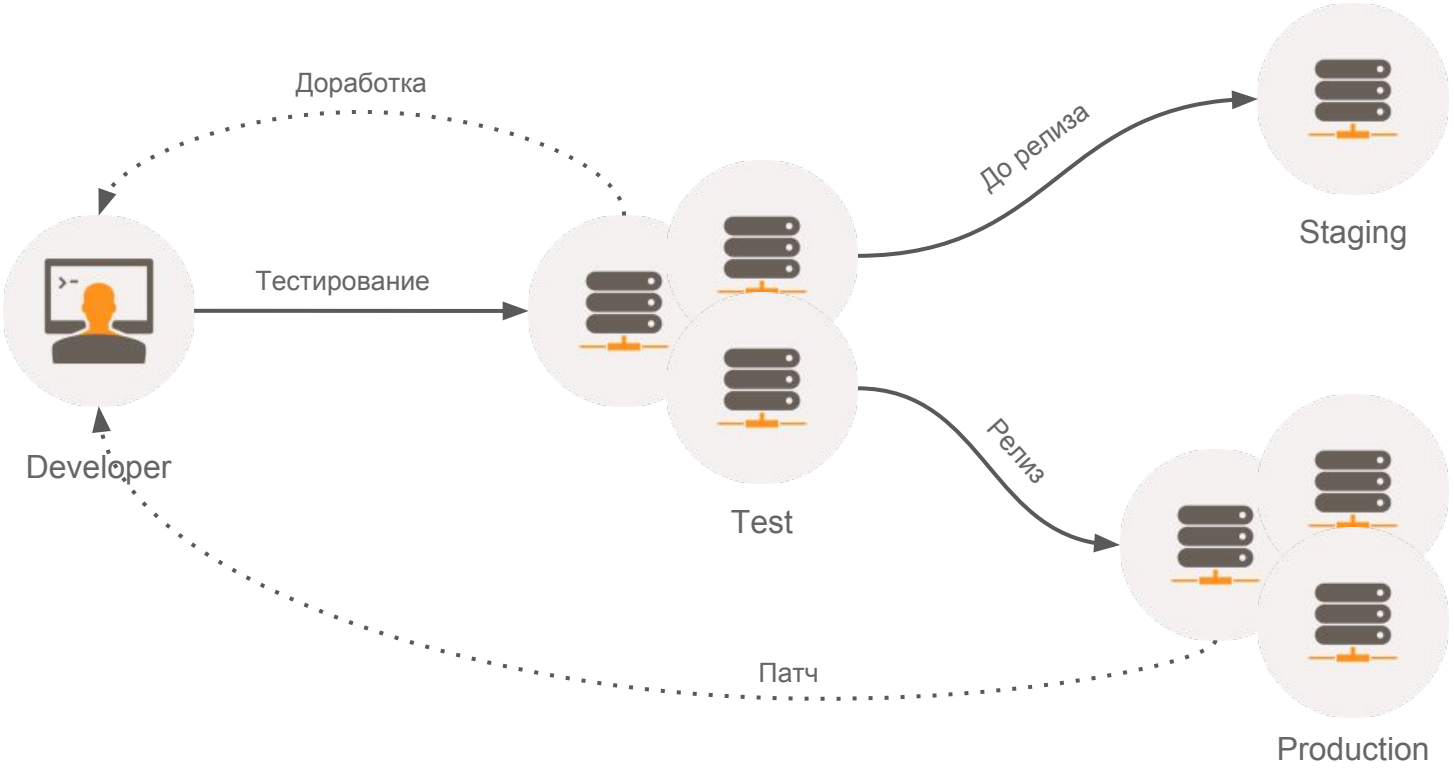
	Водопад	**Итерации**	**Agile**
Качество	Фокус меняется от фазы к фазе, тестирование в конце проекта	Переход фокуса от проектирования к разработке/тестированию в рамках каждой итераци	Фокус на всем ЖЦ, тестирование внутри спринта
Учет рисков	Отсутствует, “тушение пожаров” на фазе тестирования	Присутствует, риски идентифицируются и учитываются в следующих итерациях	Присутствует, риски идентифицируются и учитываются в следующих спринтах
Обратная связь	По заверешении проекта	После каждого релиза	После каждого спринта

Водопад VS итеративный подход VS agile

	Водопад	**Итерации**	**Agile**
Процесс	Последовательное выполнение фаз	Проектирование, разработка и тестирование ведутся одновременно с перекрытием	Над фичами работают миникоманды, которые делают весь ЖЦ
Распределение ролей	Четкое разделение ролей	Четкое разделение ролей, но подмена возможна	Каждый может работать над любой фазой задачи (проектирование, разработка, тестирование)
Назначение задач и ответственность	Задачи назначает PM, он же ответственен за реализацию проекта	Назначение задач согласуется с PM, он ответственен за выполнение итерации	За спринт ответственна *вся* команда, каждый может взять себе задачу по вкусу
Отчетность	Процент выполненного	Процент выполненного	График сгорания задач
Совещания	Еженедельные (ежедневные) отчеты о состоянии дел	Еженедельные (ежедневные) отчеты о состоянии дел	Ежедневые стэндапы: что сделали, что будем делать, проблемы

График сгорания задач (Burndown Chart)

Схема развертывания



Примерный график работ: водопад

Примерный график работ: итерации

QA	QA	QA	D													
Dev	Dev	Dev	QA	QA	QA	D										
UX	UX	Dev	Dev	Dev	Dev	QA	QA	QA	D							
BA	BA	BA	UX	UX	Dev	Dev	Dev	Dev	QA	QA	QA	D				
			BA	BA	BA	UX	UX	Dev	Dev	Dev	Dev	QA	QA	QA	D	
						BA	BA	BA	UX	UX	Dev	Dev	Dev	Dev	QA	QA	QA

Успешность проектов

Успешные проекты	Проекты с проблемами	Неудачные проекты
29%	**52%**	**19%**

Усредненные данные за 2011-2015 годы
2015 CHAOS Report, Standish Group

Успешность проектов (в зависимости от его размера)

	Успешные проекты	Проекты с проблемами	Неудачные проекты
Огромные	**2%**	**7%**	**17%**
Большие	**6%**	**17%**	**24%**
Средние	**9%**	**26%**	**31%**
Небольшие	**21%**	**32%**	**17%**
Мелкие	**62%**	**16%**	**11%**

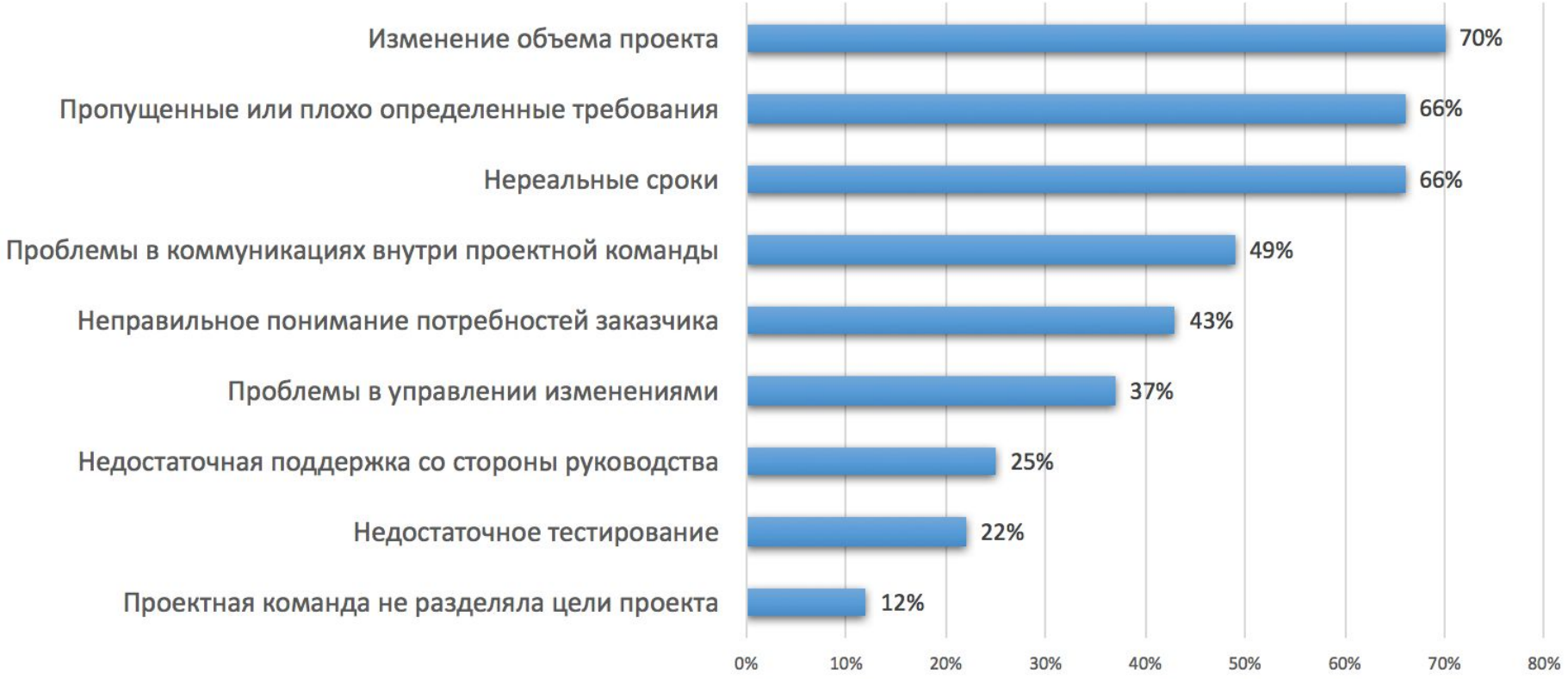
2015 CHAOS Report, Standish Group

Успешность проектов (Agile vs Waterfall)

	Agile vs Waterfall					
	Успешные проекты		Проекты с проблемами		Неудачные проекты	
Все	**39%**	**11%**	**52%**	**60%**	**9%**	**29%**
Большие	**18%**	**3%**	**59%**	**55%**	**23%**	**42%**
Средние	**27%**	**7%**	**62%**	**68%**	**11%**	**25%**
Мелкие	**58%**	**44%**	**38%**	**45%**	**4%**	**11%**

2015 CHAOS Report, Standish Group

Причины неудач проектов

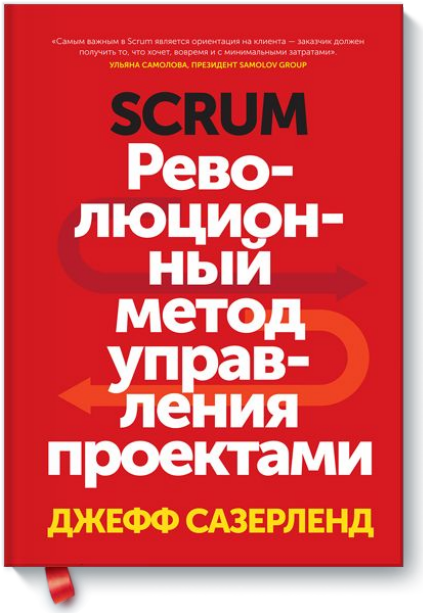


Факторы успеха проекта

Фактор	Вес
Поддержка руководства	20
Вовлеченность пользователей	15
Бизнес-процессы	15
Компетентность команды	13
Управление проектом	12
Навыки Agile	10
Ясные цели	6
Зрелость команды	5
Инструменты и инфраструктура	1

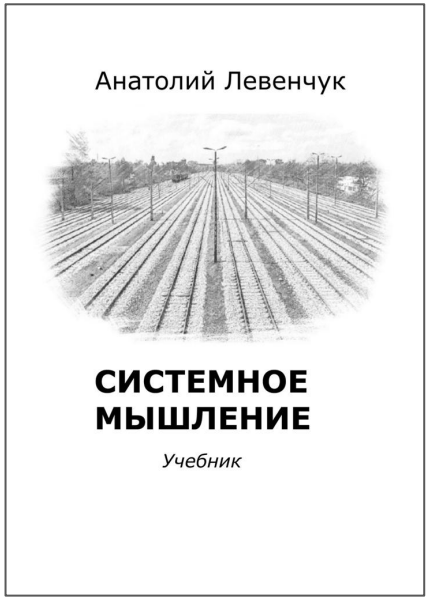
2013 CHAOS Manifesto
Standish Group

Дж. Сазерленд
Scrum. Революционный метод управления проектами



Ф. Брукс
Мифический человеко-месяц
или Как создаются программные системы

А. Левенчук
[Системное мышление]



В. Мизгулин
[Системный инженер]

Интересно про ИТ

- Мастер-класс Бориса Вольфсона. Основы Agile
- Ещё раз про семь основных методологий разработки
- Они пишут правильную вещь
- Agile vs Iterative vs Waterfall models
- Скрам Гайд (PDF)

Ссылки по системной инженерии

Системная инженерия является отдельной дисциплиной, но вы можете по своему желанию ознакомиться с нею:

- Системная инженерия по Анатолию Левенчуку
- Курс лекций Анатолия Левенчука + новый курс на Coursera
- Introduction to Systems Engineering by UNSW Australia (The University of New South Wales) (курс на Coursera.org)
- Guide to the Systems Engineering Body of Knowledge (SEBoK)