# Методы сокращения ЖЦ цикла разработки на основе принципа Системы – Систем

# Безуглый Дмитрий

- Около 20-лет опыта участия в проектах по созданию, развитию, внедрению и эксплуатации различных систем.
  - Max масштаб проекта 50 инженеров, около 30 чел-лет. Разработка ПО
  - Max бюджет проекта 2,5 млн долл. (ЦОД)
  - Max ROI проекта 400% (инвестиционный проект)
- Ведущий эксперт ООО «Системный Подход» с 2008 года

# Партнеры









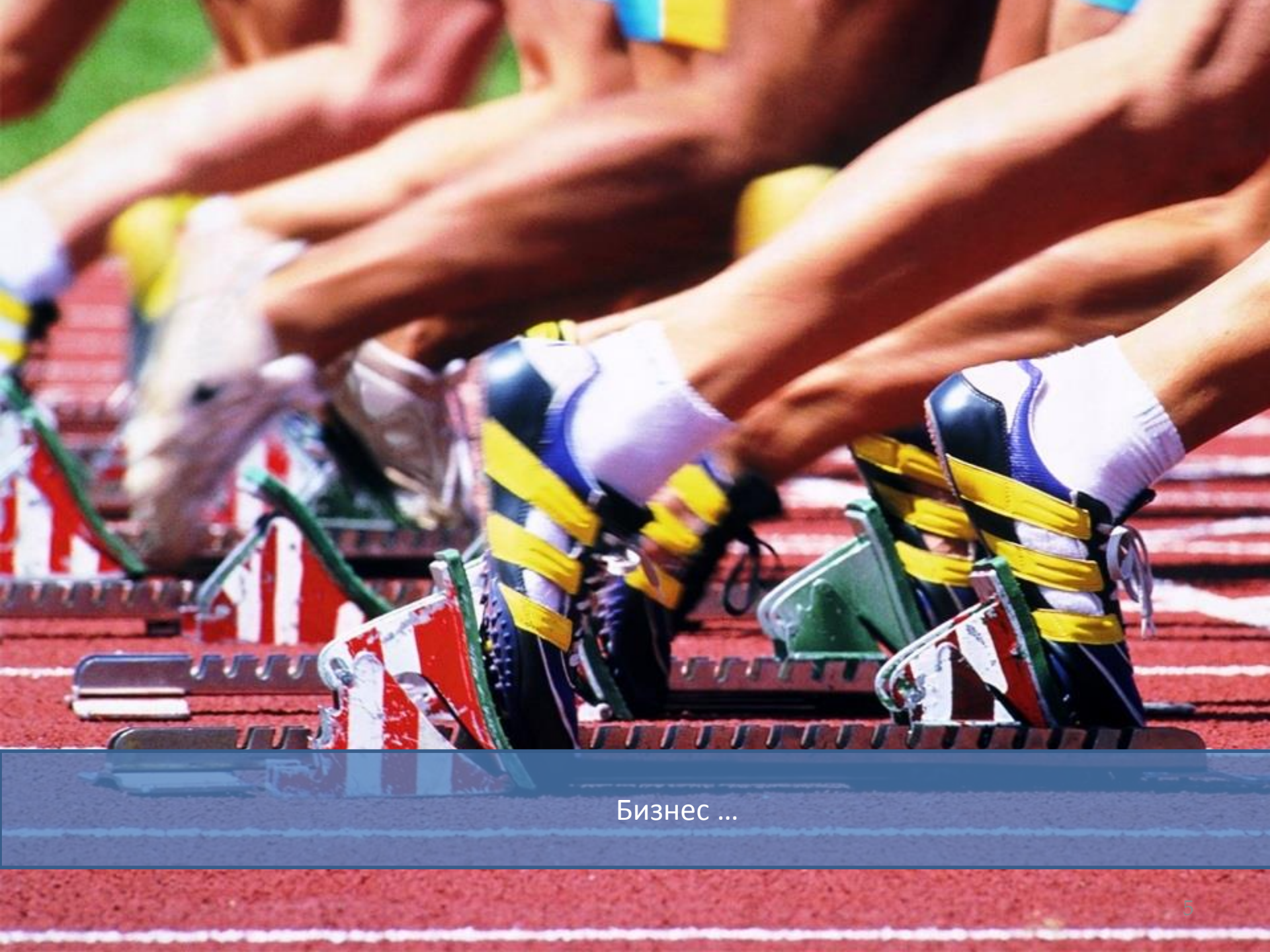







# Жизненный цикл разработки

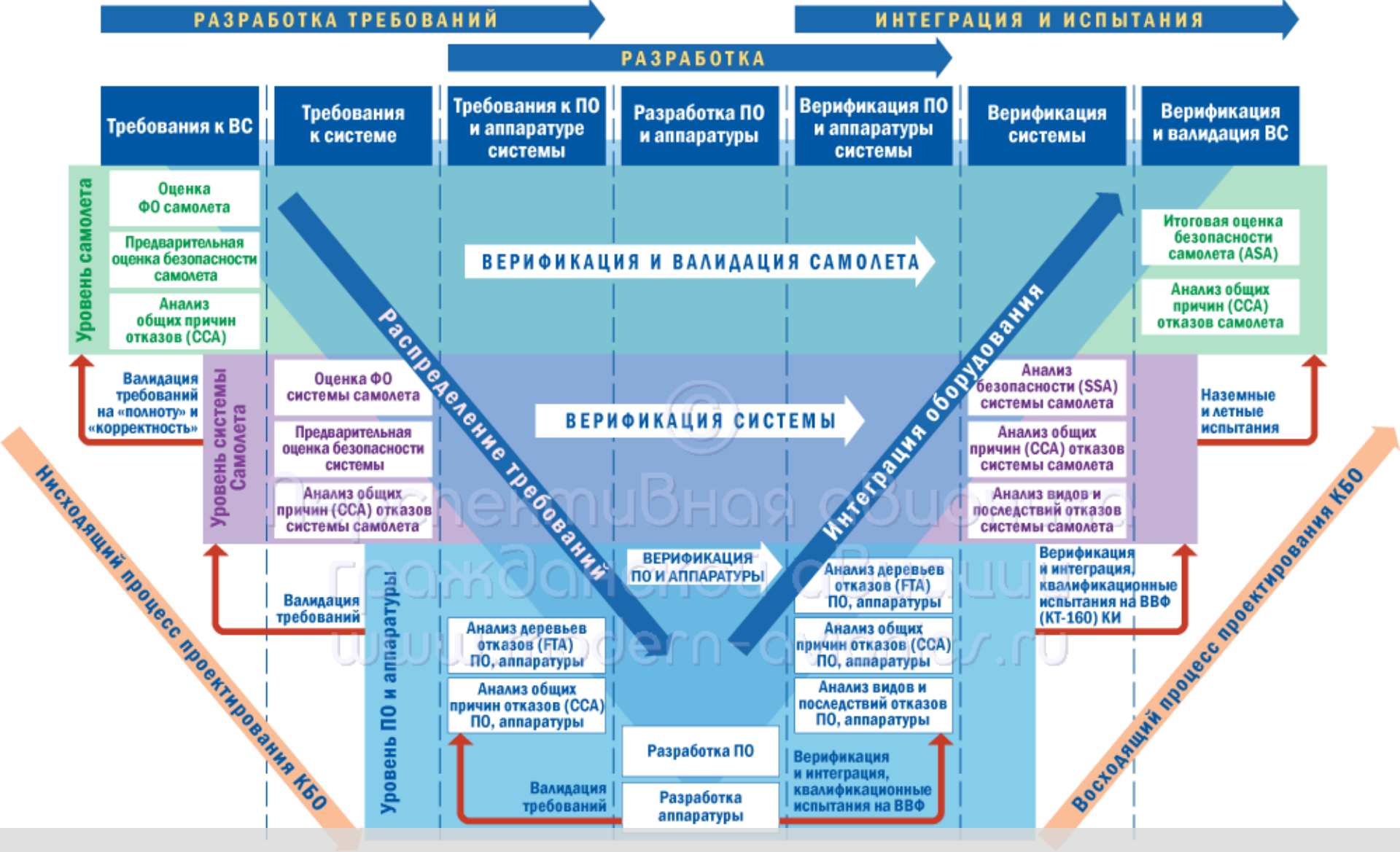
Время от появления идеи до предоставления реальной возможности использования (capability)



Бизнес …


… Ожидает, и …

…. Департамент ИТ




http://www.modern-avionics.ru/

Обеспечение качества

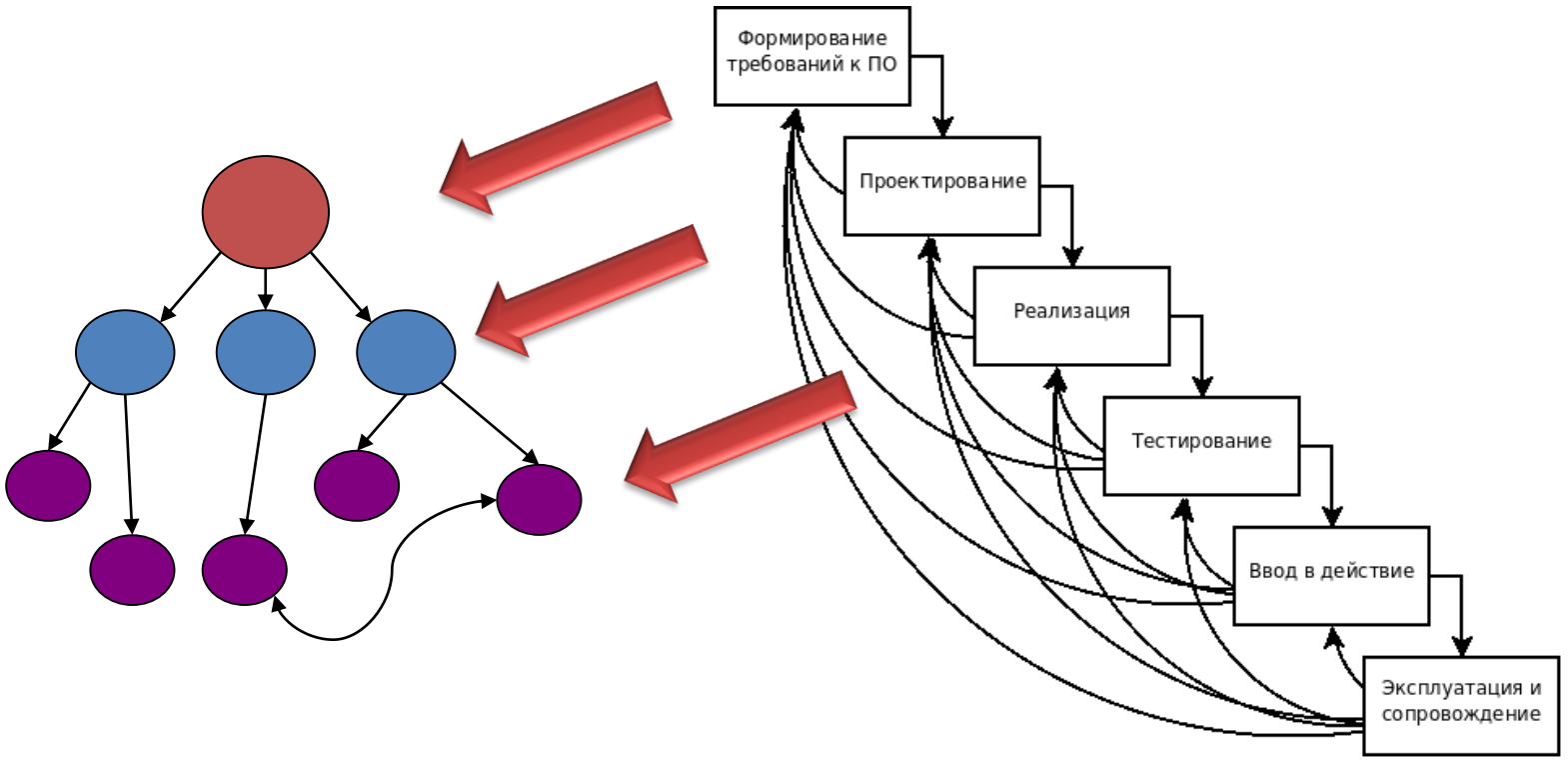
Документирование

Технические аудиты

Теория систем — это лекарство, которое превратилось в болезнь.

*Кен Уилбер*

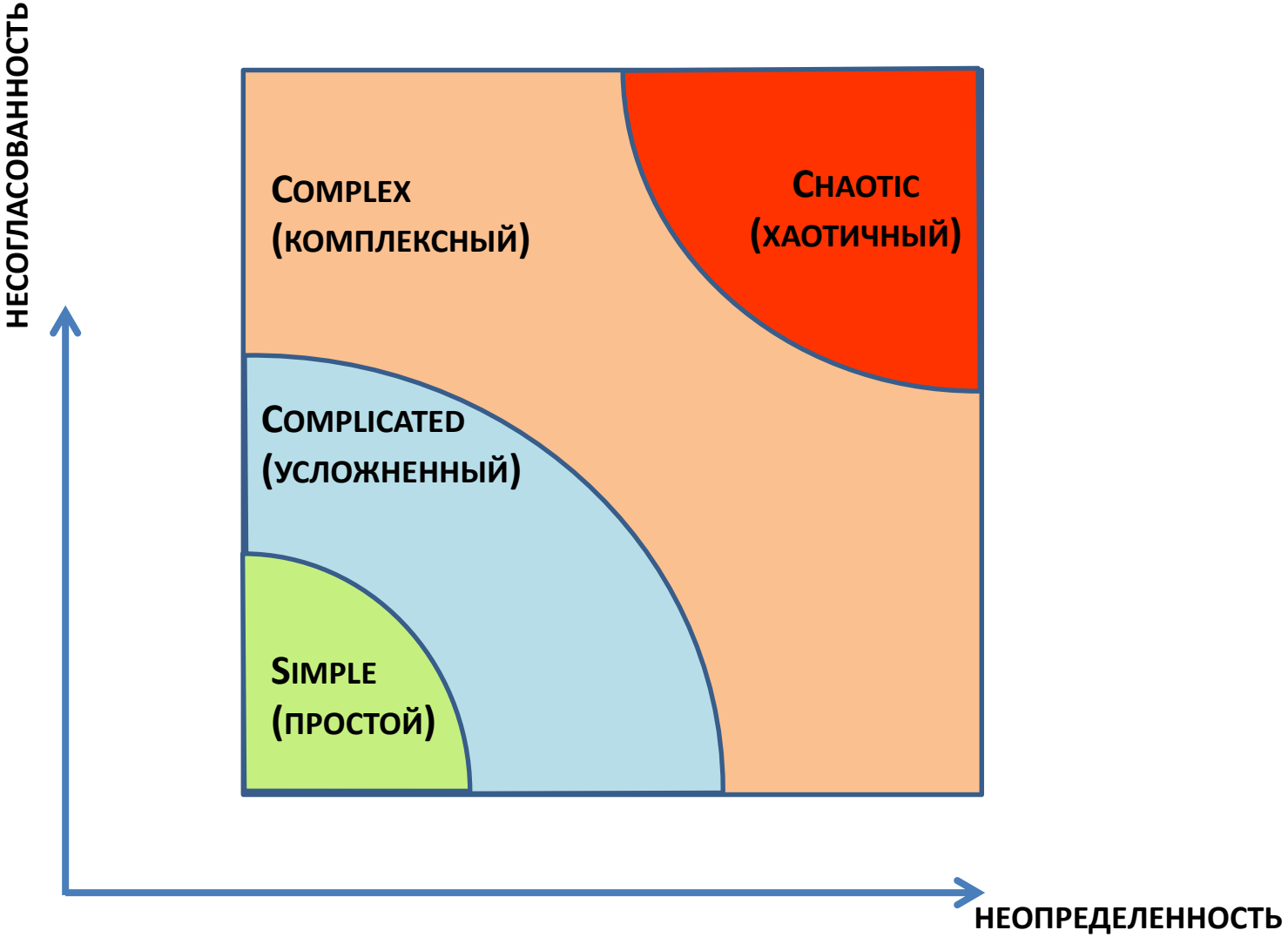
# Самое противное



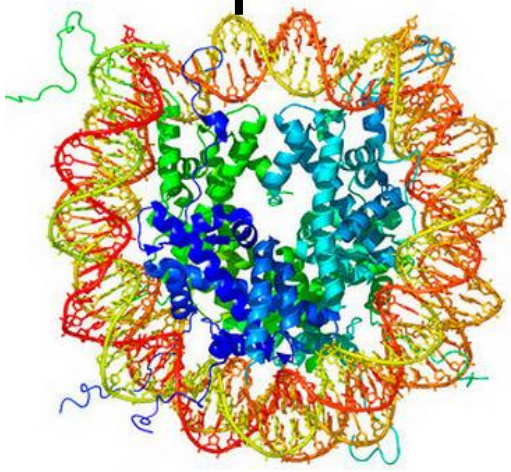
# But, but … that‘s agile and lean! Isn‘t it?

# В чем ключевая причина провалов в ИТ ?

# Сложность



# Усложненный и комплексный
# Complicated vs. Complex



- Тысячи частей,
  сотни шагов на сборку
- Кропотливая, тонкая работа,
  которую сложно завершить
- Все должно работать в определенной последовательности
- Конечная сборка должна отражать исходный план.
- **Отклонение от плана = дефект**

- Сложно предсказать детали поведения / результаты
- Результаты зависят от многих переменных
- Эти переменные
  сложно /невозможно предсказать надежно
- **Следует ожидать изменений и отклонений, затем учитывать это при планировании**

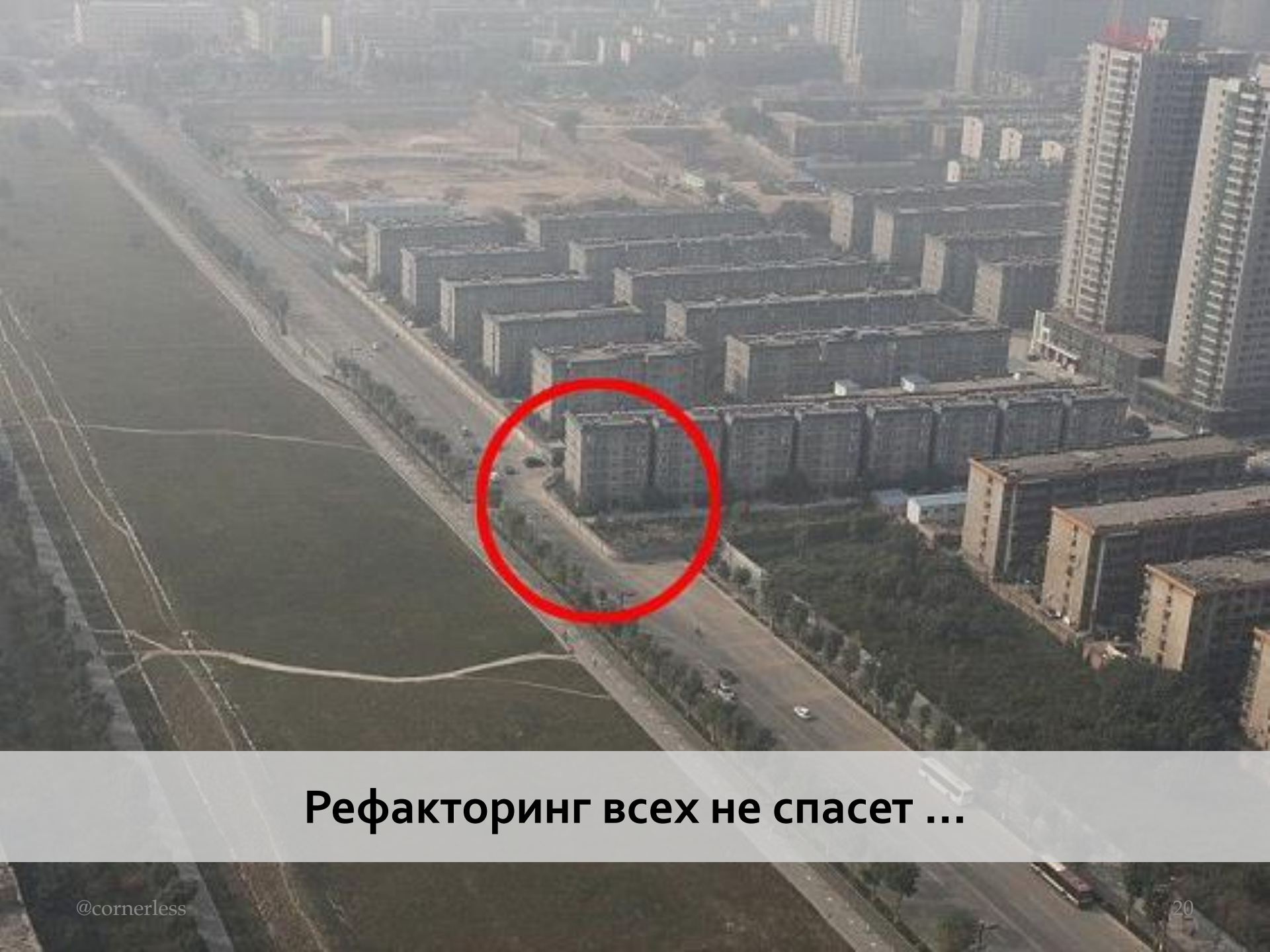
# Гибкие методологии уменьшают сложность самой системы ?

# Проблема №1 Внутренняя архитектура

Понятно что лучше чем так …

## Но и не так ...

## Проблема №2 Внешняя архитектура
## Представим организацию победившего Agile

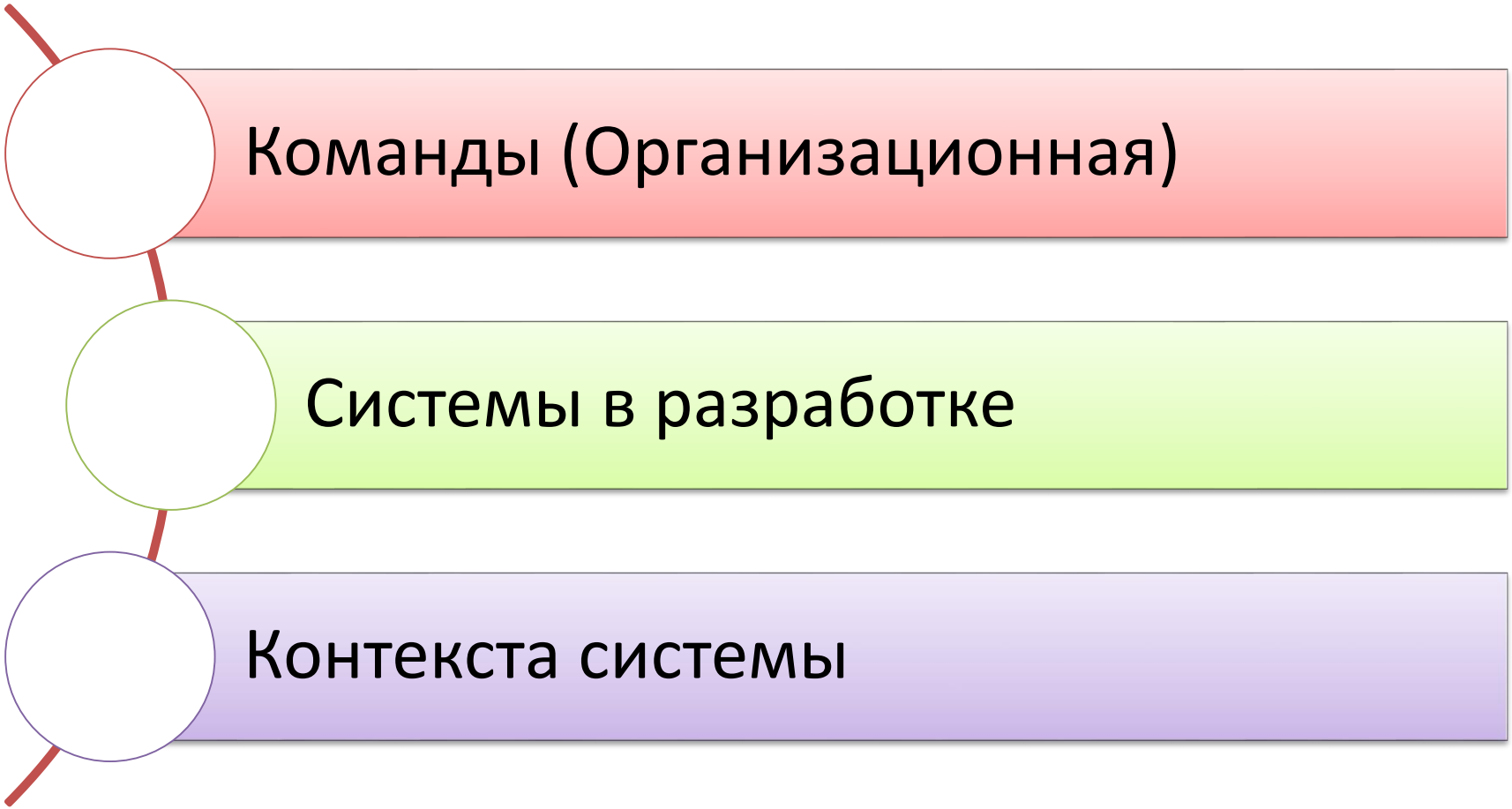


# Рефакторинг всех не спасет …

Среднестатистическая организация насчитывает порядка 500-700 систем, и около 20-40 команд разработки

Гибкие методологии не исключают, работы с требованиями, создания Архитектуры и документации, однако они и не определяют сколько либо пригодного процесса и его места в метологии

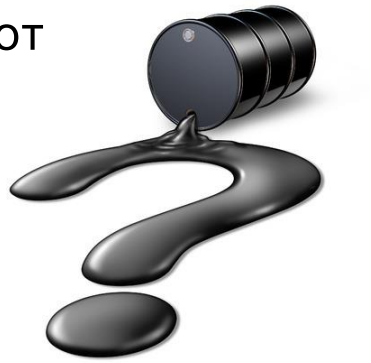
# Гибкость



Цепь изменений, мы по прежнему не знаем к чему на самом деле могут привести изменения

# Ключевые проблемы «Теория систем»

- Подход на основе системы-подсистем подразумевает выявление предназначения совокупности подсистем и полное раскрытие неопределенности на этапе работы с требованиями
- Однако:
  - На этом этапе информации недостаточно, поэтому требуются сложные технологии, и тем не менее полностью предсказать будущее не получается
  - Ошибки на этапе работы с требованиями запускают рекурсивные процессы стабилизации требующие значительного времени



# Ключевые проблемы «Гибкая разработка»

- Решая проблему «отклика» команды разработки для заказчика и локальной стабильности
  - Маскирует проблемы связанные со сложностью системы или ее окружения,
  - Что неизбежно влечет за собой отложенные и неконтролируемые последствия для систем класса Complicated или Complex.

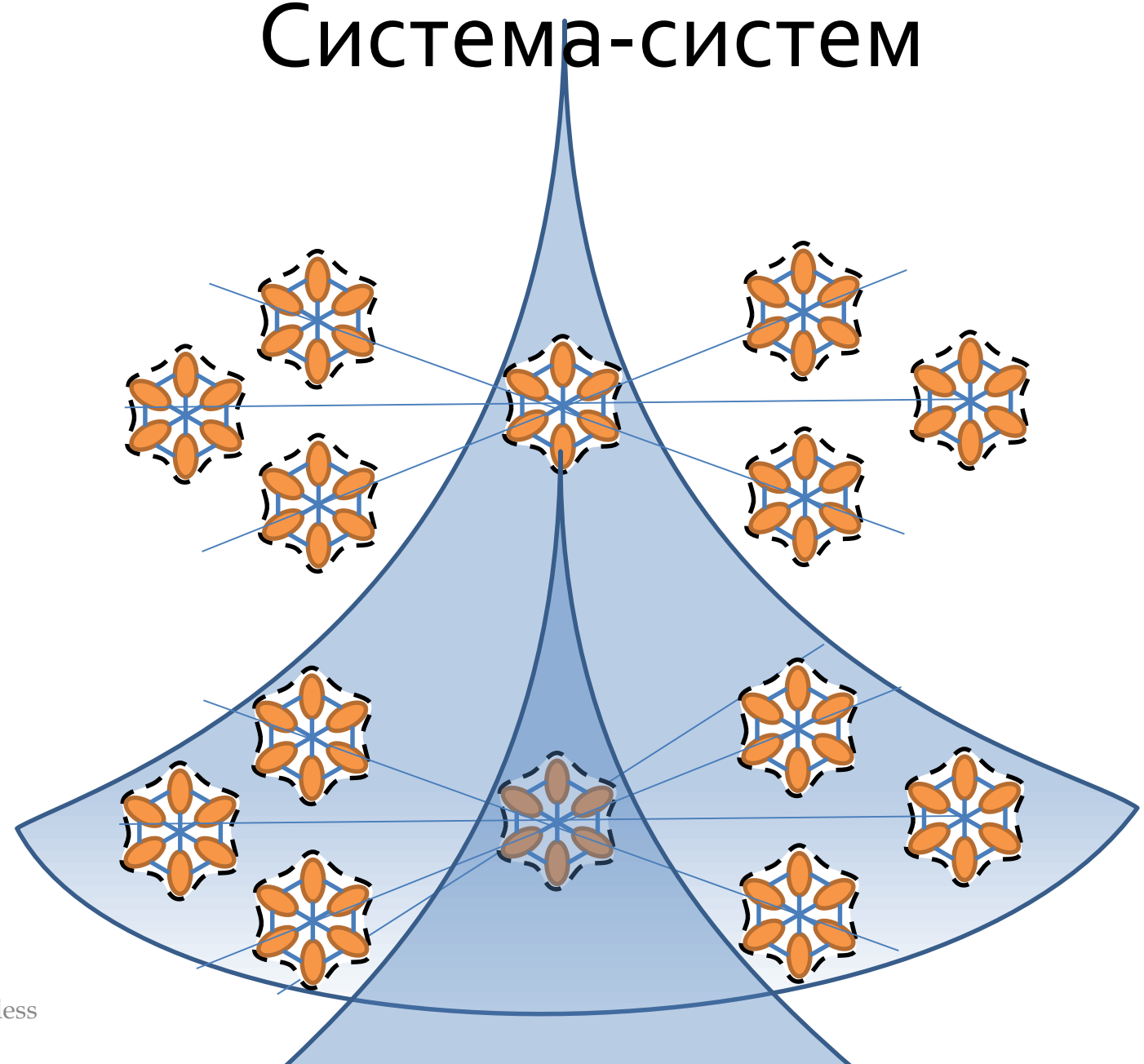
# Конечная структура разрабатываемой системы в точности отражает структуру команды которая ее создает

*Эмпирическое правило архитектуры*

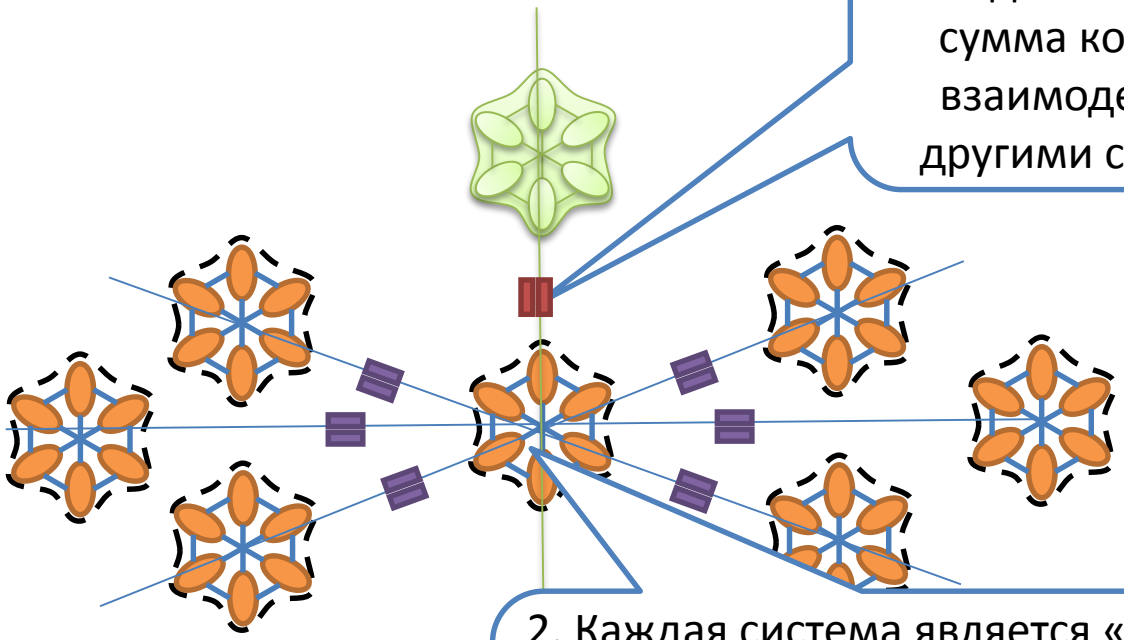
*Филипп Кратчен*

# Иии …

# Система-систем



# Что изменилось ?

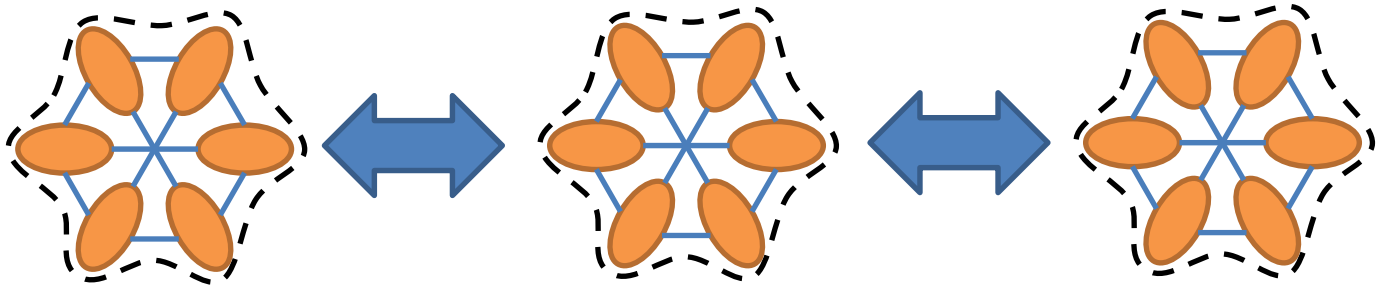


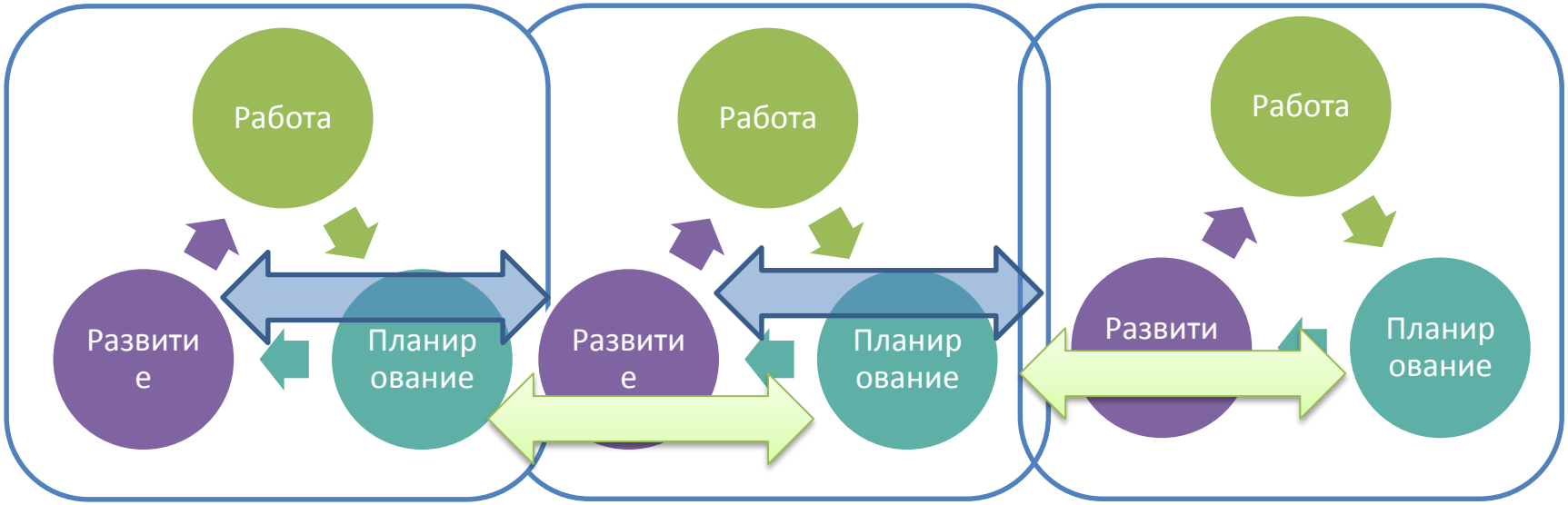
1. Контракт с надсистемой есть сумма контрактов взаимодействия с другими системами

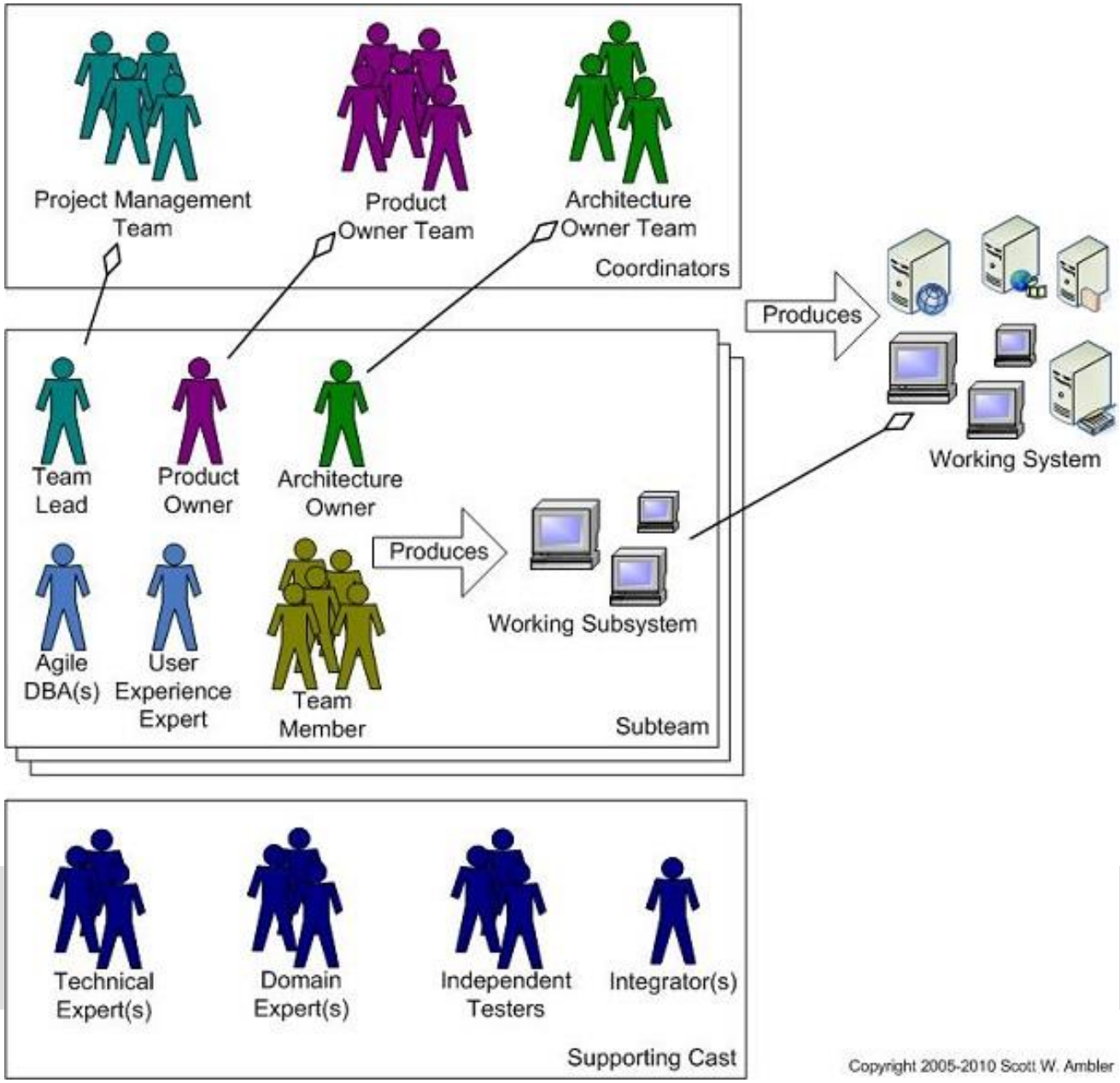
2. Каждая система является «живым» развивающимся организмом, что требует координации на этапах планирования , реализации и эксплуатации

# Принцип №1 Планирование низкой связанности

# Часть работ управление интеграцией на всех уровнях создания





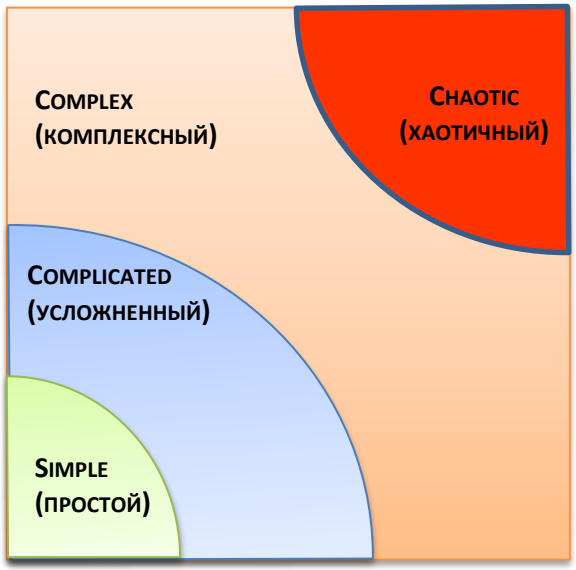


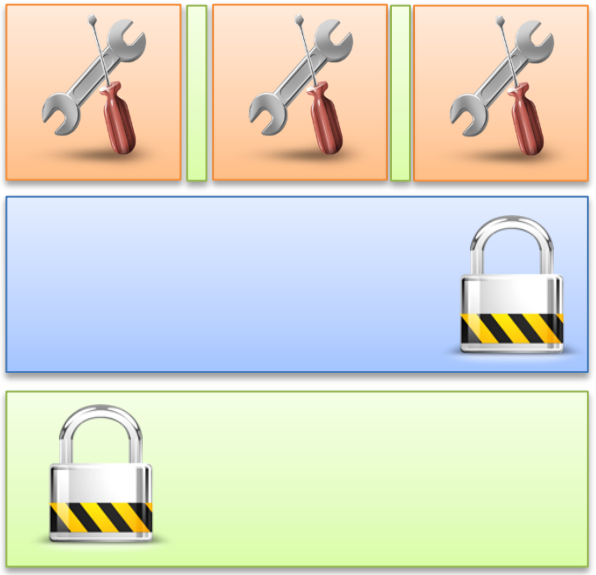
# Ключевой принцип решения проблемы наладить взаимодействие на всех этапах

*Создать структуру взаимодействующих команд*


**Принцип №2 Независимость слоев. Замок нельзя построить на зыбкой почте**

# Шаблон В каждый момент времени комплексный слой может быть только один



# Итоги

**Было**

- Система (complicated) технический объект, который сдается в эксплуатацию
- Детальные требования как контракт на разработку
- Проектирование архитектуры системы-подсистем

**Будет (уже есть)**

- Система (complex) социо-техническая система, которая живет
- Требования как поддерживающий механизм координированного развития
- Координированная эволюция

# Что еще ждет ?


# Спасибо за внимание !

**Дмитрий Безуглый**

**+7 915 09 09 700**

**https://www.facebook.com/dmitry.bezuglyy**

**@cornerless**

**bdl@system-approach.ru**

**ООО «Системный Подход»**

**https://www.facebook.com/SystemApproach**

**www.system-approach.ru**

# Прочее …

- Упомянутые презентации
  - http://www.slideshare.net/Cless/laf2014
  - http://www.slideshare.net/Cless/rapid-foresight-for-product-strategy
- Авторское право:
  - http://ru.123rf.com/profile_alexmit
  - http://ru.123rf.com/profile_somchaij
  - http://ru.123rf.com/profile_lightwise
  - http://ru.123rf.com/profile_everythingpossible
  - http://ru.123rf.com/profile_coramax