

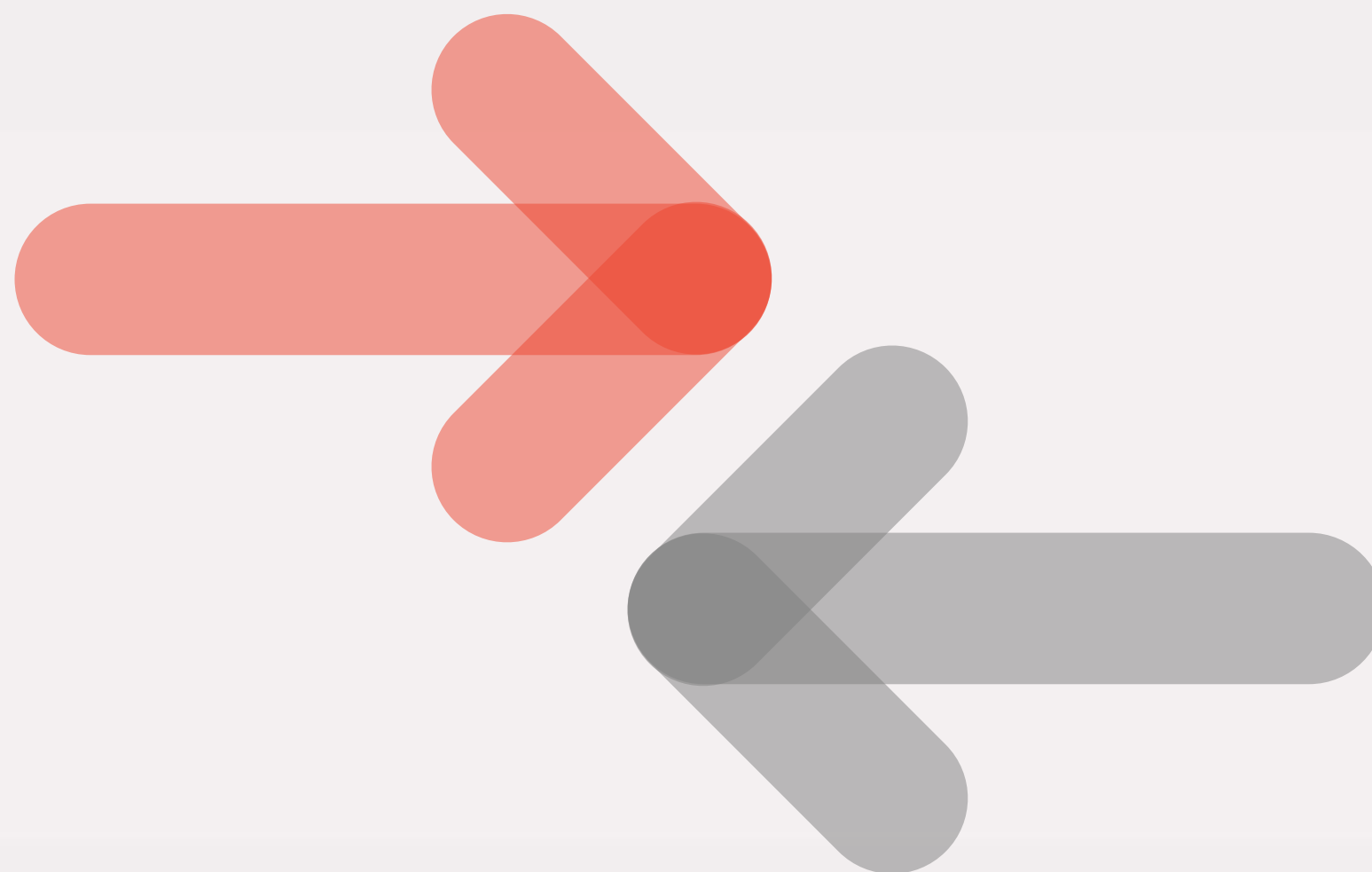


# Архитектура как стихия

Обуздываем энтропию проекта


# Представление

**Анна**  
Мелехова,  
[ann.melekhova@gmail.com](mailto:ann.melekhova@gmail.com)



**Более 15 лет в  
разработке**  
Parallels, Virtuozzo, Yandex

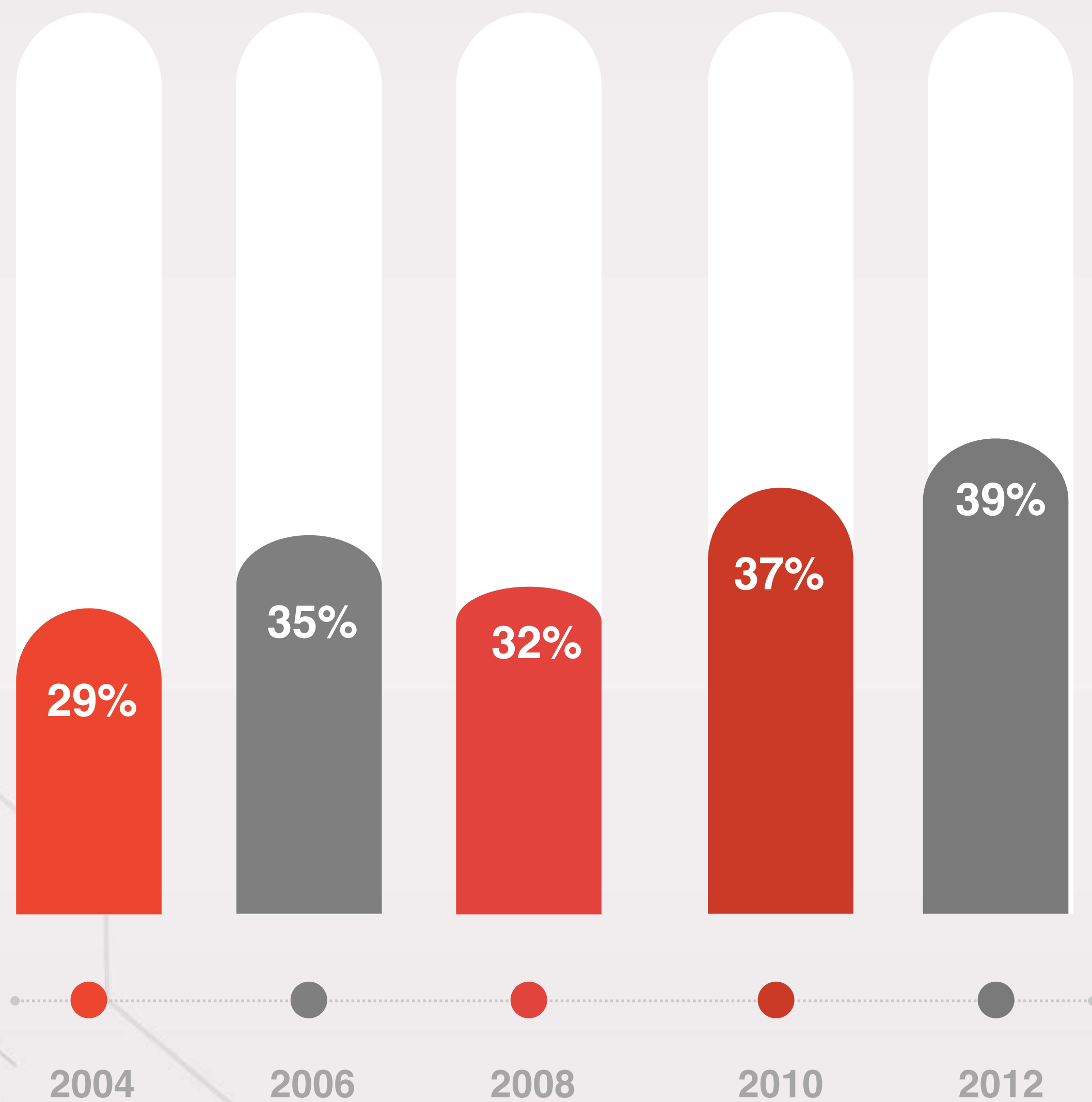




**Не бывает отсутствия  
архитектуры. Бывает архитектура  
спонтанно-неумышленная**



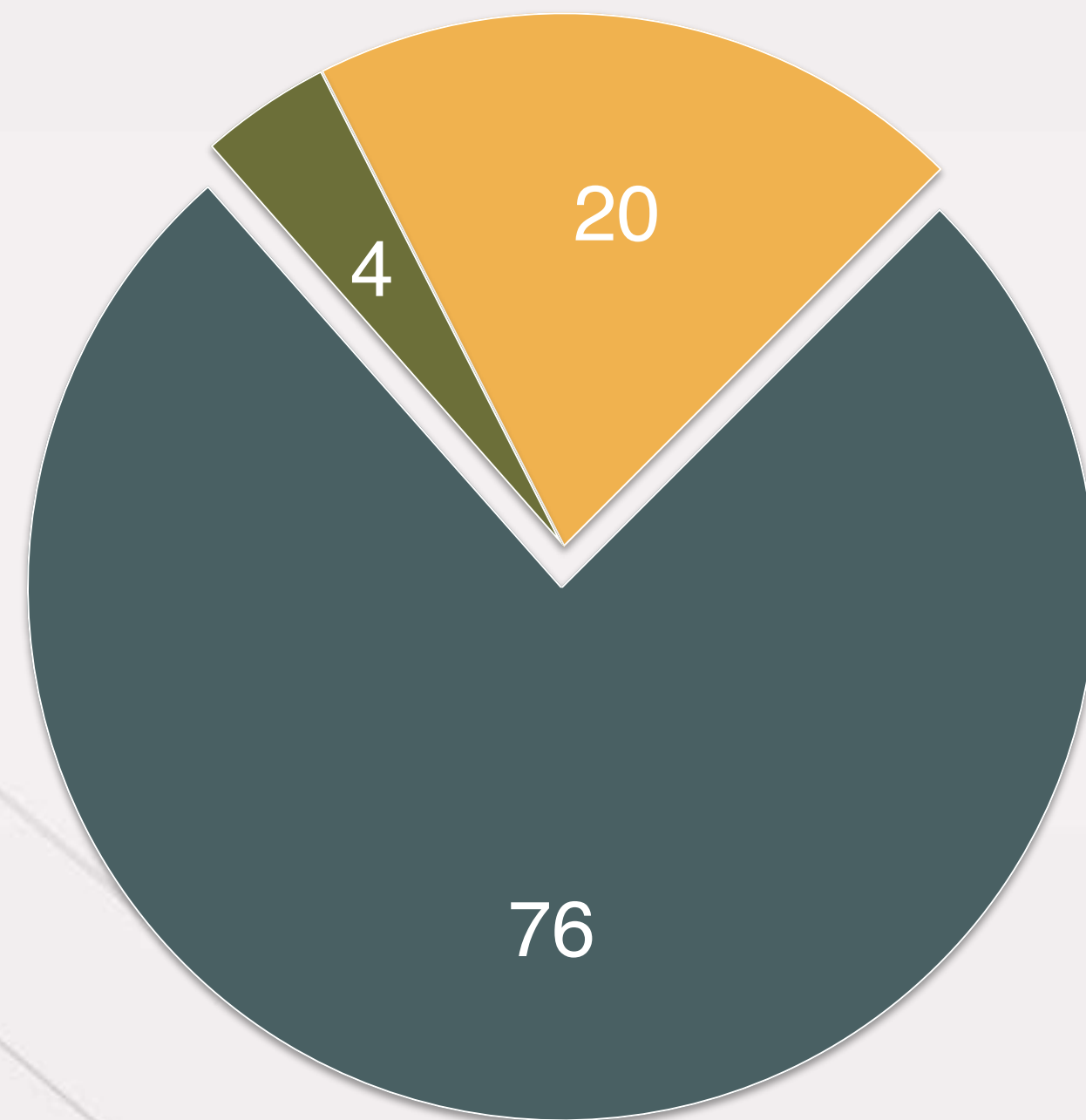
# Процент успешных проектов



(c) CHAOS Report  
The Standish Group  
International, Inc

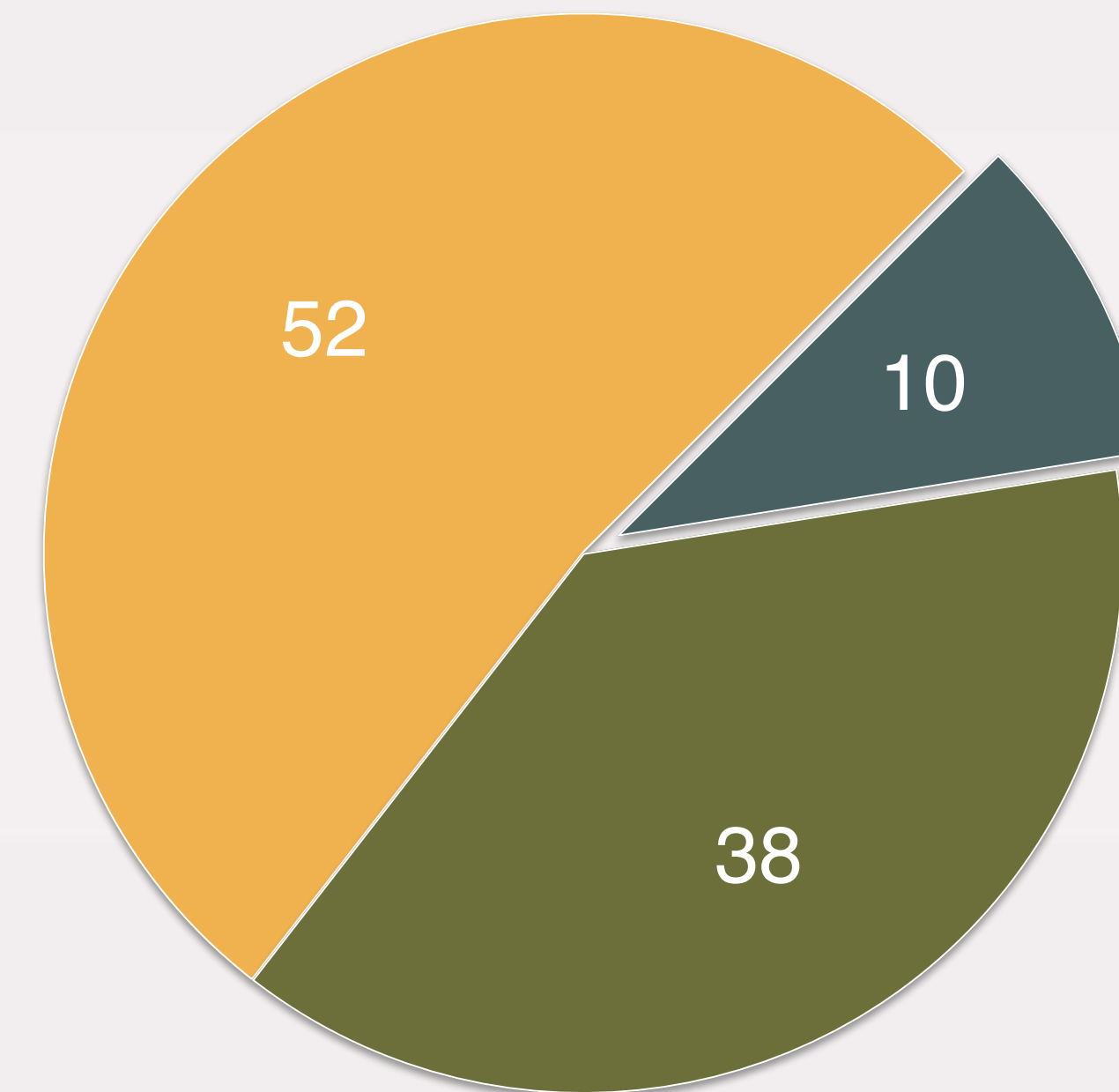
# Сложность проекта и размер

Небольшие проекты (<1млн\$)



- Успешные
- Неудачные
- Затянувшиеся

Большие проекты (>10млн\$)



- Успешные
- Неудачные
- Затянувшиеся

## Проблематизируем

Одним из основных источников неудач IT проектов является сложность

IT complexity crisis

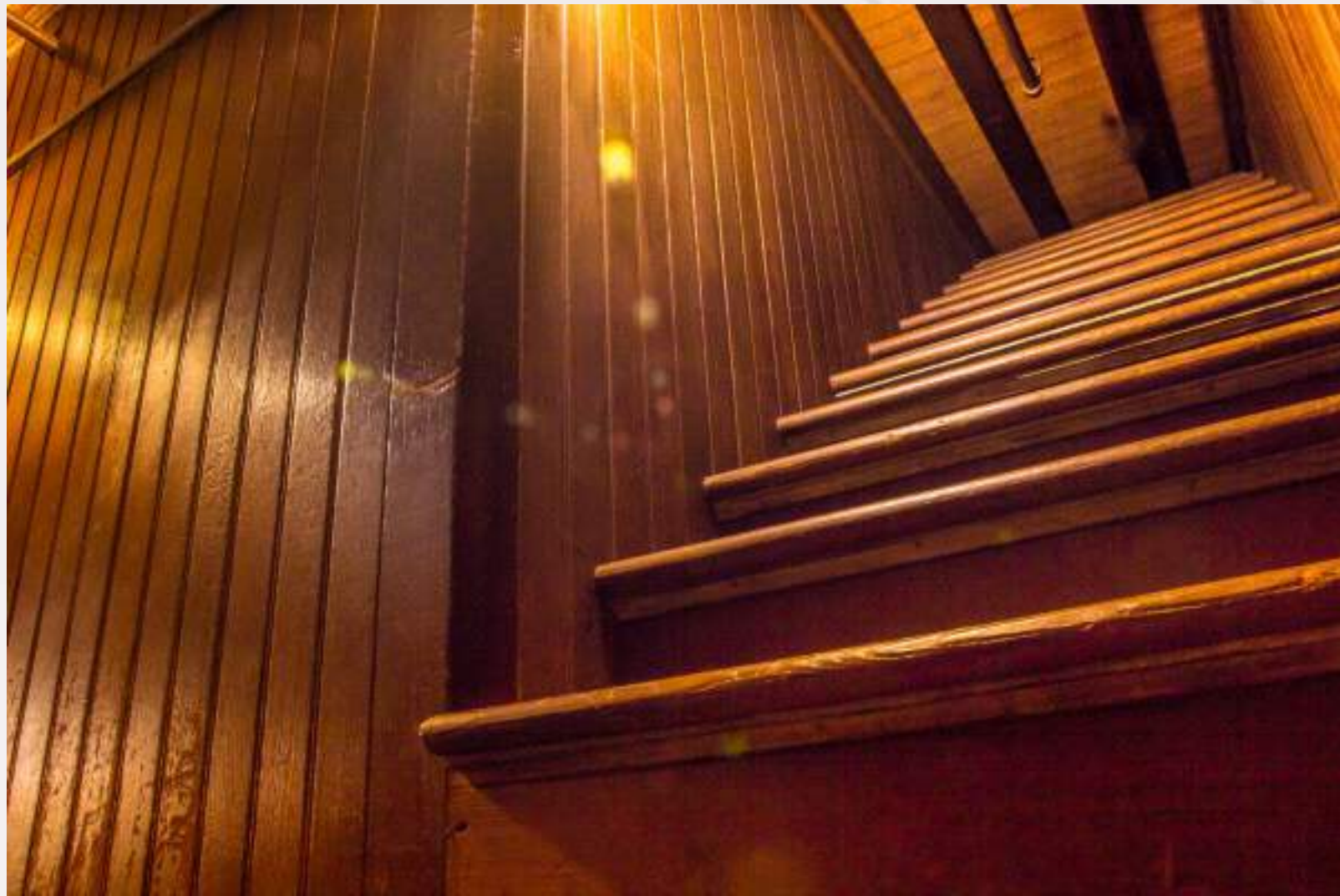
Проектам в IT нужно уменьшать сложность, а наращивать управление

Gartner











## Теорема сложения энтропий:

при объединении независимых систем  
их энтропии складываются

Энтропия может интерпретироваться как мера неопределённости (неупорядоченности) некоторой системы, например, какого-либо опыта (испытания), который может иметь разные исходы, а значит, и количество информации. Таким образом, другой интерпретацией энтропии является информационная ёмкость системы.



Кажется уже плохо, но докинем

При увеличении сложности проблемы на 25%, сложность решения возрастает на 100%

Glass' Law





**Чем больше система, тем выше ее энтропия, тем больше шанс неудачи проекта**



## Работа со сложностью

**Простой (simple)** = структура понятна

**Сложный (complicated)** = структура неясна

**Упорядоченный (ordered)** = поведение полностью предсказуемо

**Сложный/составной? (complex)** = поведение частично предсказуемо

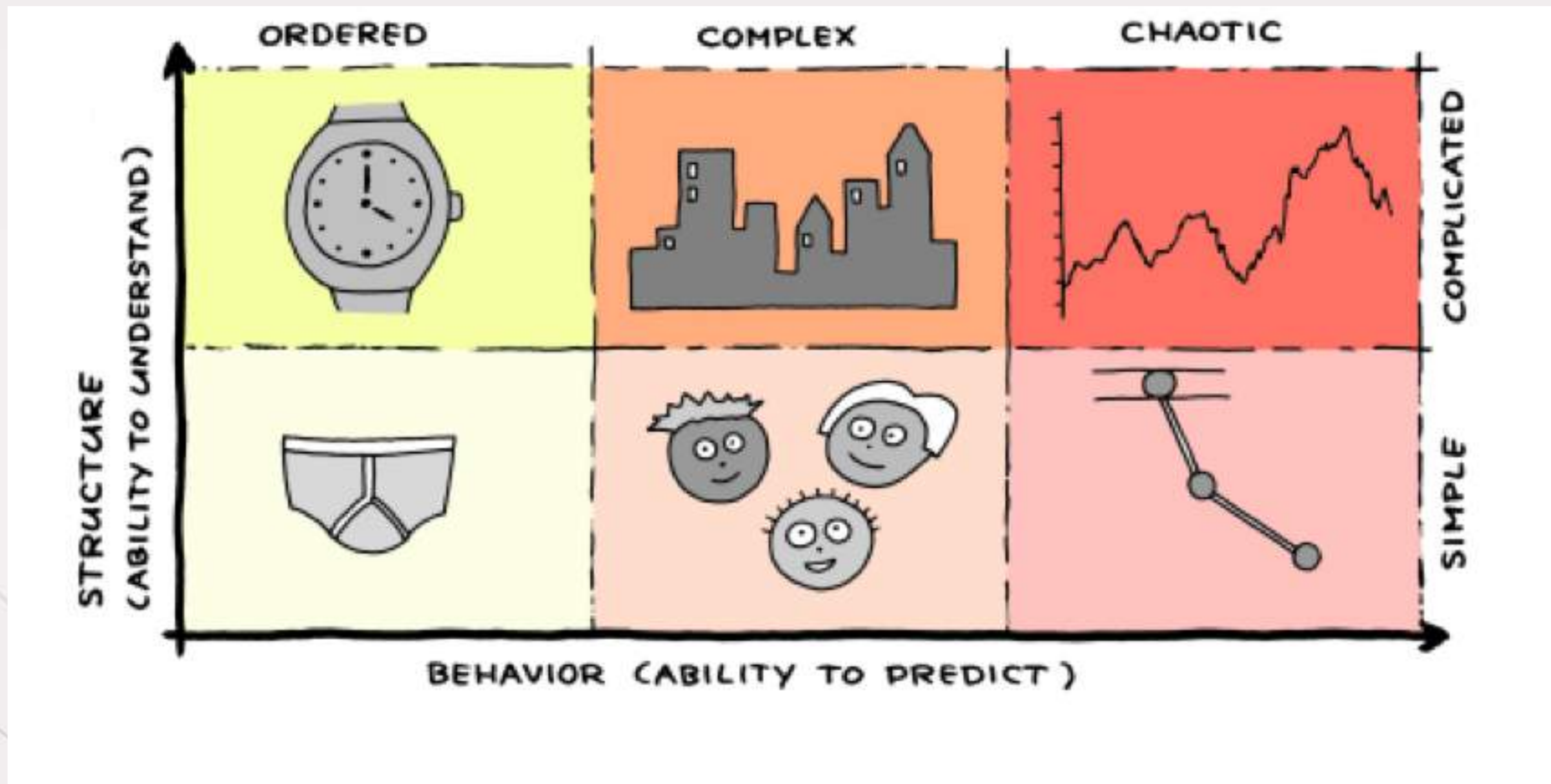
**Хаотичный (chaotic)** = поведение имеет высокую неопределенность

**Упрощение (simplification)** = делаем что-то более простым для понимания

**Уплотнение (linearization)** = делаем что-то более предсказуемым



# Работа со «сложностью»(2)



Jurgen Apello

Упрощение - это попытка сделать что-то более понятным.  
Уплотнение (линеризация) - это попытка сделать систему более предсказуемой



## Работа со «сложностью»(3)

Раньше я умел объяснять  
сложные вещи простыми словами  
Потом я узнал как  
сложные вещи  
на самом деле  
устроены.



Atkritka.com

Беспальчук, Custis



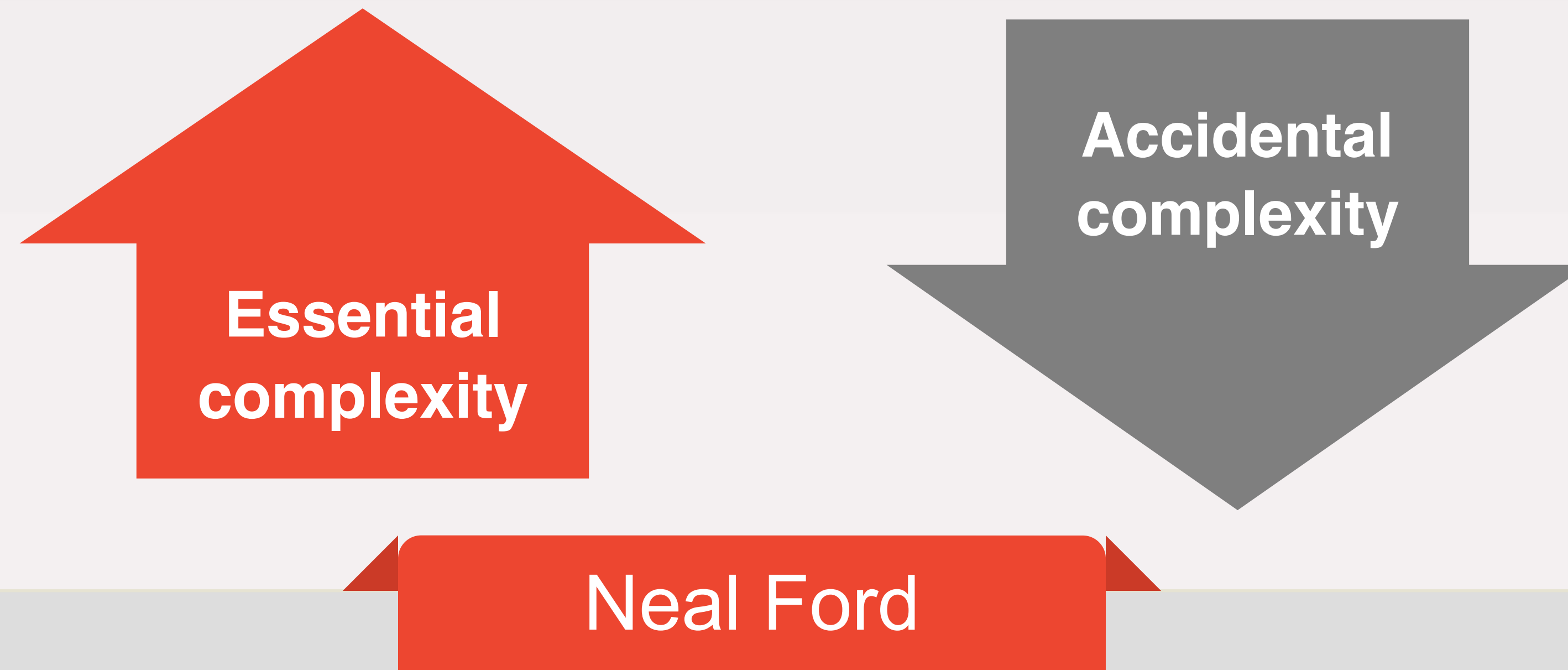
## Работа со «сложностью»(4)

Простой хорошо структурированный продукт может демонстрировать очень сложное (complex) поведение, в то время как сложной (complicated) путанный продукт может вести себя упорядоченно и полностью предсказуемо

Jurgen Apello



## Работа со «сложностью»(5)



Это прямая обязанность архитектора - решать проблемы, проистекающие от essential complexity без добавления accidental complexity

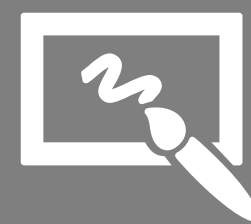


# Борьба со «сложностью»



## БРИТВА ОККАМА

- Из всех систем, реализующих функциональность, выбрать самую простую



## CONCEPTUAL INTEGRITY

Единообразии архитектурных тактик и паттернов



## МОДУЛЬНОСТЬ

Разбиение по модулям как искусство

## Аа

## ДОКУМЕНТАЦИЯ

Четкая и понятная архитектурная документация, внедренная во все стадии процесса, доступная всем



А если ничего не делать?



<http://www.laputan.org/mud/>

**Большой  
шмоток  
грязи  
(Big Ball  
of Mud)**



A dramatic photograph of a stormy sea. The sky is filled with dark, heavy clouds, and several bright, jagged lightning bolts are visible, striking down. The water is dark blue and turbulent, with white foam from the waves. The overall mood is intense and powerful.

**Conceptual integrity? Понятная архитектурная документация?**



# Conceptual integrity





## Conceptual integrity

Эволюция вновь и вновь немного по-разному использует одни и те же наборы генов <..> Эволюция не делает ничего нового на пустом месте... <...> Она работает с тем, что уже имеется, преобразуя ту или иную систему для выполнения новой функции или соединяя несколько систем, чтобы получить новую, более сложную

Франсуа Жакоб



# Conceptual integrity

Ныне я убежден более, чем когда-либо. Conceptual integrity - это центральная характеристика для оценки качества продукта.



Фред Брукс



# Архитектурная документация



## EXECUTIVE SUMMARY

Кратко для самых занятых



## ОПИСАНИЕ ПРОЕКТА

В чем суть запланированного



## ROADMAP

И когда планируется



## ARCH DRIVERS

Quality attributes, requirements, constraints



## КОНТЕКСТ

Система/продукт не в вакууме



## ПРИНЯТЫЕ АРХ РЕШЕНИЯ

С отсылками на контекст и architectural drivers



## VIEWS

Картинки для разных нужд



## ИНТЕРФЕЙСЫ

Их описание со всеми ограничениями



# Принятые архитектурные решения

- ✓ Проблема
- ✓ Решение
- ✓ Обоснование решения
- ✓ Рассмотренные альтернативные варианты
- ✓ Причины отказа от альтернативных вариантов
- ✓ Контекст принятого решения (стадия проекта, откуда исходила постановка вопроса)
- ✓ Состав участников (с кем обсуждали)

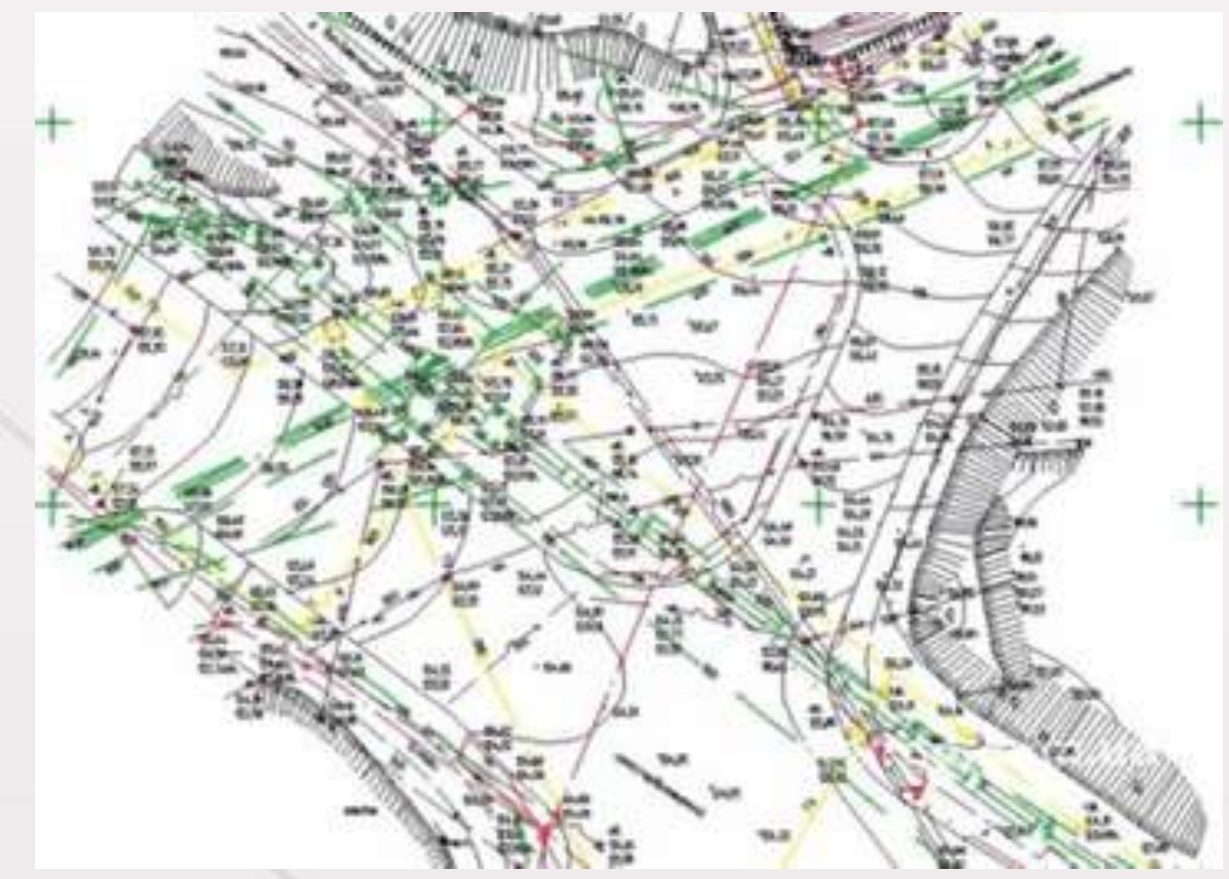
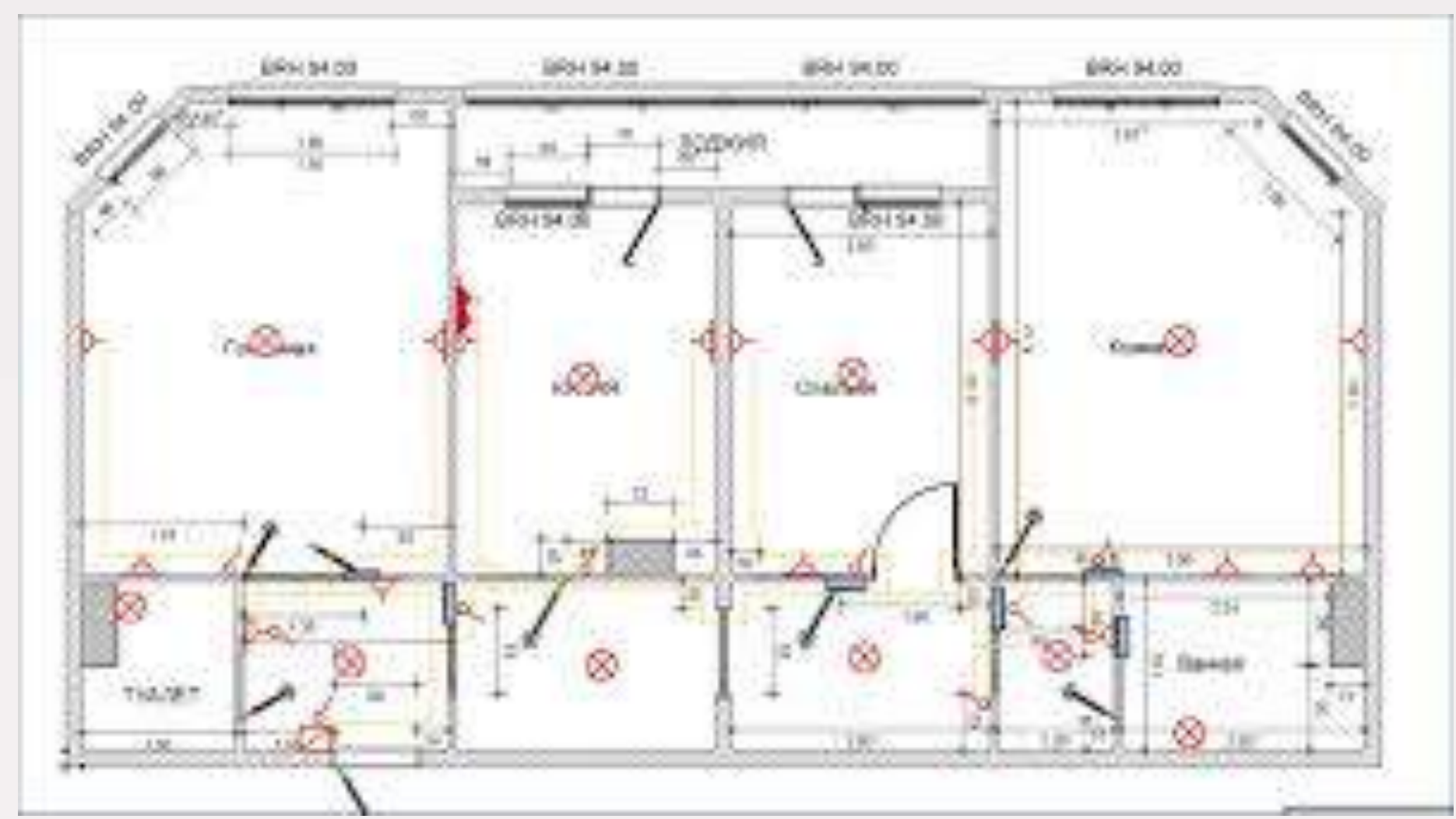


# Architectural drivers

- ✓ Функциональные требования (что делает система)
- ✓ Нефункциональные требования (как), aka quality attributes:
  - ✓ Производительность, отказоустойчивость, масштабируемость
  - ✓ Тестируемость, способность к отладке, модифицируемость, переносимость
  - ✓ и пр
- ✓ Constraints:
  - ✓ От бизнеса (время и стоимость разработки, команда и ее экспертиза, продуктовая линейка)
  - ✓ От разработки («несущие стены» проекта, языки и framework-и, legacy системы и их поддержка)



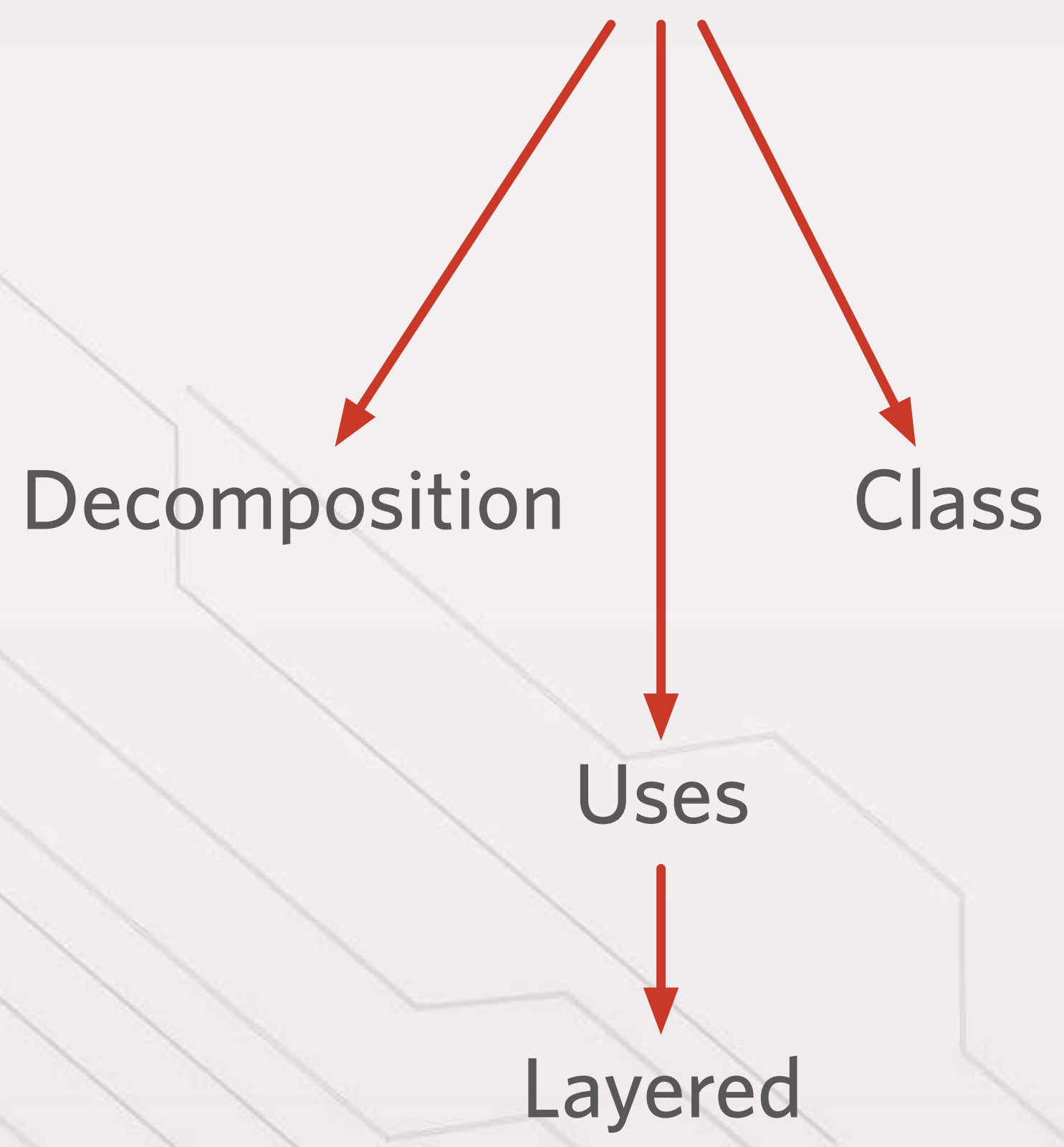
# Виды (views)



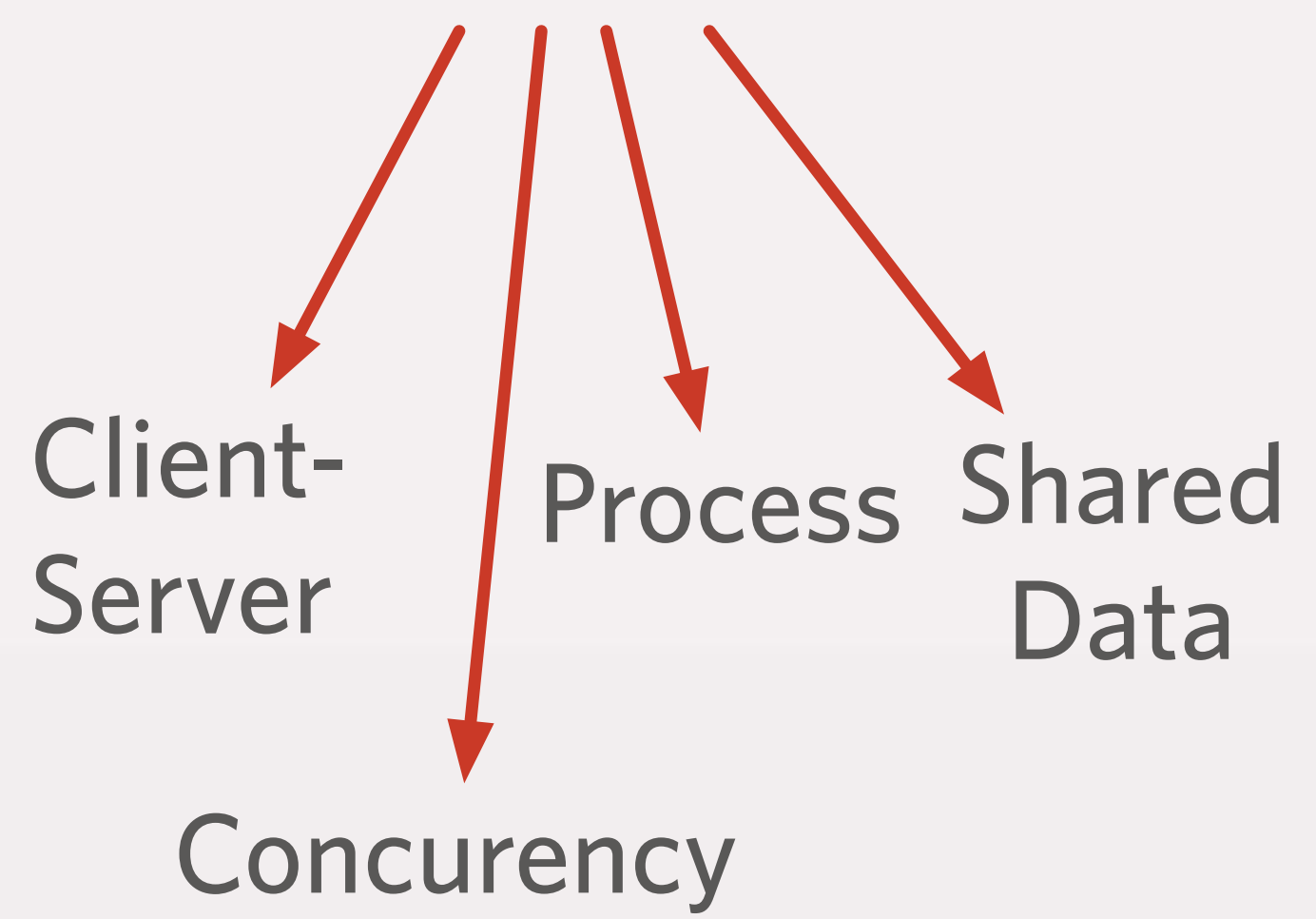


# Views: SEI (software engineering institute, CMU)

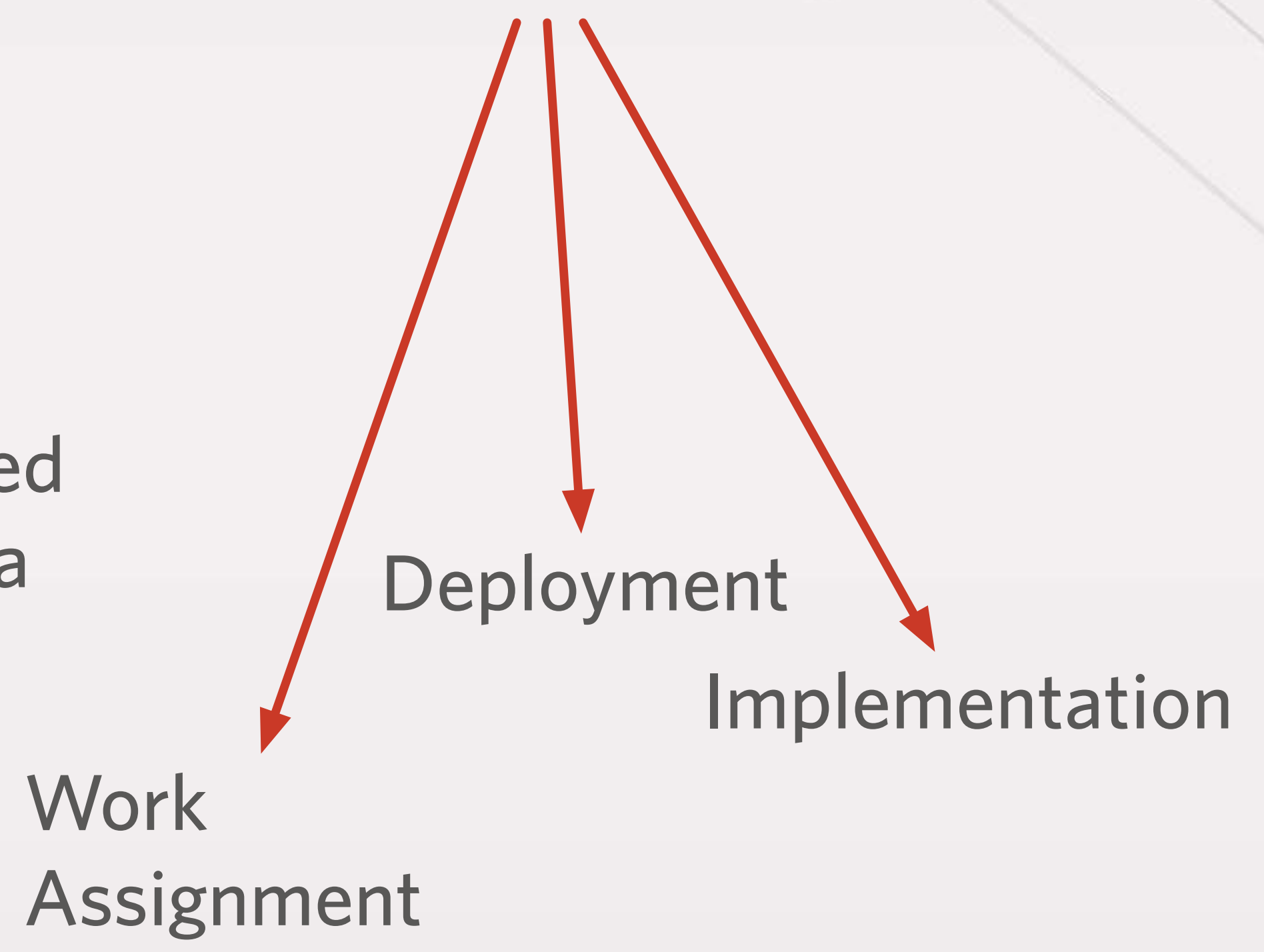
## Module



## Component-and-Connector

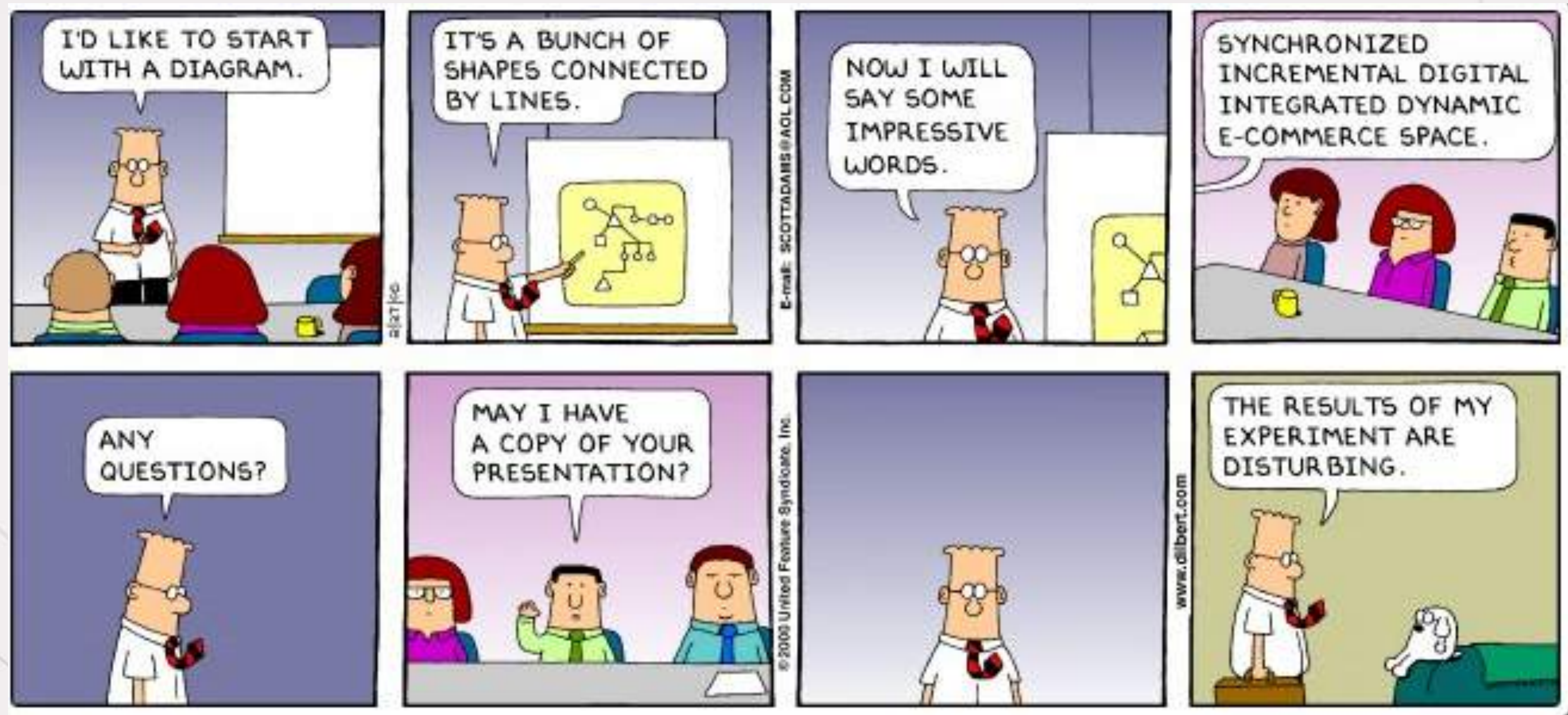


## Allocation





# UML



(c) United Features Syndicate, Inc



An aerial photograph of a mountain range, possibly the Himalayas, with a strong blue color cast. The mountains are rugged and layered, receding into the distance. The lighting is soft, highlighting the textures of the rock and the shadows in the valleys.

**Звучит здорово, но достаточно ресурсоемко. Как обосновать?**



## Проблемы с «внедрением» архитектурных процессов

- ✓ Менеджмент не воспринимает разработку архитектуры как серьезную инженерную деятельность
- ✓ На разработку архитектуры не выделяется ресурсов
- ✓ У архитектора нет career path; роль архитектора туманна для менеджмента
- ✓ Разработанная архитектурная документация не используется
- ✓ Принятые архитектурные решения не внедряются

(c) Anthony Lattanze,  
Infusing Architectural Thinking into Organizations



## Кредо архитектора

Каждое архитектурное решение в равной степени  
оказывается и техническим и экономическим решением

Anthony Lattanze



# Зачем нужно проектировать архитектуру (1)

1. Архитектура ограничивает или делает возможным основные нефункциональные характеристики системы, служит обоснованием решений и базой для предсказаний свойств системы
2. Архитектурная документация увеличивает понимание между заинтересованными лицами
3. Архитектура продукта задает структуру организации и наоборот.

Carnegie-Mellon: Len Bass, Paul Clements, Rick Kazman



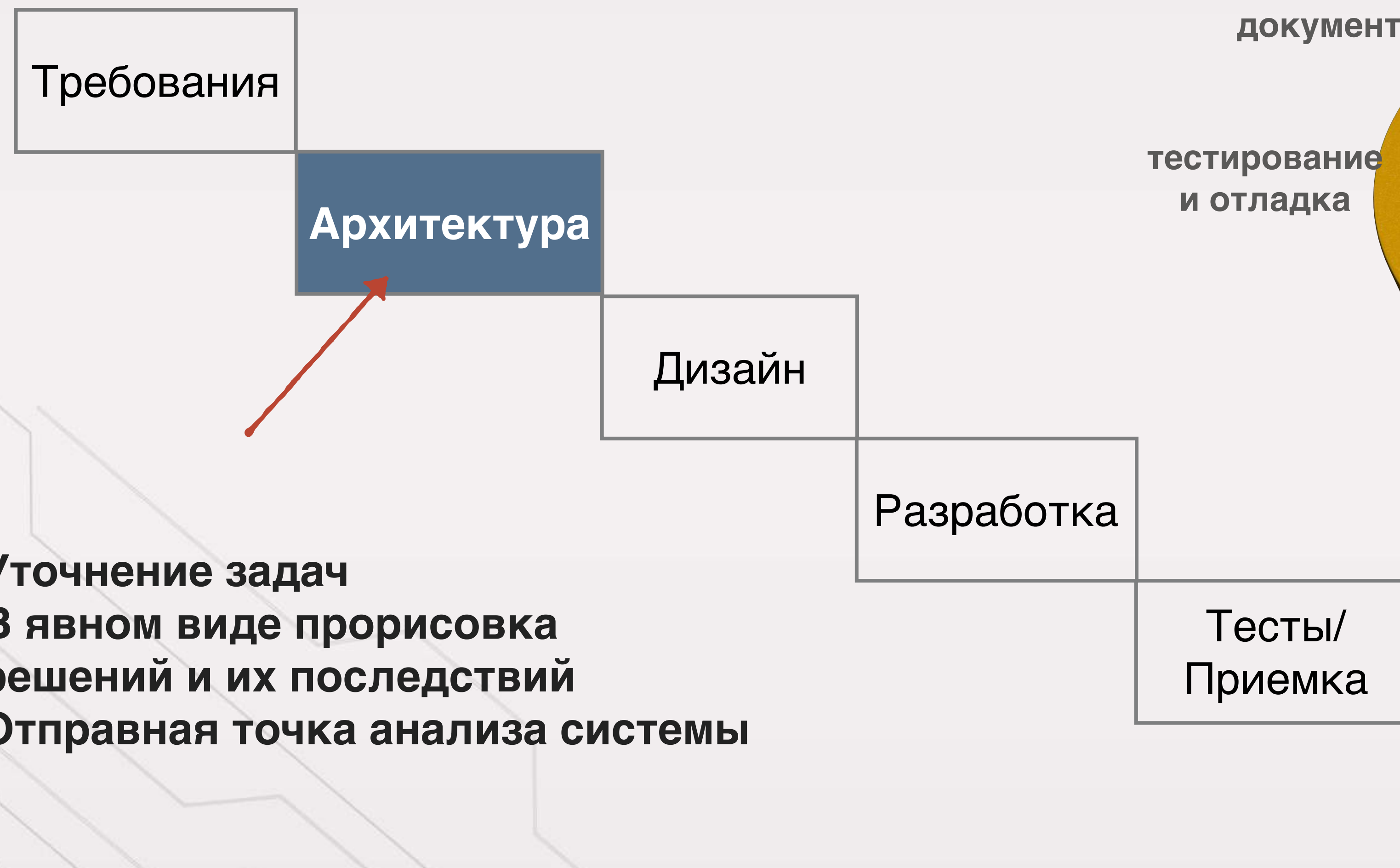
## Зачем нужно проектировать архитектуру (2)

4. Архитектура может стать базисом для итеративного прототипирования, она же основной артефакт для аргументирования сроков и стоимостей
5. Архитектурные процессы сосредоточены на построении системы в целом, а не на разработке отдельных компонентов
6. Архитектура уменьшает сложность системы

Carnegie-Mellon: Len Bass, Paul Clements, Rick Kazman



# Архитектура. Зачем?



**Уточнение задач**  
**В явном виде прорисовка решений и их последствий**  
**Отправная точка анализа системы**

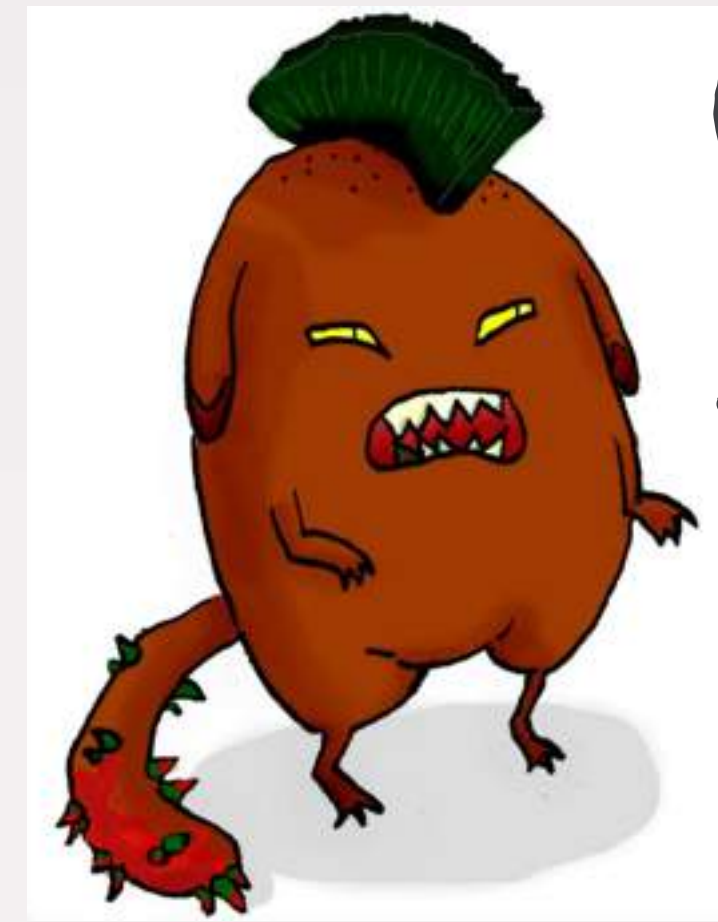
**Снижение стоимости поддержки**  
**прямо и косвенно**





# Роль архитектора

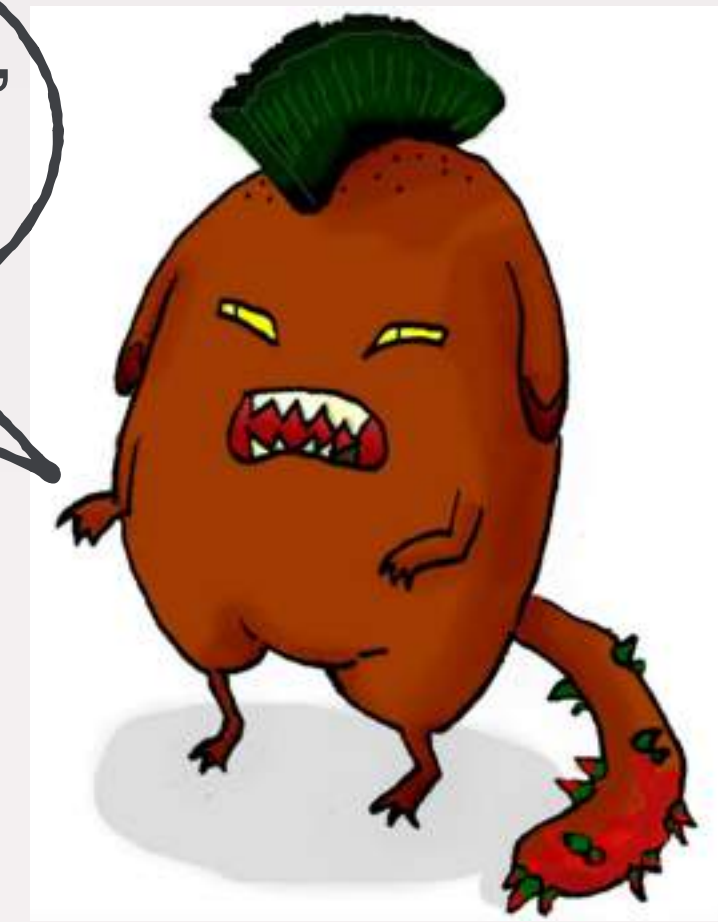
**Менеджмент**



низкая стоимость разработки

конкурентные фиши, WOW эффект, time to market

**Маркетинг**



modifiability

**Сопровождение**



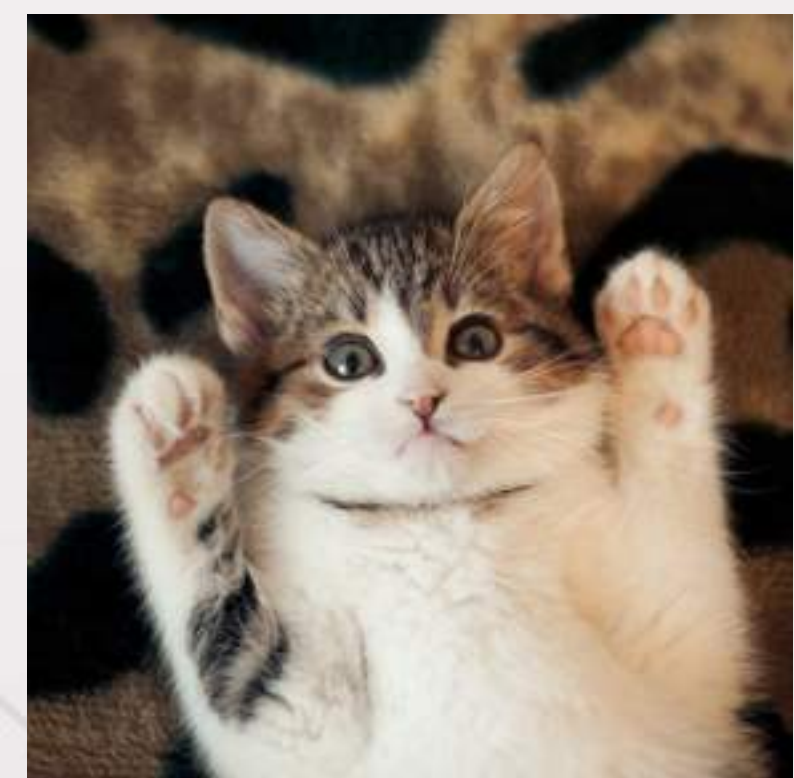
низкая цена, не слишком частые апдейты

**Конечные пользователи**

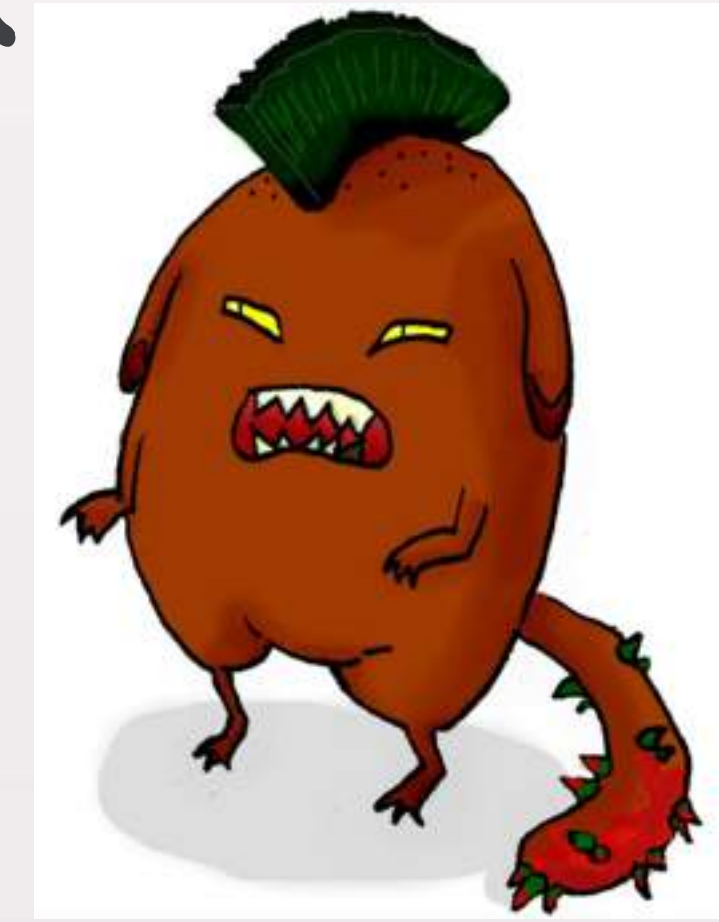


скорость, безопасность, удобство

**Архитектор**



**Покупатели**





## Роль архитектора

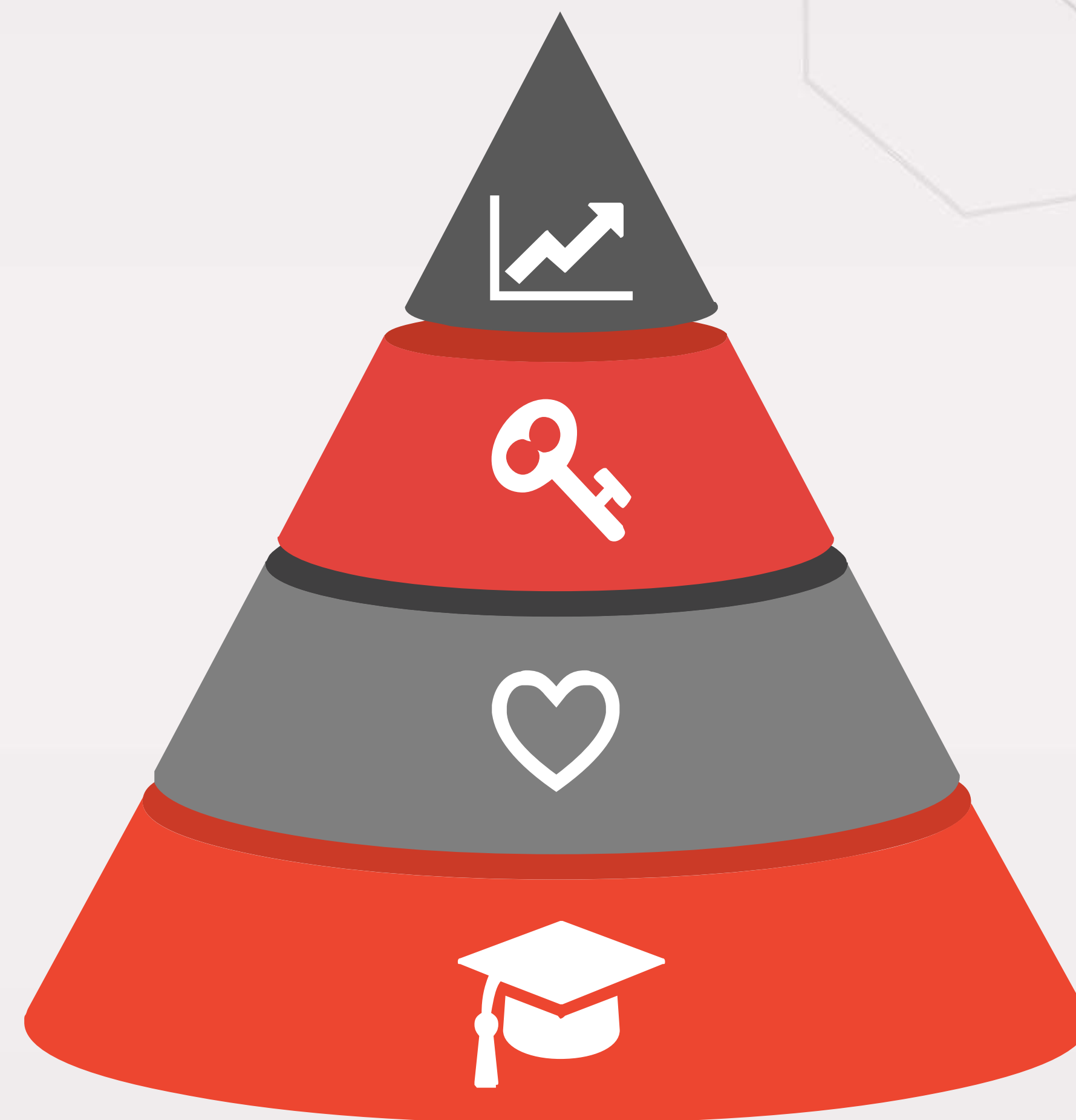
Архитектор - не тот человек, который озвучивает ограничения. Архитектор - это тот профессионал, который в условиях ограничений предлагает возможности

из разговоров

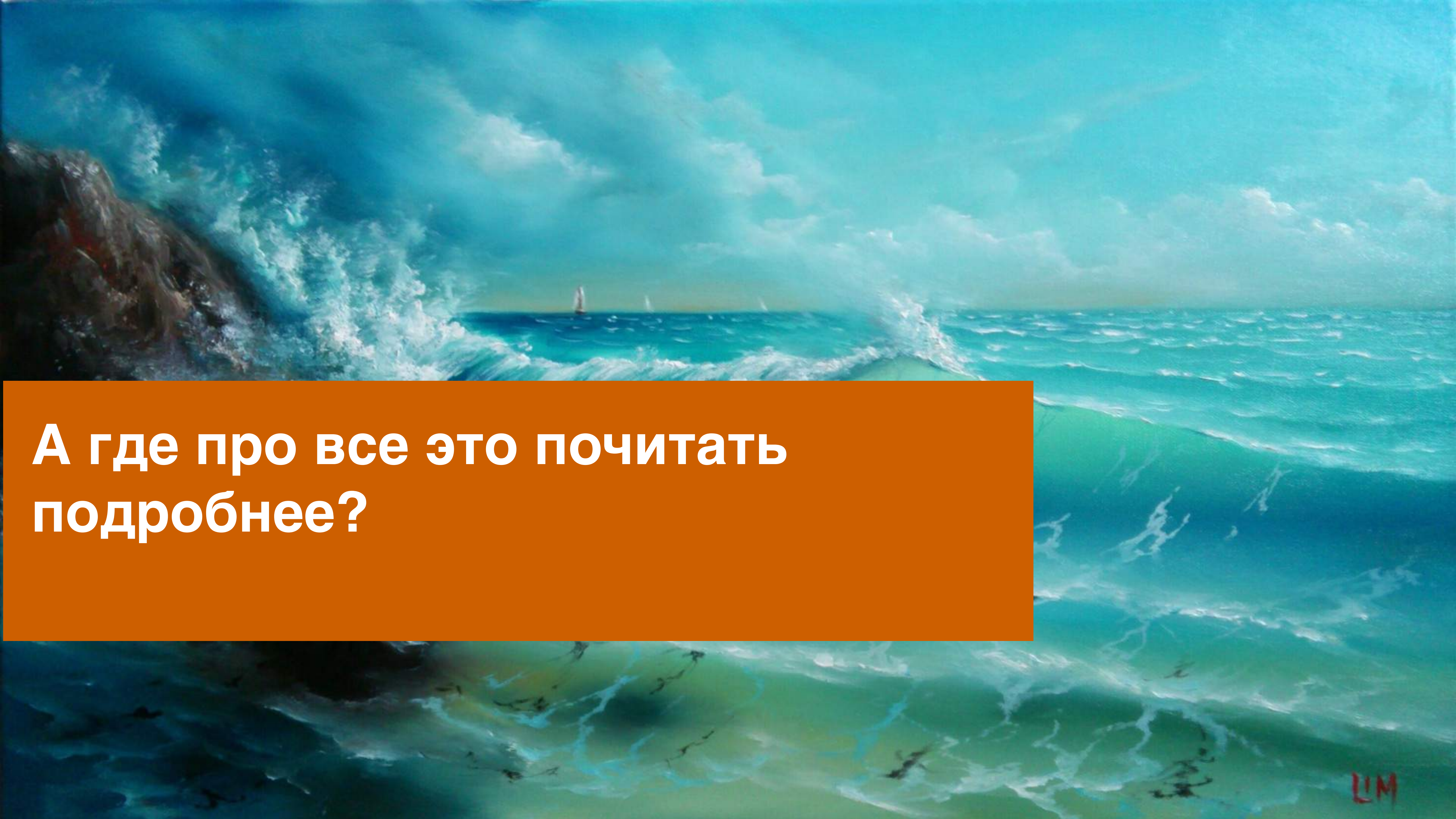


# Навыки архитектора

- ✓ **Деловые качества**  
Прагматизм; vision; понимание бизнес процессов; инновации
- ✓ **Личные качества**  
Страсть; быстрое переключение контекста; «незаметность»
- ✓ **Искусство построения отношений**  
Лидерство; великодушие; тактичность; коммуникативность; переговорческий навык
- ✓ **Техническая база**  
Безусловно необходима, но совершенно недостаточна







**А где про все это почитать  
подробнее?**





# **Software Engineering Institute**

## **Carnegie Mellon**



# Availability Tactics

Fault → Detect

→ Recover from

→ Prevent

→ Fault Masked or Repair Made

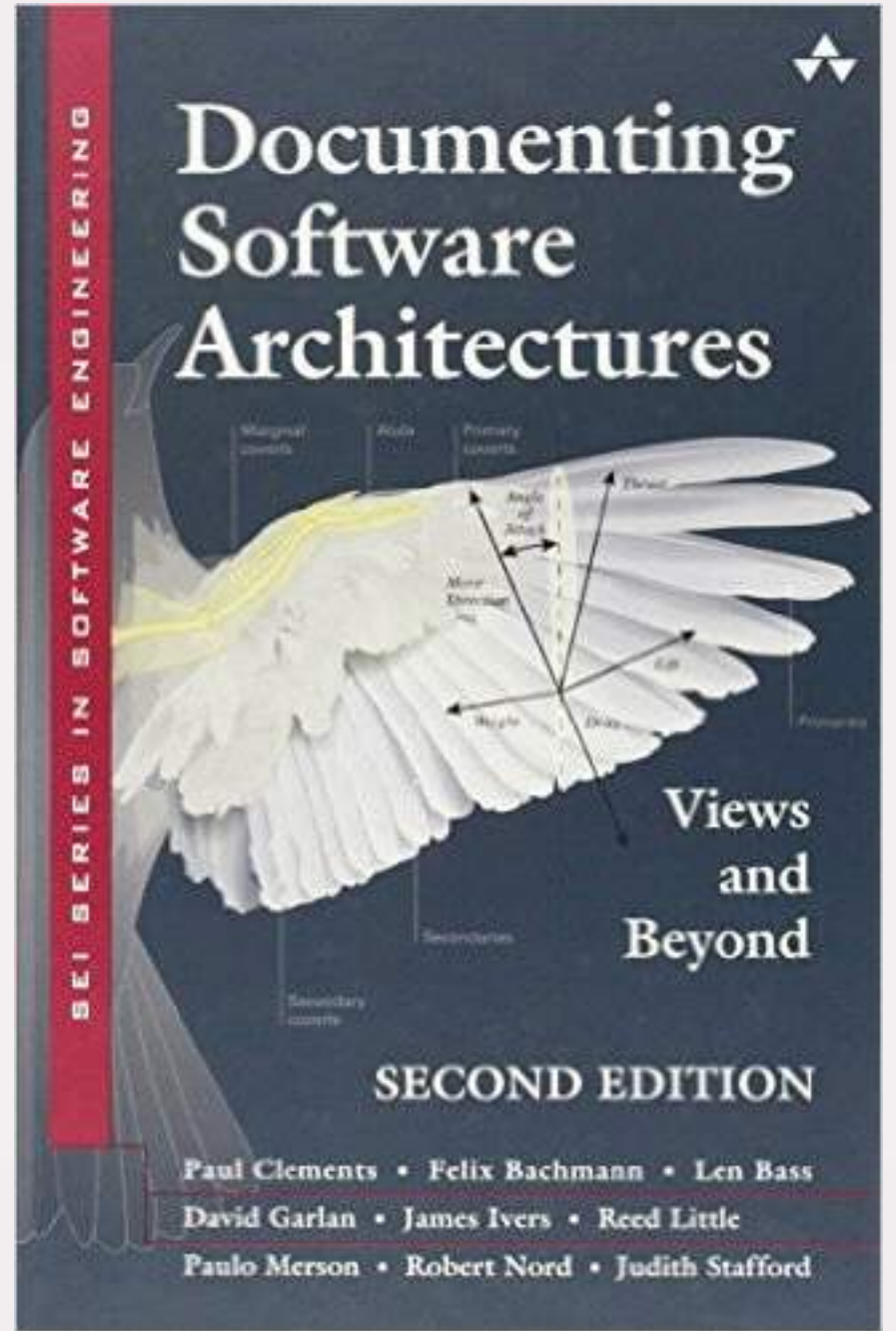
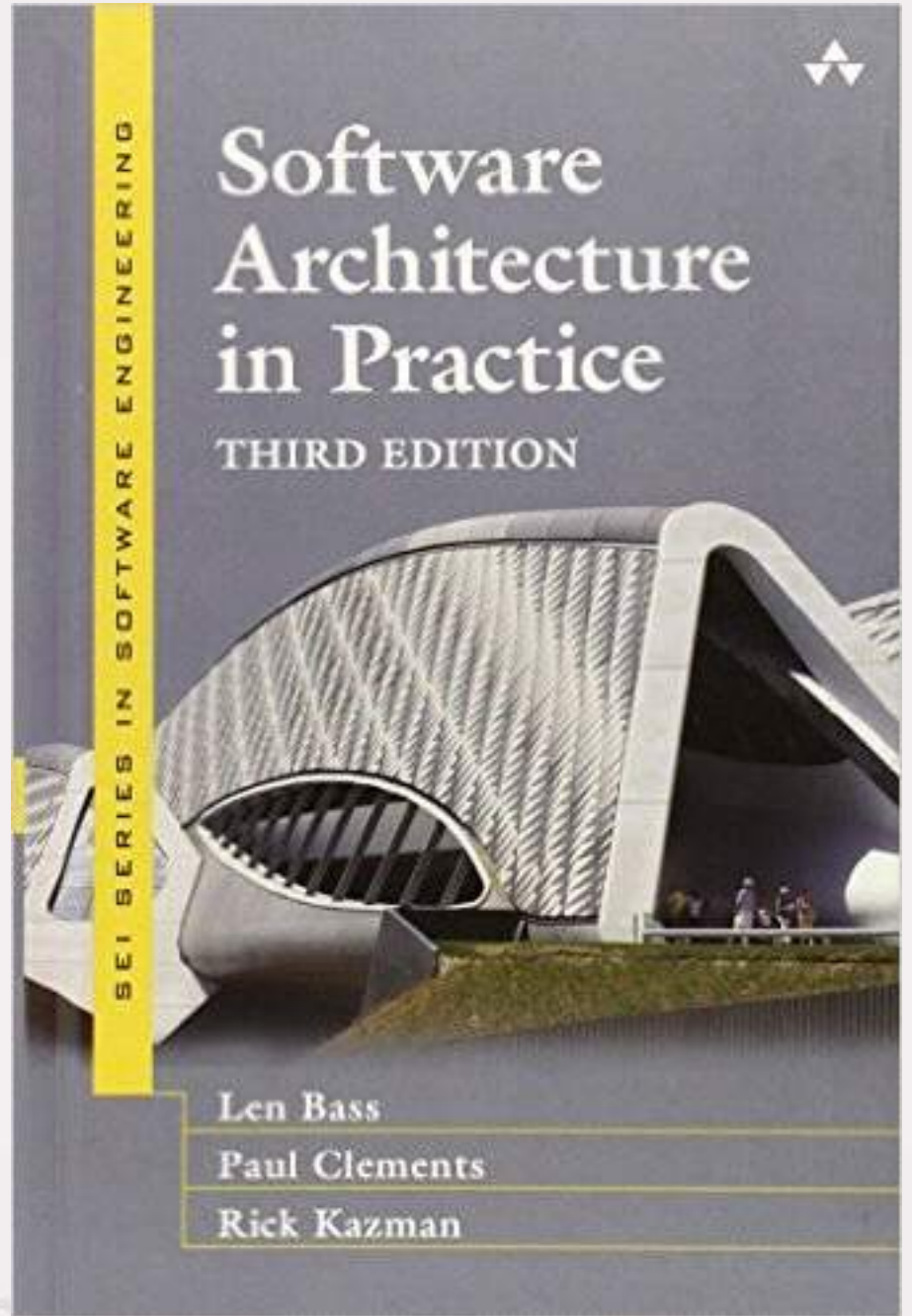
- Ping/Echo
- Monitor
- Heartbeat
- Timestamp
- Sanity Checking
- Condition Monitoring
- Voting
- Exception Detection

- Preparation
- Active Redundancy
  - Passive Redundancy
  - Spare
  - Exception Handling
  - Rollback
  - Software Upgrade
  - Retry
  - Ignore Faulty Behavior
  - Degradation

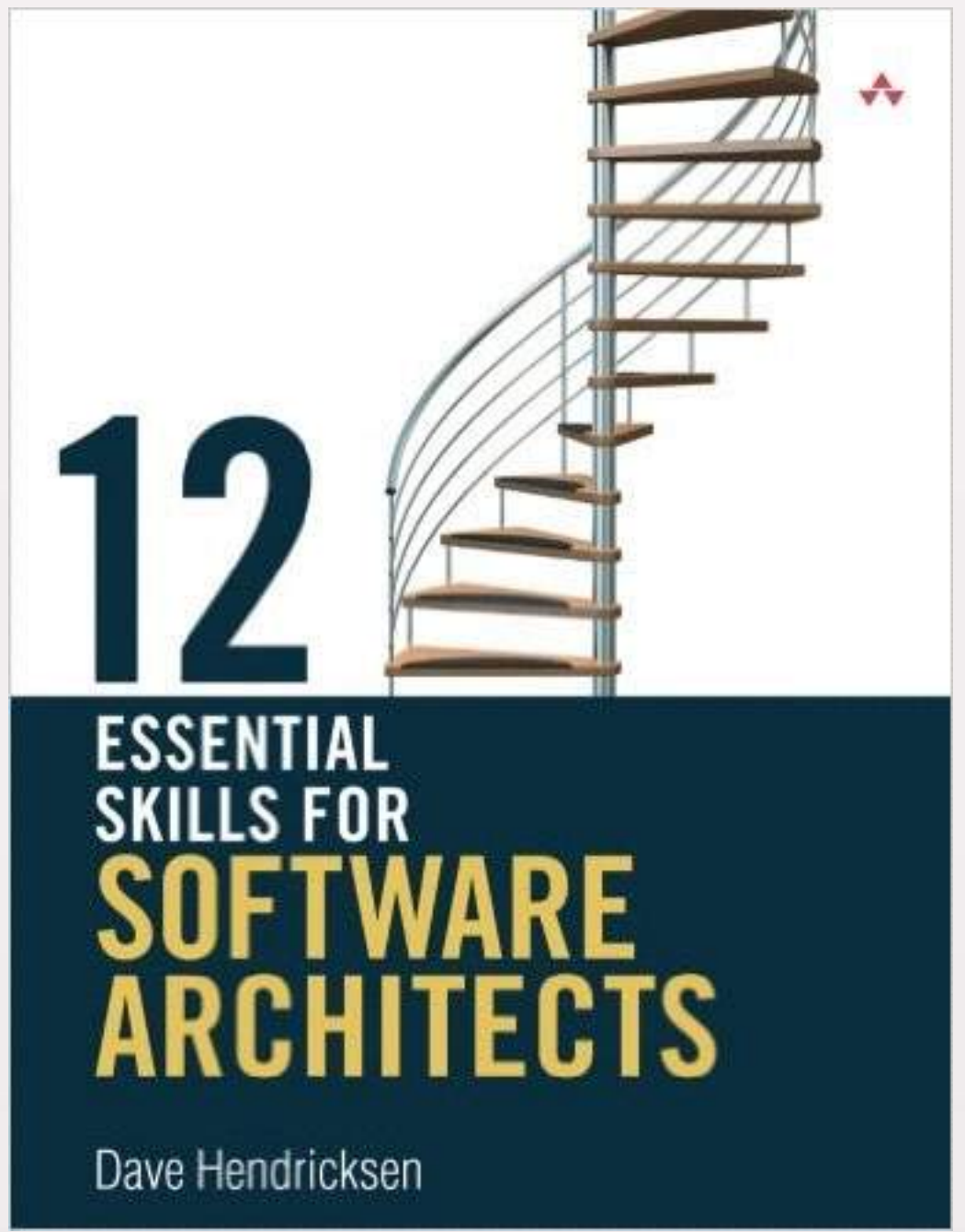
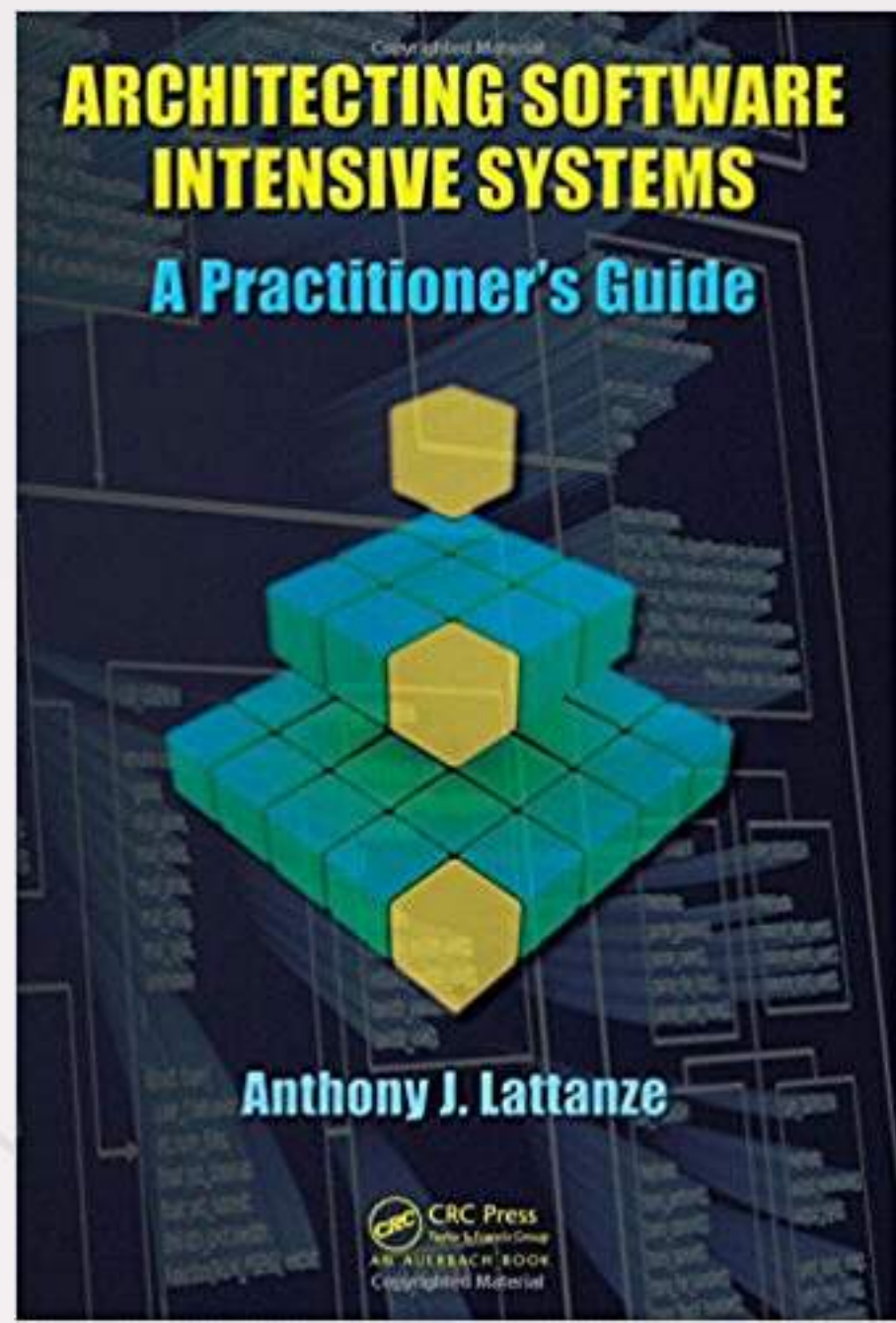
- Reintroduction
- Shadow State
  - Resynchronization
  - Escalating Restart
  - Non-Stop Forwarding

- Removal from Service
- Transactions
- Predictive Model
- Exceptional Prevention
- Increase Competence Set

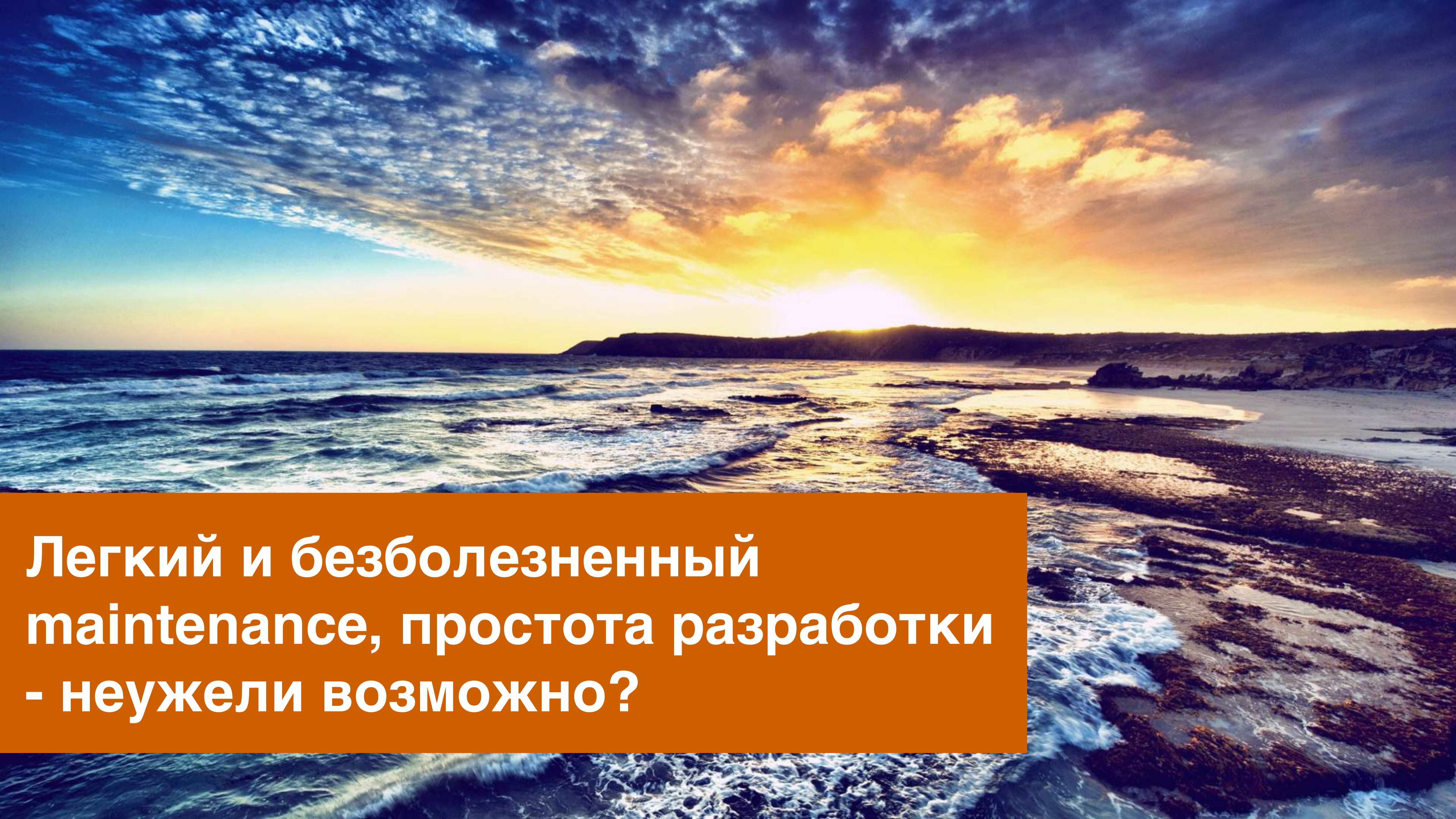












**Легкий и безболезненный  
maintenance, простота разработки  
- неужели возможно?**



# Завершая

✓ **Не бывает отсутствия архитектуры**  
неумышленная архитектура имеет неконтролируемый рост сложности и энтропии

✓ **Сложность**  
это не приговор. Со сложностью можно и нужно работать

✓ **Архитектура**  
это целый пласт знания: развивайтесь, узнавайте.



**Архитектор**  
обеспечивает взаимодействие разработки и бизнеса

**Введение архитектурных процессов**  
вызывает сопротивление системы, но...

**...НО ОНО ТОГО СТОИТ**  
ибо улучшает качество продукта и облегчает сопровождение системы





**Спасибо за внимание!**