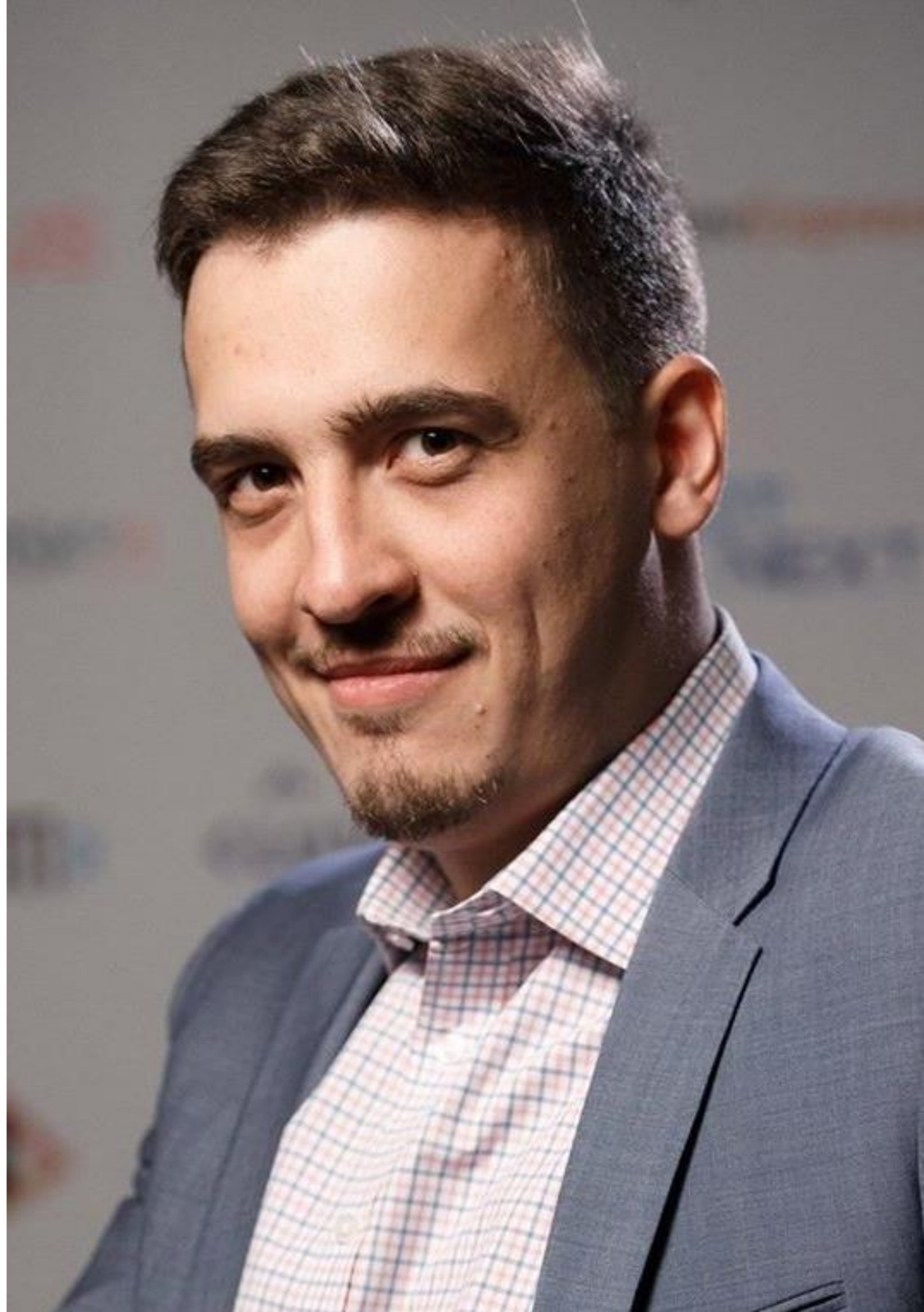
Архитектура, какая ещё архитектура?

Или как не угробить проект за пару ~~лет~~ месяцев

Михаил Дружинин Tech Train 2018

DataArt

- 12 лет в IT
- Множество "вскрытий" систем
- Неравнодушен к системам, которые живут долго



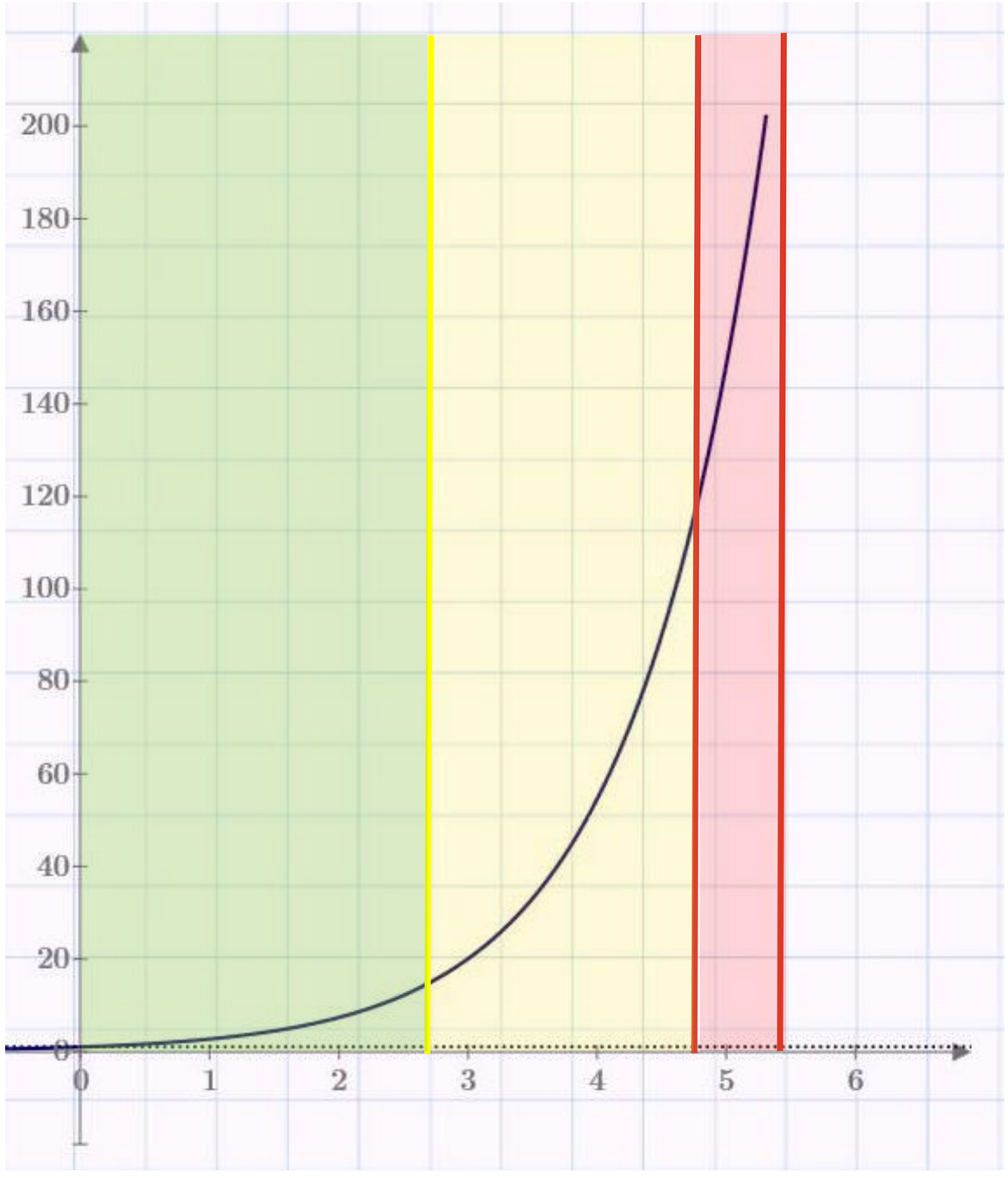
Проблема

- Этот код сложно поменять
- Архитектура нашей системы 'не очень'
- Всё нужно переписать

- Новые функции делаются слишком долго

Сложность

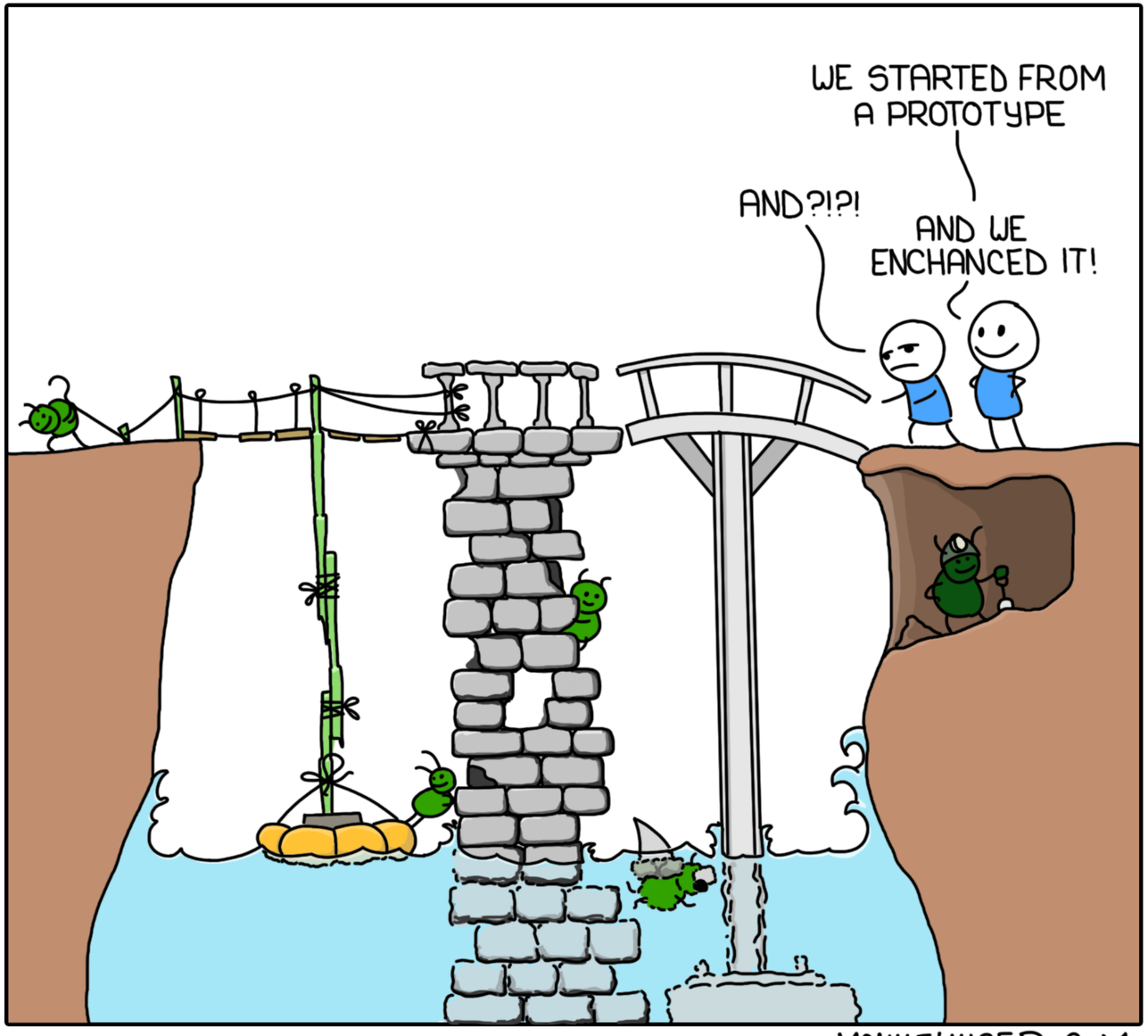
Внести изменение, не сломав чего-то становится не возможно


200
180
160
140
120
100
80
60
40
20
0
0
1
2
3
4
5
6

Почему растёт сложность системы

- Не поняли целей/задачи и сделали иначе
- Что-то захардкодили
- … потому что спешили

PRODUCTION READY



О чём будем говорить?

- Что мы регулярно упускаем из виду
- Что происходит с системой во времени
- Как можно избежать «стандартных ошибок»
- Как использовать архитектуру для контроля сложности

Хорошо делай – хорошо будет

Не увеличивай сложность

Какая архитектура у вашей системы?

Что такое архитектура?

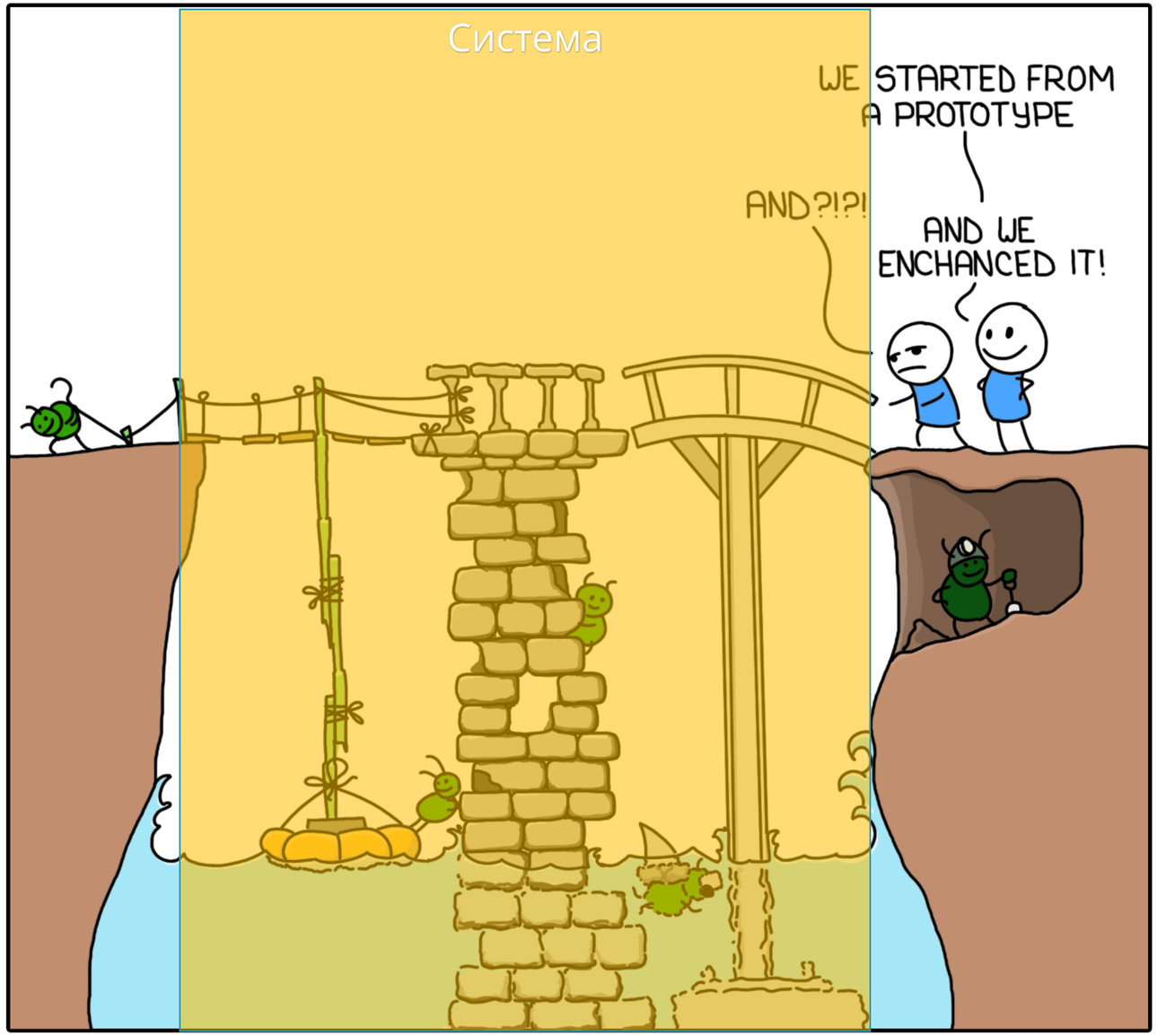
The important stuff (whatever that is)

Ralph Johnson (Gang of Four)

Вспомните свою систему...

Какая она, что в ней самое важное, почему...

PRODUCTION READY



Хрустальный шар не работает

Большинство ваших предположений **не совсем** верно

Искажения выбора

- Успешные решения в прошлом
- Имеющиеся навыки
- Предыдущая боль
- Решения общих задач

Система делается, чтобы что?

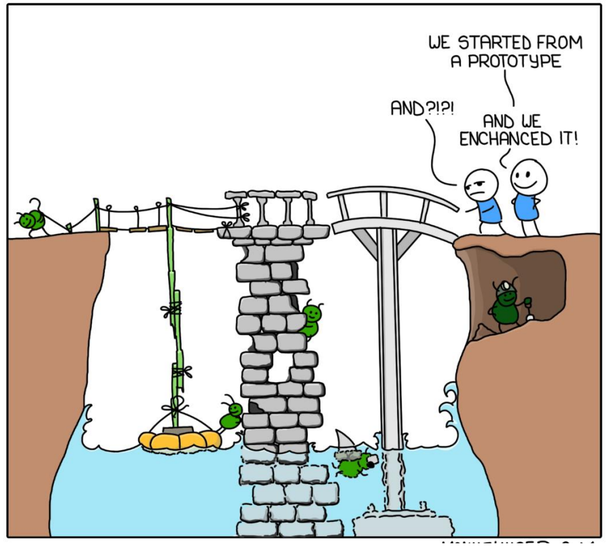
Знаете ли вы зачем делается ваша система?

Есть функциональные требования

- Как пользователь я хочу …. чтобы ….
- Переходить по мосту, чтобы попасть из точки А в точку Б

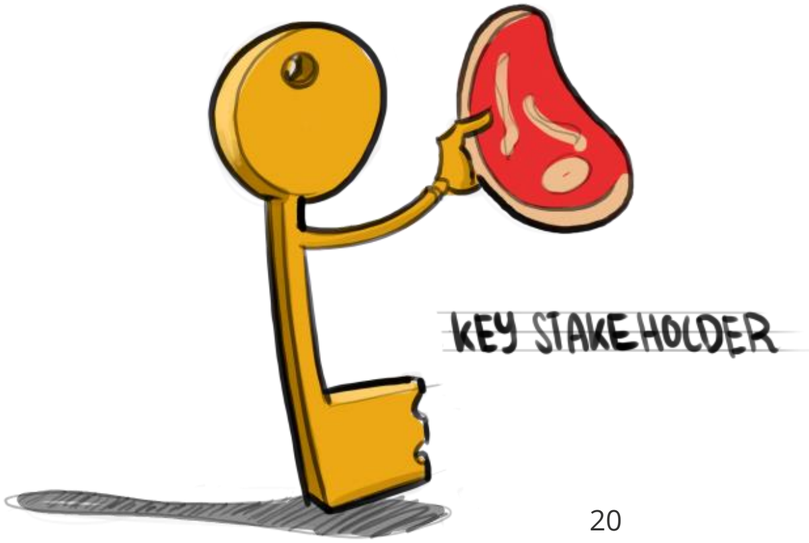
Цели и “беспокойства”

- Система нужна, чтобы перемещать объекты между точками A и B и зарабатывать за перемещение
- Мы планируем, что через систему будут перемещаться 10 млн. объектов в день к 2020 году
- Система должен поддерживать разные виды объектов для перемещения
- Недопустимо причинения вреда, в ходе перемещения



Заинтересованные лица

- Пользователи
- Заказчик
- Команда разработки
- Команда поддержки
- Кто-то чьи интересы система задевает



Нужно делать так, как нужно.
А как не нужно, делать не нужно!

(c) Винни-Пух

Какие могут быть цели

- Уменьшение стоимости владения
- Улучшение свойств системы или бизнес процессов
- Улучшить позицию компании на рынке

- Система **не может** быть целью

Хорошо ли система работает?

Каковы метрики вашей системы?

...abilities

- Accessibility
- Adaptability
- Auditability and control
- Availability (see service level agreement)
- Backup
- Capacity, current and forecast
- Certification
- Compliance
- Configuration management
- Cost, initial and Life-cycle cost
- Data integrity
- Data retention
- Dependency on other parties
- Deployment
- Development environment
- Disaster recovery
- Documentation
- Durability
- Efficiency (resource consumption for given load)
- Effectiveness (resulting performance in relation to effort)
- Emotional factors (like fun or absorbing or has "Wow! Factor")
- Environmental protection
- Escrow
- Exploitability
- Extensibility (adding features, and carry-forward of customizations at next major version upgrade)
- Failure management
- Fault tolerance (e.g. Operational System Monitoring, Measuring, and Management)
- Legal and licensing issues or patent-infringement-avoidability
- Interoperability
- Maintainability (e.g. Mean Time To Repair - MTTR)
- Management
- Modifiability
- Network topology
- Open source
- Operability
- Performance / response time (performance engineering)
- Platform compatibility
- Privacy (compliance to privacy laws)
- Portability
- Quality (e.g. faults discovered, faults delivered, fault removal efficacy)
- Readability
- Reliability (e.g. Mean Time Between/To Failures - MTBF/MTTF)
- Reporting
- Resilience
- Resource constraints (processor speed, memory, disk space, network bandwidth, etc.)
- Response time
- Reusability
- Robustness
- Safety or Factor of safety
- Scalability (horizontal, vertical)
- Security (cyber and physical)
- Software, tools, standards etc. Compatibility
- Stability
- Supportability
- Testability
- Throughput
- Transparency
- Usability (Human Factors) by target user community
- Integrability ability to integrate components

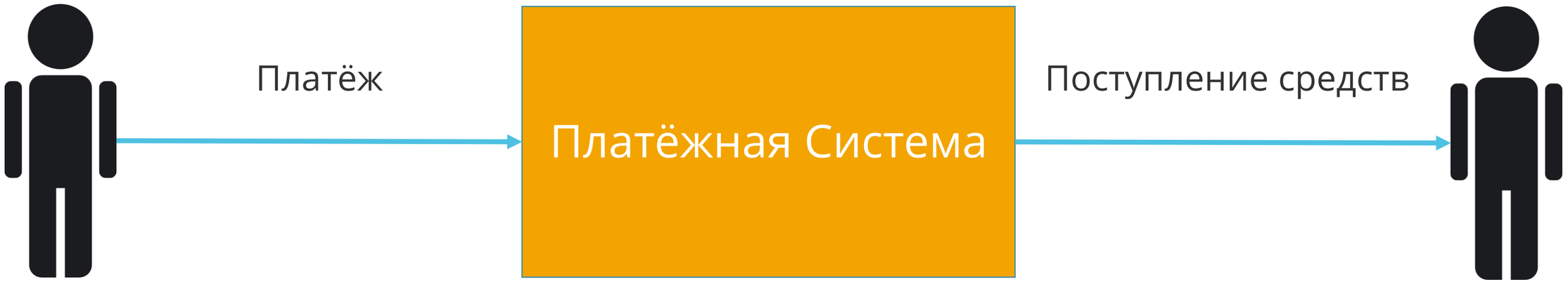
...abilities

Группы атрибутов качества

- Доступности и времени восстановления после сбоя
- Развития и ремонтопригодности
- Управляемости и поддерживаемости
- Безопасности и регуляторов
- Тестируемости
- Производительности и масштабируемости
- и т.д

Требований много

Учтя всё можно не получить места для решения


Платёж
Платёжная Система
Поступление средств

Проверки платежа vs скорость платежа

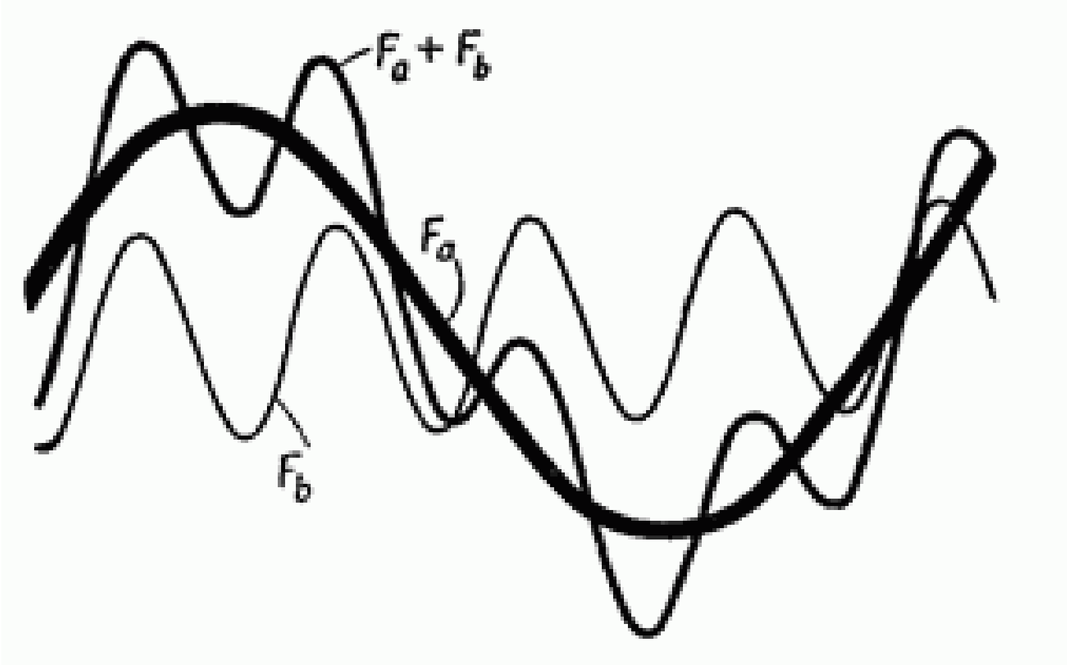
- Корректность получателя
- Антифрод
- Проверка доступности средств

- Скорость проведения платежа – быстрее 5 сек (95% percentile)

- Стоимость разработки

Решение - приоритезируй

- Знай, что убьёт бизнес прежде всего
- Знай, чем можно пожертвовать



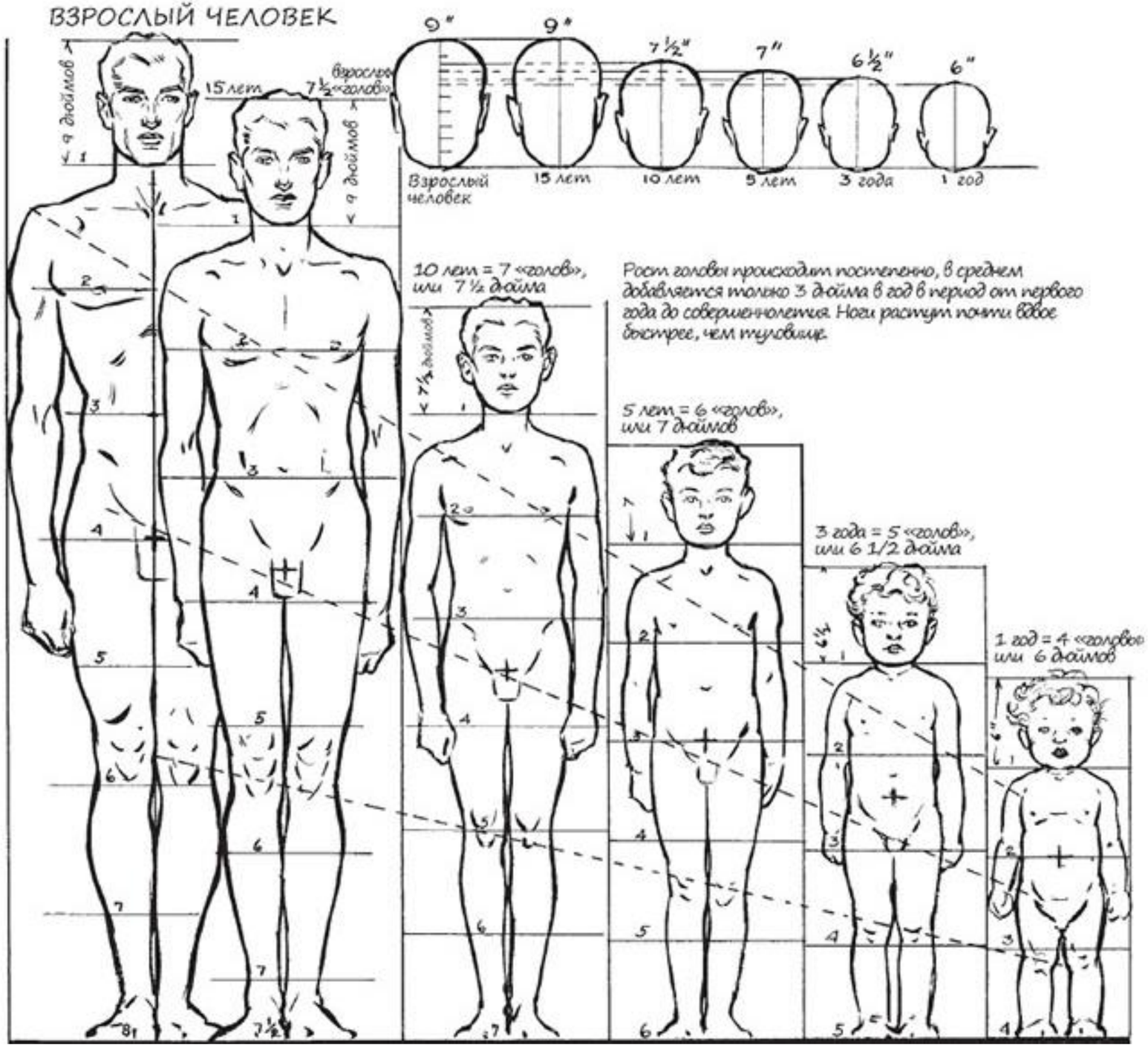
Архитектура это:

курс развития системы

и набор приоритезированных ограничений в которых система будет развиваться длительное время

*Когда человек не знает, к какой пристани он держит путь, для него ни один ветер не будет попутным. //*Сенека

Изменчивый мир


ВЗРОСЛЫЙ ЧЕЛОВЕК
9 дюймов
15 лет
7½ «голов»
9 дюймов
9"
9"
7½"
7"
6½"
6"
Взрослый человек
15 лет
10 лет
5 лет
3 года
1 год
10 лет = 7 «голов», или 7 ½ дюйма
7½ дюймов
Рост головы происходит постепенно, в среднем добавляется только 3 дюйма в год в период от первого года до совершеннолетия. Ноги растут почти вдвое быстрее, чем туловище.
5 лет = 6 «голов», или 7 дюймов
3 года = 5 «голов», или 6 1/2 дюйма
1 год = 4 «головы» или 6 дюймов

Мы не готовы к изменениям

Решение – Зоны вариативности

- Выяснять, где будет наибольшая вариативность
- Постараться оставить наибольшую свободу в этих местах

Всё забывается, люди меняются

Причины принятия тех или иных решений забываются
Договорённости как делать правильно, тоже

Решение – Автоматизация

- Лучший способ не забыть - сделать автоматическую проверку
- ArchUnit

- Что не автоматизируется – задокументировать с фокусом на главном
- Architecture Decision Records

Architecture Decision Records

- Контекст
- Решение
- Статус
- Последствия

Не все решения долговечны

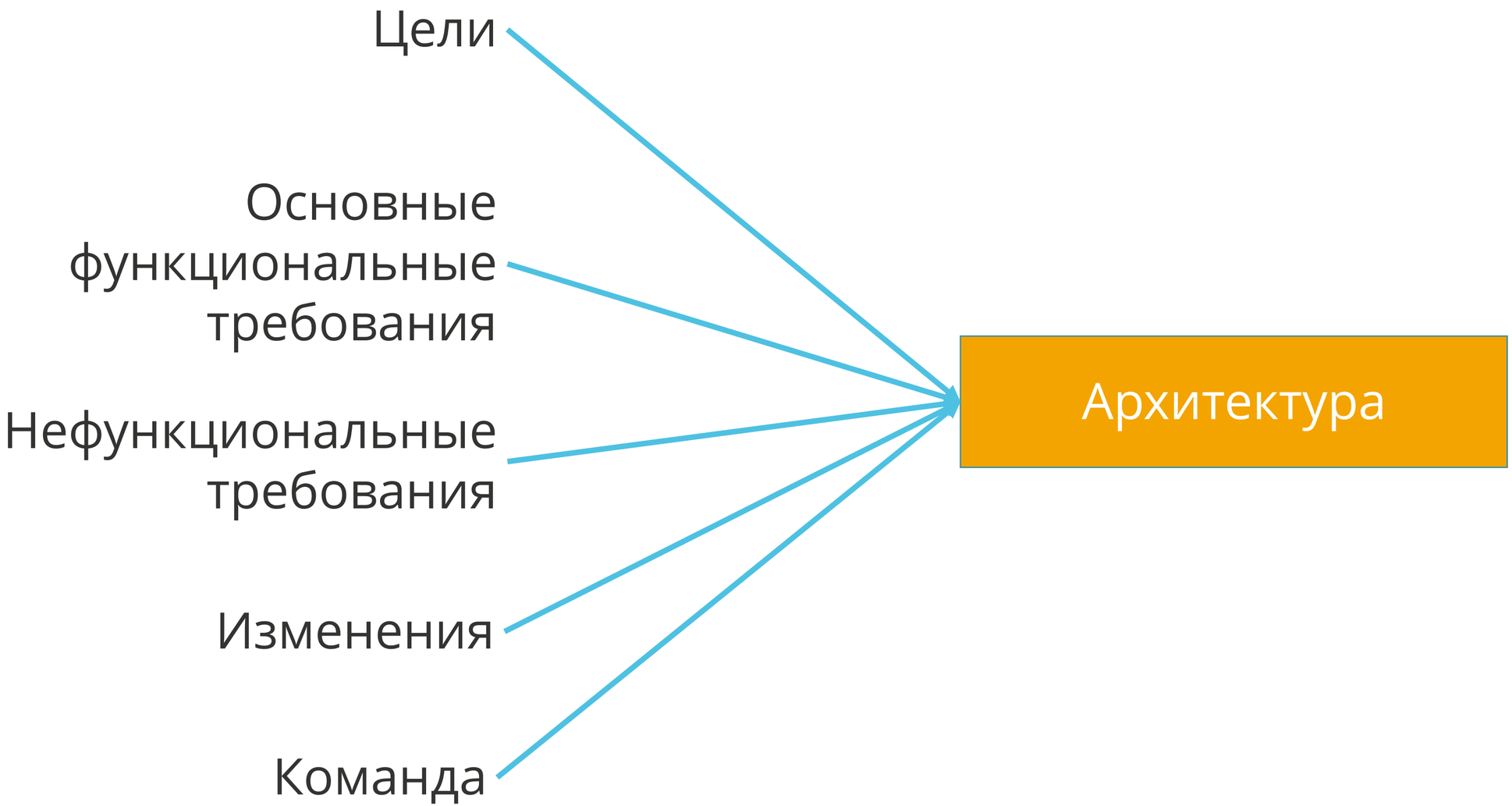
То, что было хорошо вчера, может быть плохим сегодня

Архитектура это:

предусмотренные степени свободы, там где это может быть нужно

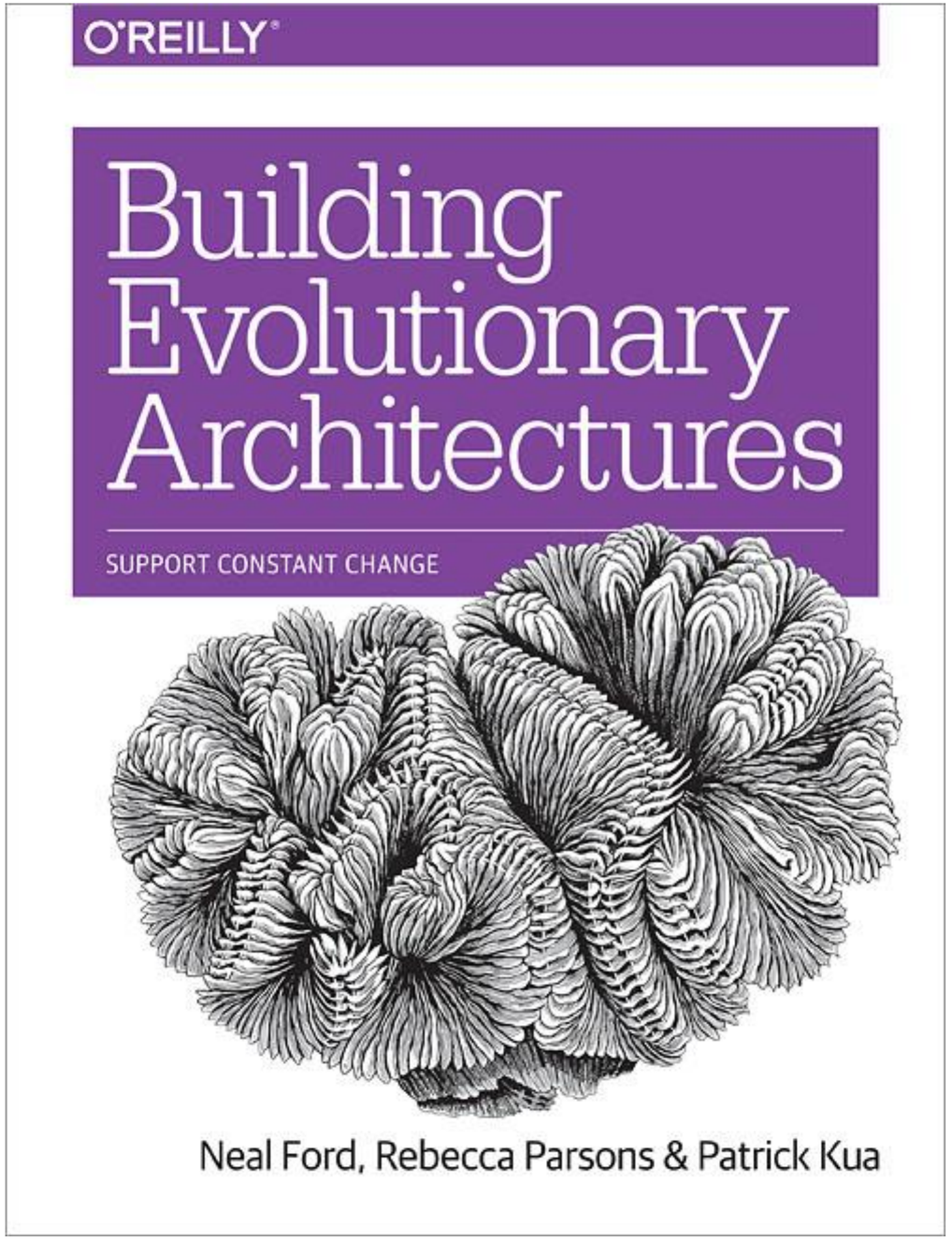
контроль того, что требования соблюдаются после изменений

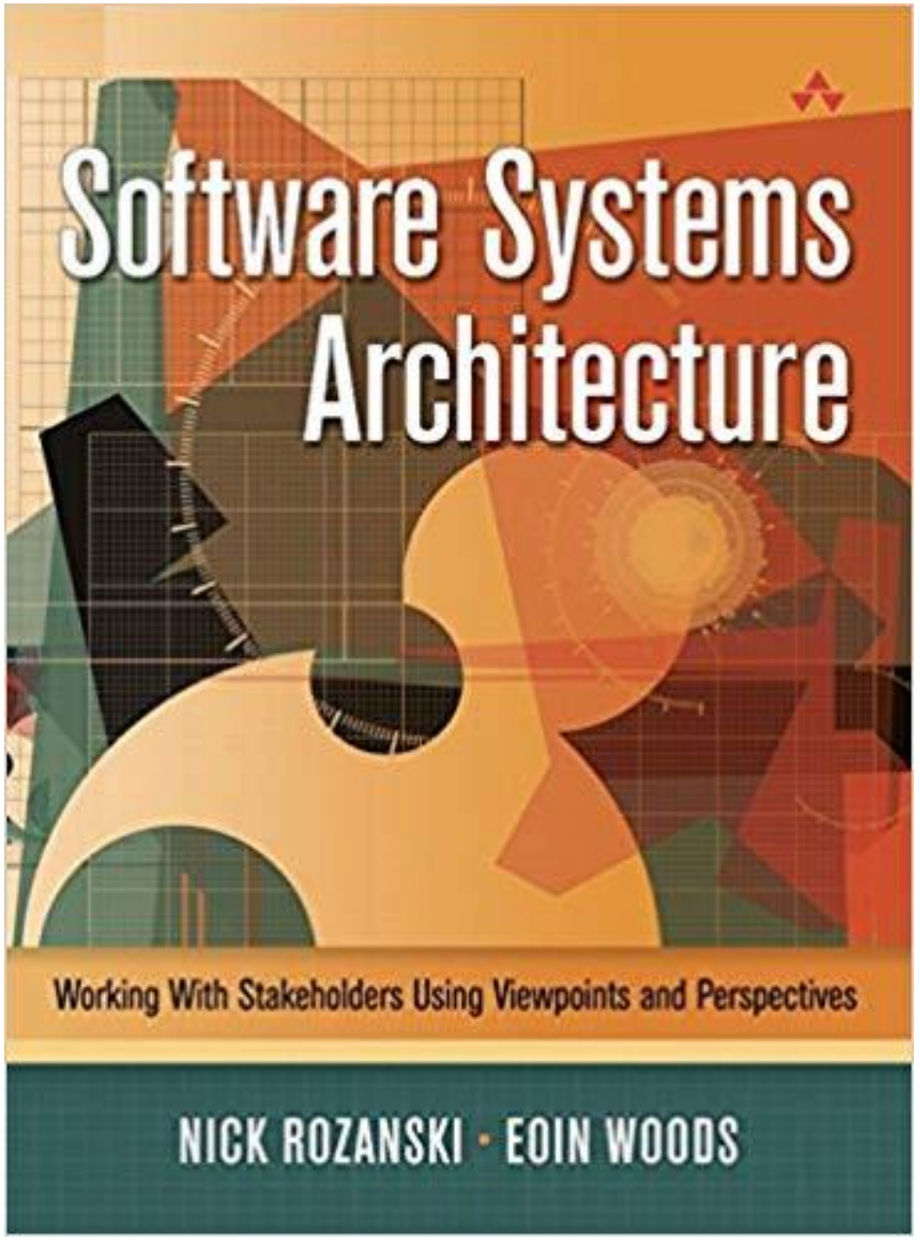
Заключение

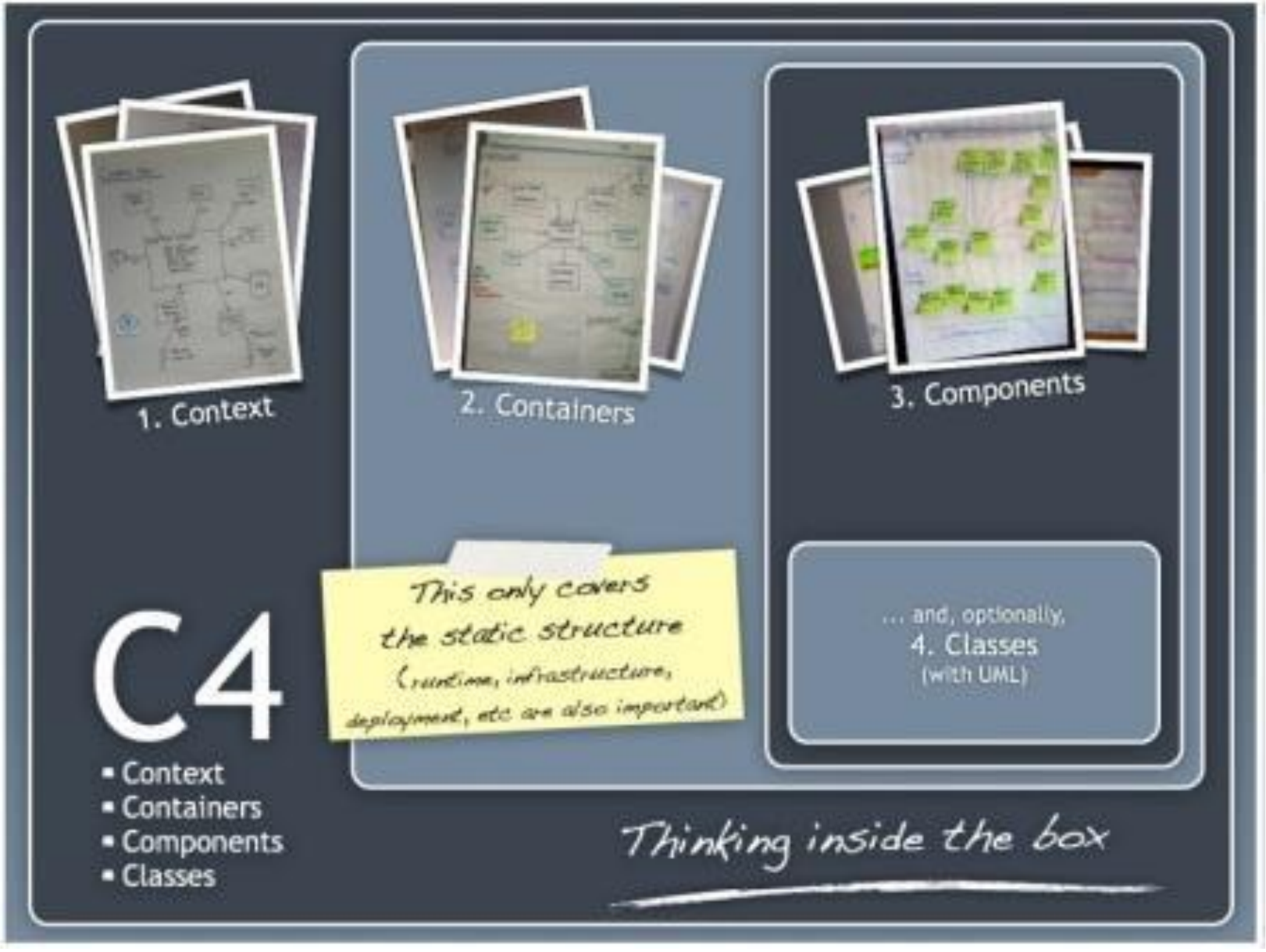

Цели
Основные функциональные требования
Нефункциональные требования
Изменения
Команда
Архитектура

Хрустальный шар не работает

- Делай эксперименты
- Делай мониторинг реальной системы
- Не верьте себе на слово
- Не верьте своему предыдущему опыту
- Все врут (с)


O'REILLY
Building Evolutionary Architectures
SUPPORT CONSTANT CHANGE
Neal Ford, Rebecca Parsons & Patrick Kua


Software Systems Architecture
Working With Stakeholders Using Viewpoints and Perspectives
NICK ROZANSKI · EOIN WOODS


1. Context
2. Containers
3. Components
C4
This only covers the static structure
(runtime, infrastructure, deployment, etc are also important)
... and, optionally,
4. Classes
(with UML)
Context
Containers
Components
Classes
Thinking inside the box

Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

Who Needs an Architect?

Martin Fowler

Wandering down our corridor a while ago, I saw my colleague Dave Rice in a particularly grumpy mood. My brief question caused a violent statement, "We shouldn't interview anyone who has 'architect' on his resume." At first blush, this was an odd turn of phrase, because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry's standards, "architect" and "architecture" are terribly overloaded words. For many, the term "software architect" fits perfectly with the smug controlling image at the end of *Matrix Reloaded*. Yet even in firms that have the greatest contempt for that image, there's a vital role for the technical leadership that an architect such as Dave plays.

What is architecture?

When I was fretting over the title for *Patchitect*.) However, as so often occurs, inside the blighted cynicism is a pinch of truth. Understanding came to me after reading a posting from Ralph Johnson on the Extreme Programming mailing list. It's so good I'll quote it all.

A previous posting said

> The RUP, working off the IEEE definition, defines architecture as "the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Johnson responded:

> I was a reviewer on the IEEE standard that used that, and I argued uselessly that this was clearly a completely bogus definition. There is no highest level concept of a system. Customers have a different concept than developers. Customers do not care at all about the structure of significant

Спасибо за внимание!

Михаил Дружинин
md@rcbd.org