Software Life Cycle Models

HUSZERL Gábor
huszerl@mit.bme.hu

MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

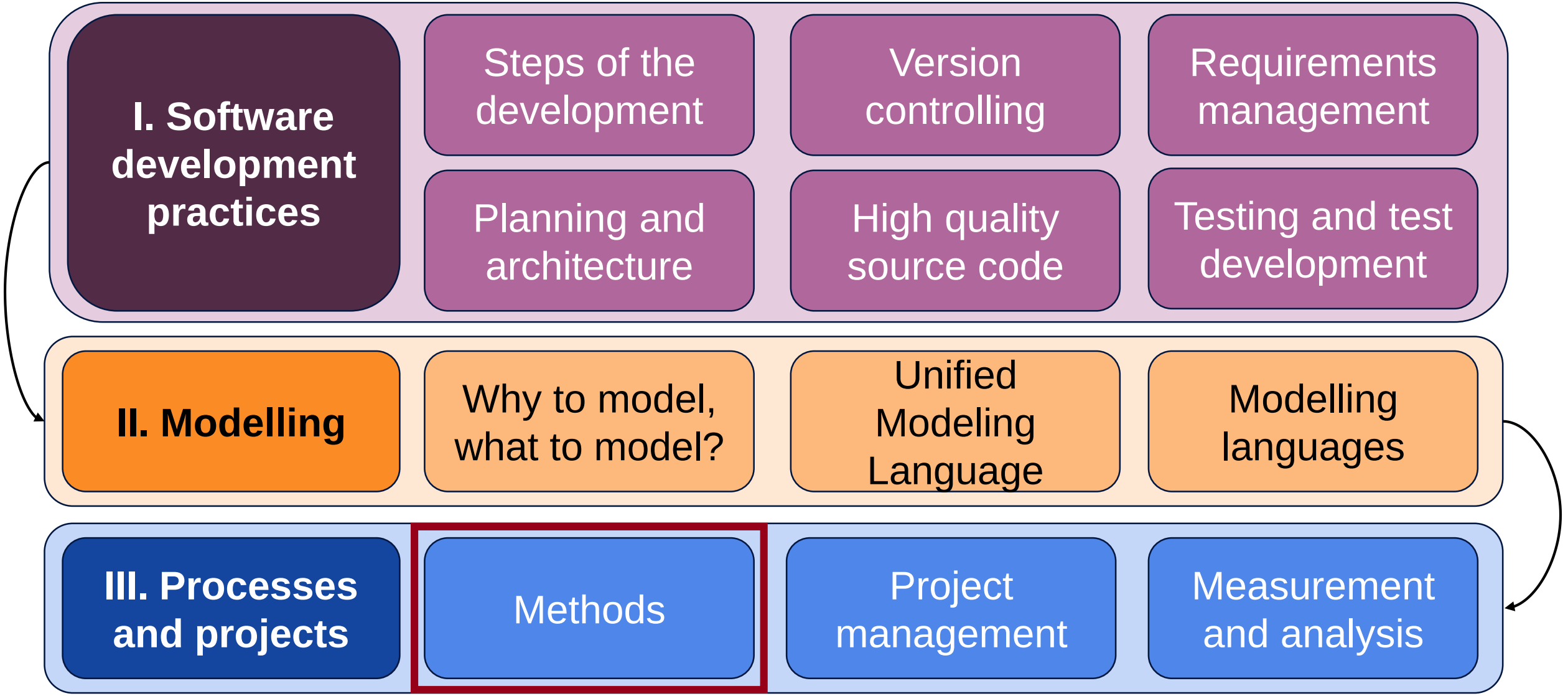
Learning Outcomes

- At the end of the lecture the students are expected to be able to

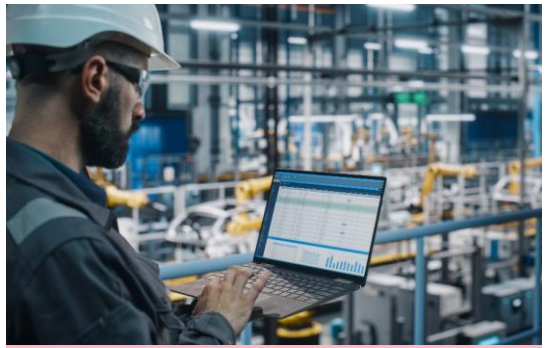
- (K1) understand the goals of the software life cycle models, and their application,

- (K1) identify the basic life cycle models (Waterfall, V-Model, Agile development).

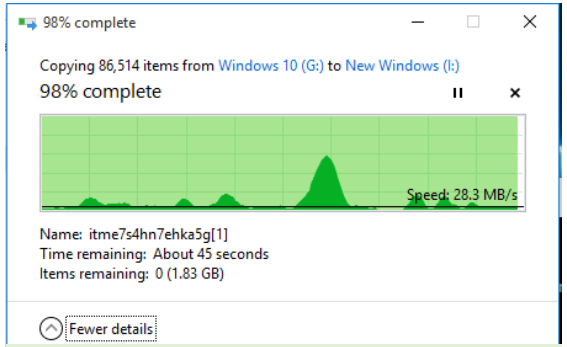
Further Topics of the Subject



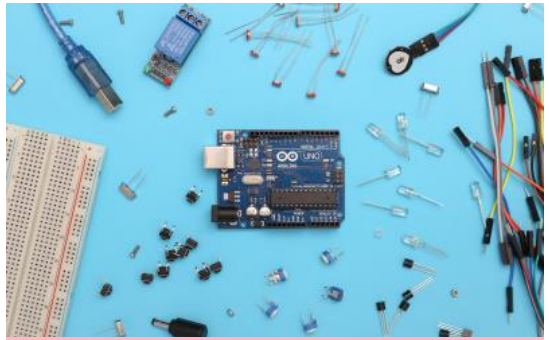
What Are the Specificities of Software (Rep.)



Production

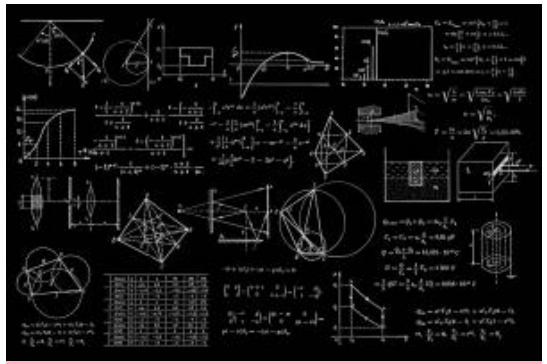


Easy reproduction



Occurrences of faults

Consistency



Physical constraints

Logical concepts

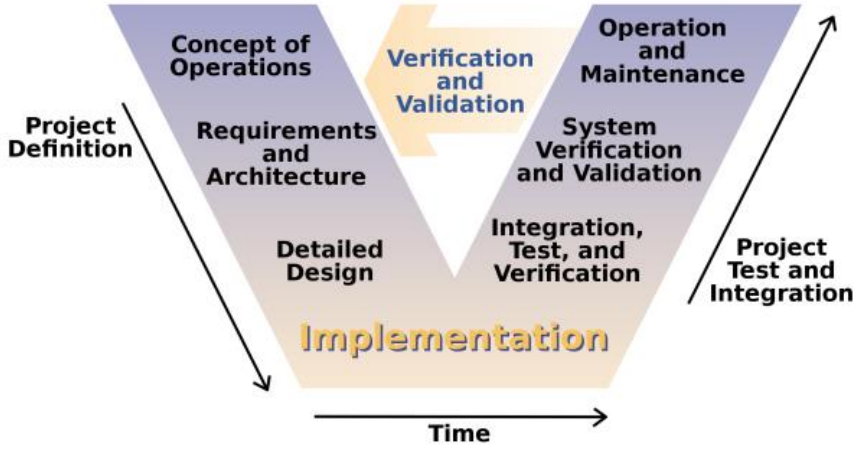
Difficult modification

Easy rewrite

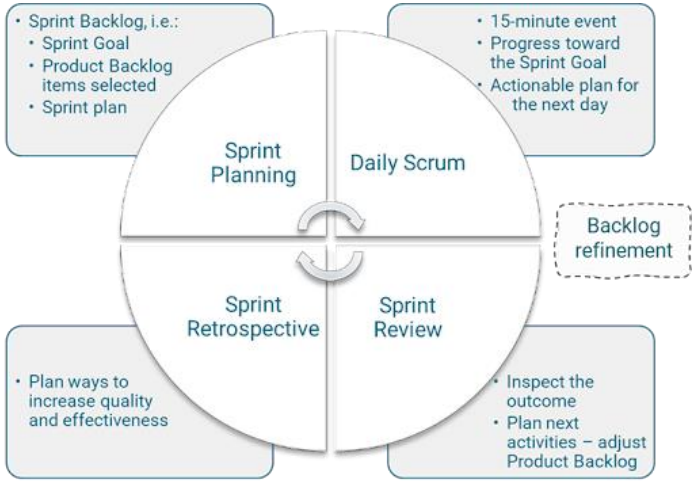
Pictures: Unsplash

Software Engineering Methods (Repetition)

How to make (good) software?



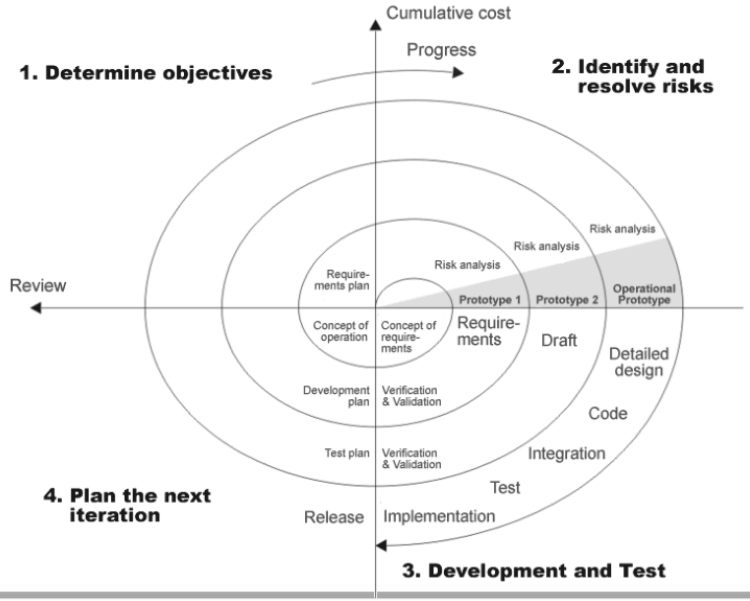
V-Model



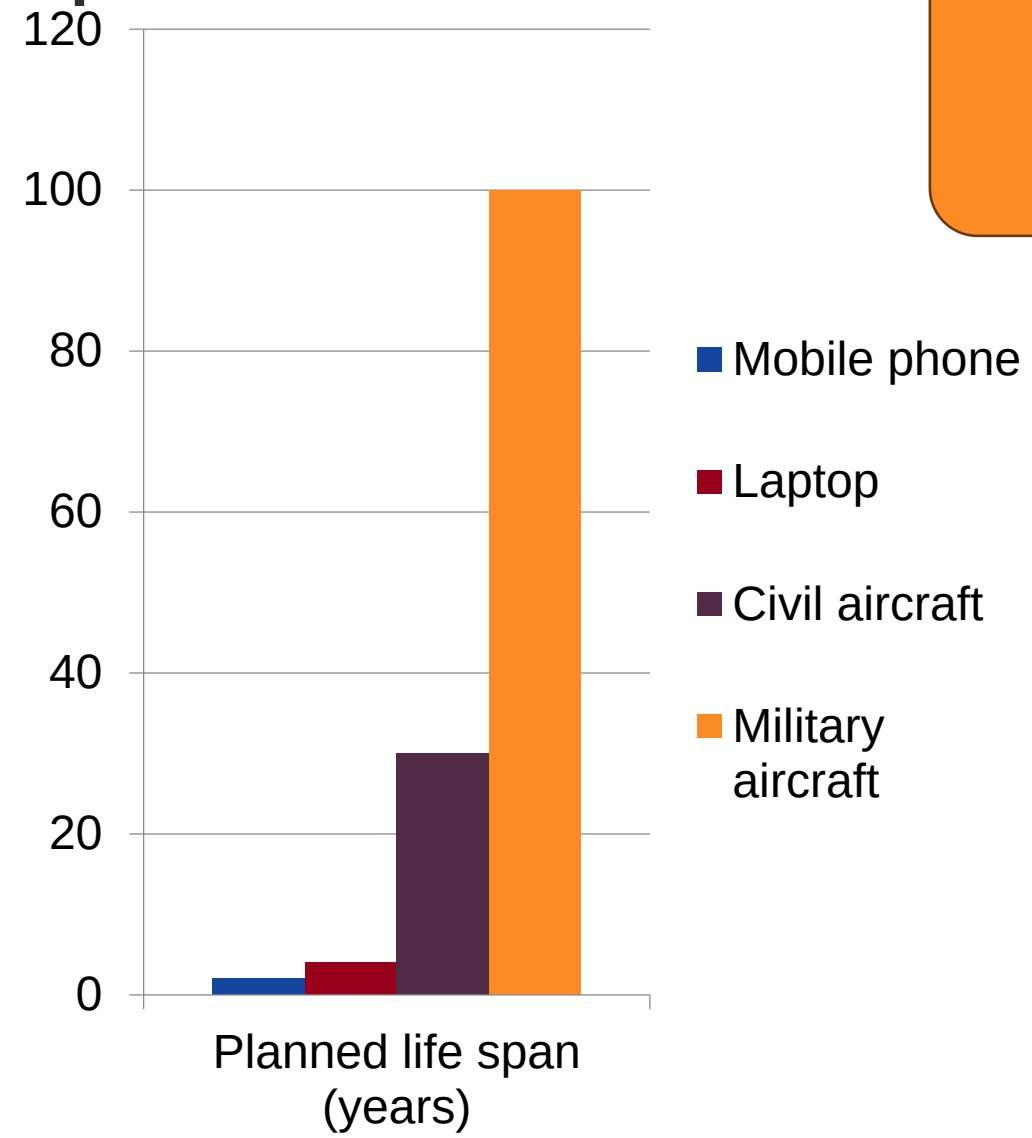
Scrum (agile)

DevOps

Life Cycle


Cumulative cost
Progress
1. Determine objectives
2. Identify and resolve risks
Review
Require-ments plan
Risk analysis
Risk analysis
Risk analysis
Prototype 1
Prototype 2
Operational Prototype
Concept of operation
Concept of require-ments
Require-ments
Draft
Detailed design
Development plan
Verification & Validation
Code
Test plan
Verification & Validation
Integration
Test
4. Plan the next iteration
Release
Implementation
3. Development and Test

Life Span

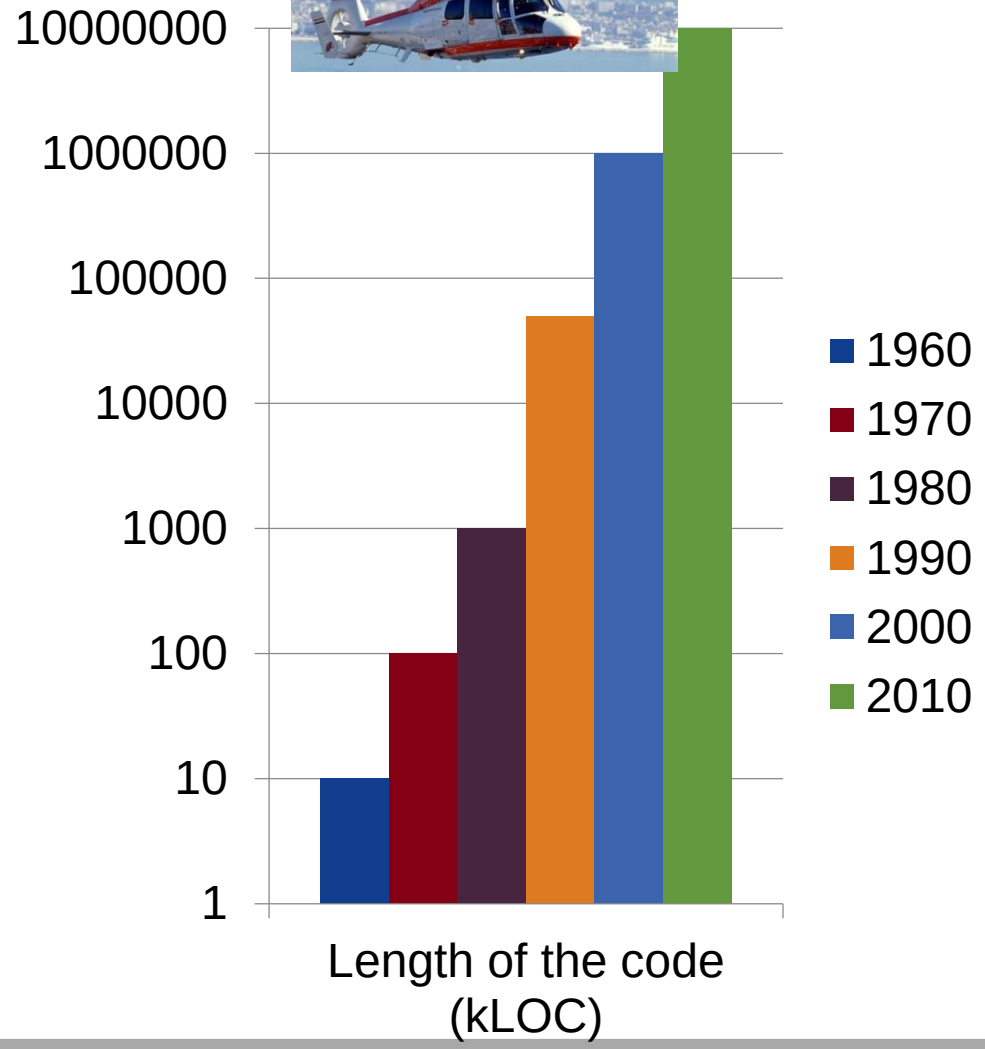


The expected life span of an aircraft: 30-100 years

In these 30 years, no dangerous errors may occur!

Main design aspects:
outstanding quality
profitable production
safe operation

“Size” of the Software



Nowadays a critical system consists up to 70-80% software!

Each main software component > 1 million lines of code

An average programmer makes 10-12 errors in each 1000 lines of code

How can they still work?

Software Development Life Cycle

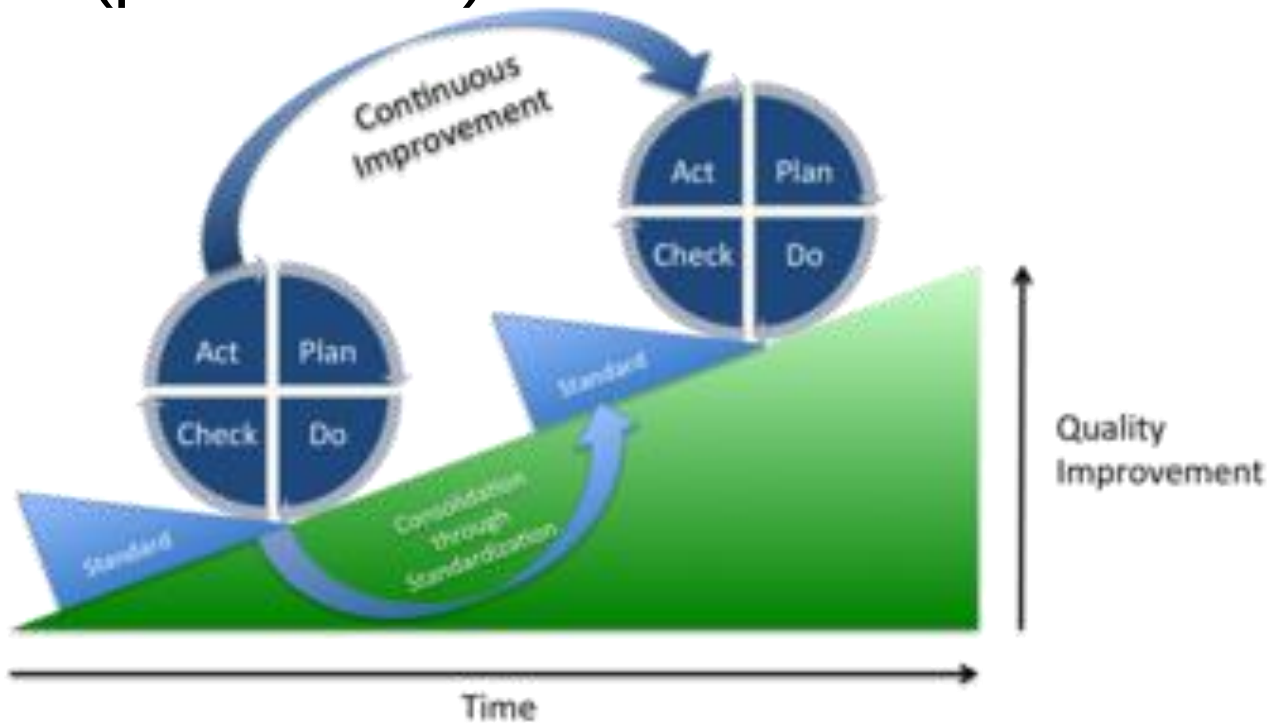
- Software Development Lifecycle Process
- 1970: SDM (Pandata, The Netherlands)
- Bell Telephone Laboratories, 1920 (production)

→Deming, 1940

→*Plan – Do – Check – Act*

From where is it familiar?

Agile development



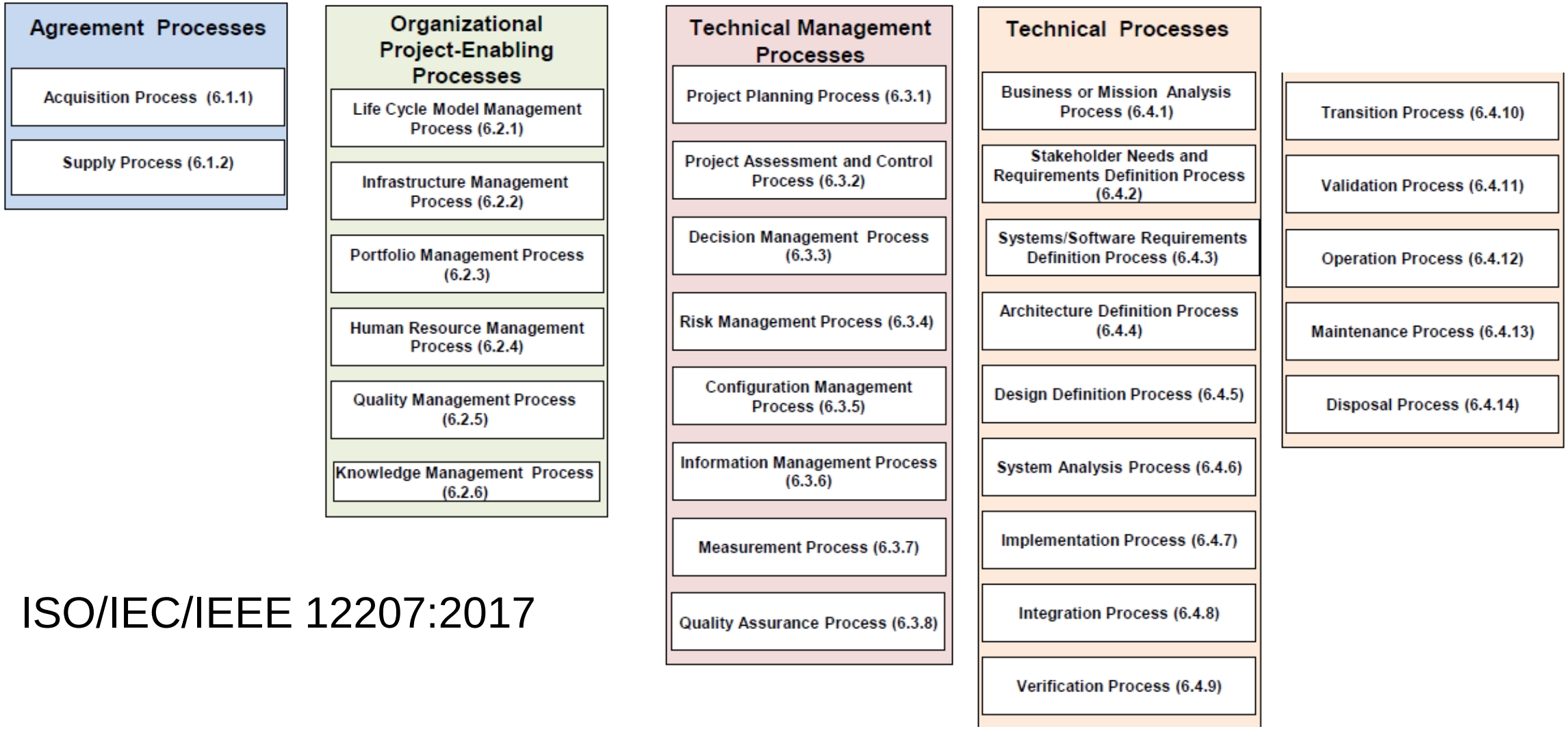
Life Cycles, Processes of Development and Operation

- Historical background
  - Scheduling (e.g. GANTT diagrams)
  - Modelling of production/administrative processes
  - IDEF-0: US AirForce, 1980s
  - Describing logistics processes
  - IT operation: “runbook”

Common Elements

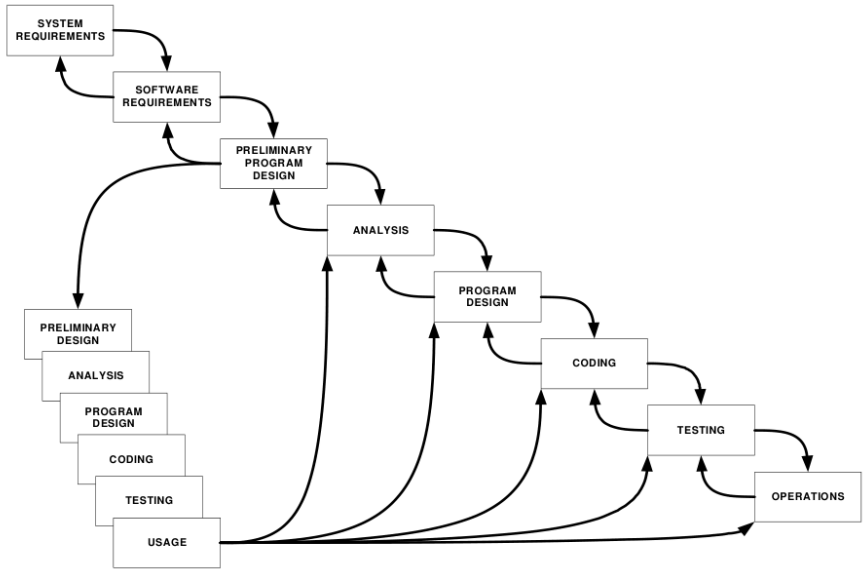
Steps, **phases**	• With well defined **outputs**
Their dependencies	• Time? • Data? • Order?
Decision points	• When is it ready/complete? • Does it satisfy the original design/requirements?
Roles	• Entitlement • Responsibility/approval

System Development – Life Cycle Processes

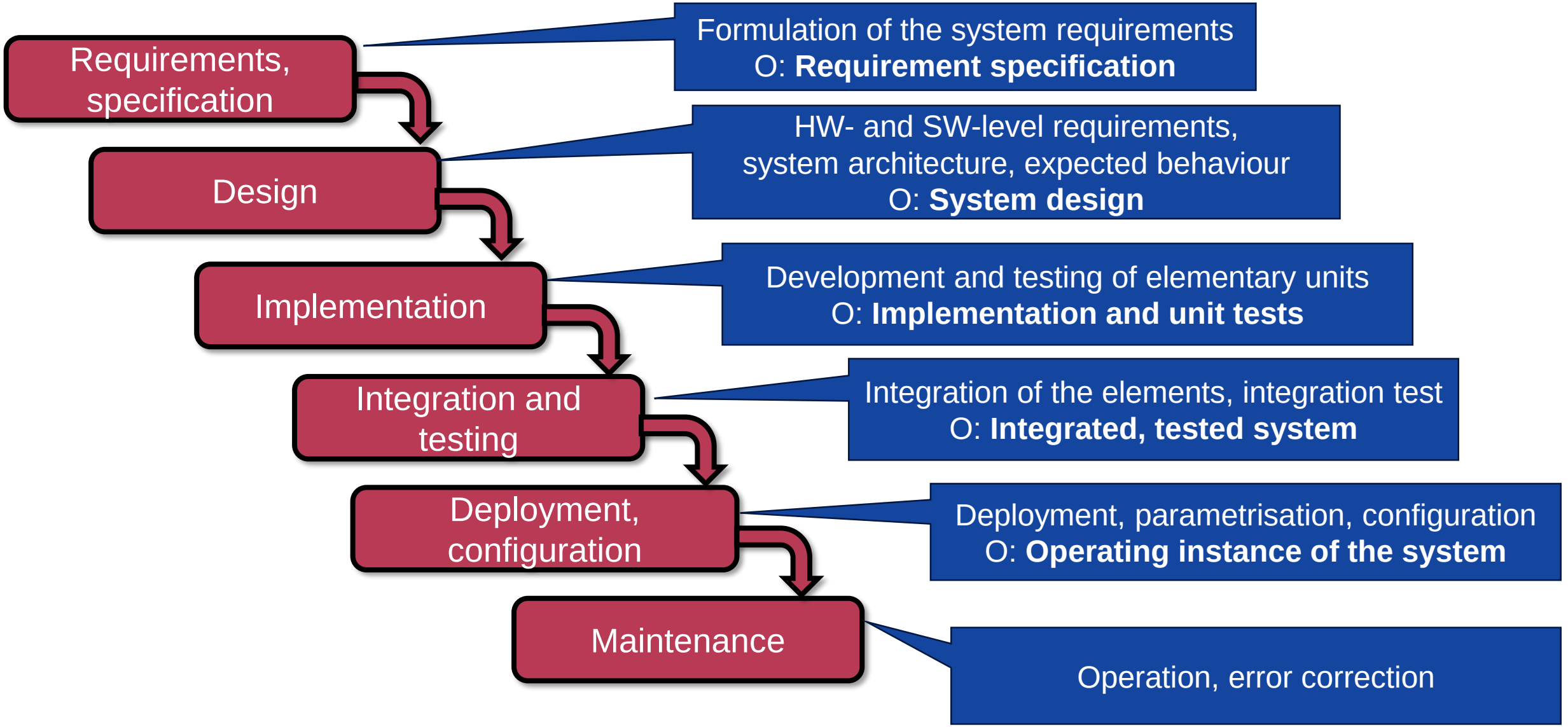


ISO/IEC/IEEE 12207:2017

Waterfall Model



The Waterfall Model



The Role of the Waterfall Model

- The first well defined software life cycle model
  - Winston W Royce. Managing the development of large software systems. In proceedings of IEEE WESCON, number 8, pages 328–338. Los Angeles, **1970.**
- Advantages
  - Simple, clear structure
  - Well defined flow of information
- Disadvantages
  - Handling changes is difficult
  - Client cannot see any results until very late

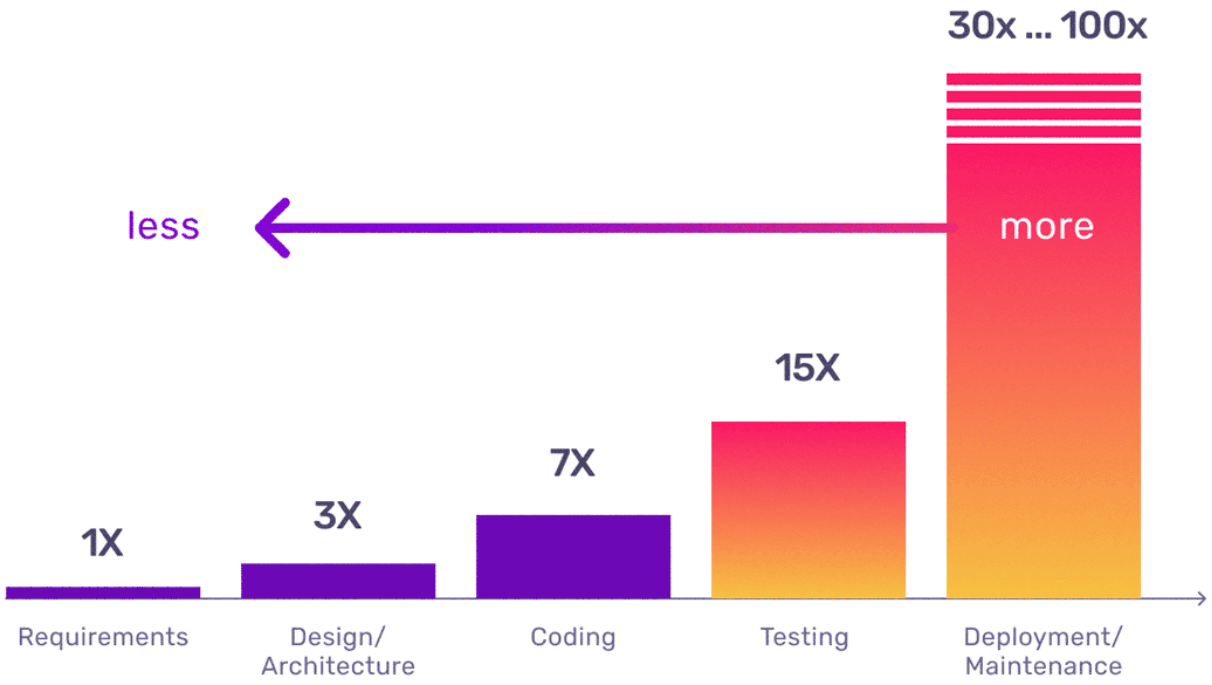
Case Study (from real life)

- A system is already operating. Some specific data should become …
- … editable
  - Consistency? Integrity?
- … pre-set, but editable for those who require this
  - Roles? Entitlements? Access management?
- … editable but with an external tool
  - Maintainability? Error corrections?

At the end the feature request was cancelled.

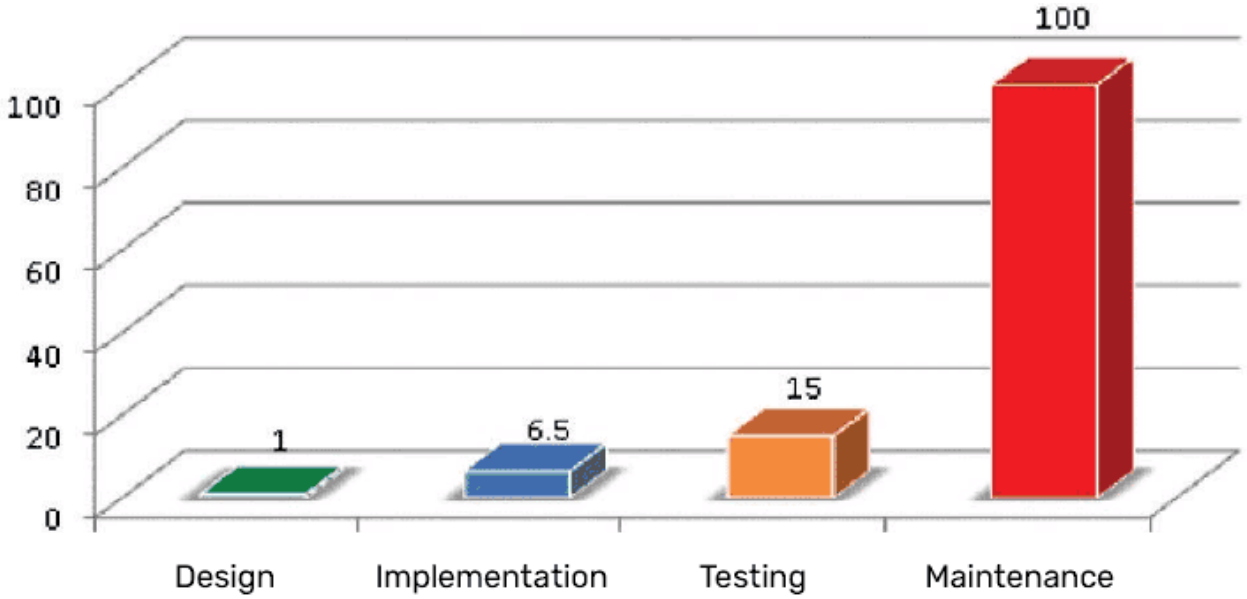
The Costs of Error Correction

Costs/Damage caused by the errors



National Institute of Standards and Technology

Costs of error correction



IBM Service Sciences Institute

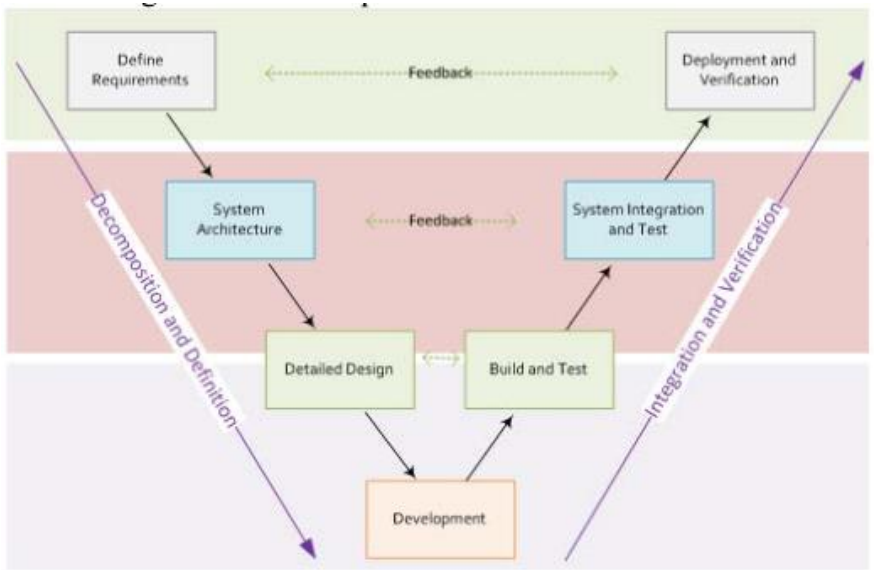
When Is the Waterfall Model a Good Choice?

- The project is a small one
- The technology is already known (little risk)
- The requirements are known (little risk)
  - Many iterations would cost much …

What Is the Solution?

- Systematic measurements
  - Additional costs and time
  - Higher quality
  - Predictability
- Changing and backtracking
  - Iterative methods

V-Model

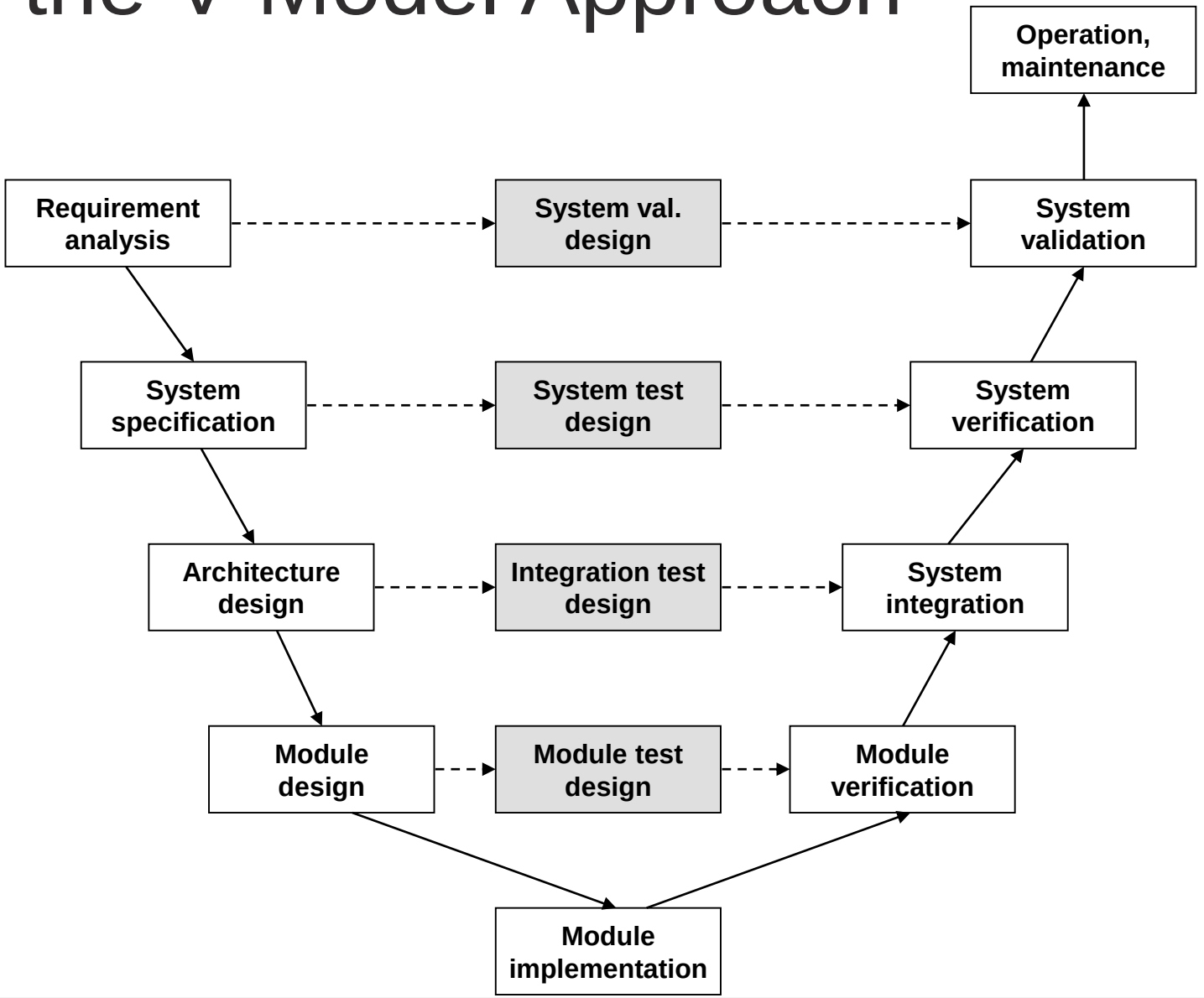


Development of critical systems

What Is a Critical System?

- Safety critical system
  - An error may have catastrophic consequences
  - Environmental damages, human lifes
- Mission critical system
  - Critical for the existence of an organisation
- Business critical system
  - Critical for the normal operation of an organisation
  - E.g. the account management system of a bank
- Primary aspect during the development
  - Traceability
  - Specific standards must be applied
  - DO187-C, ISO 26262, AS9001

Steps of the V-Model Approach



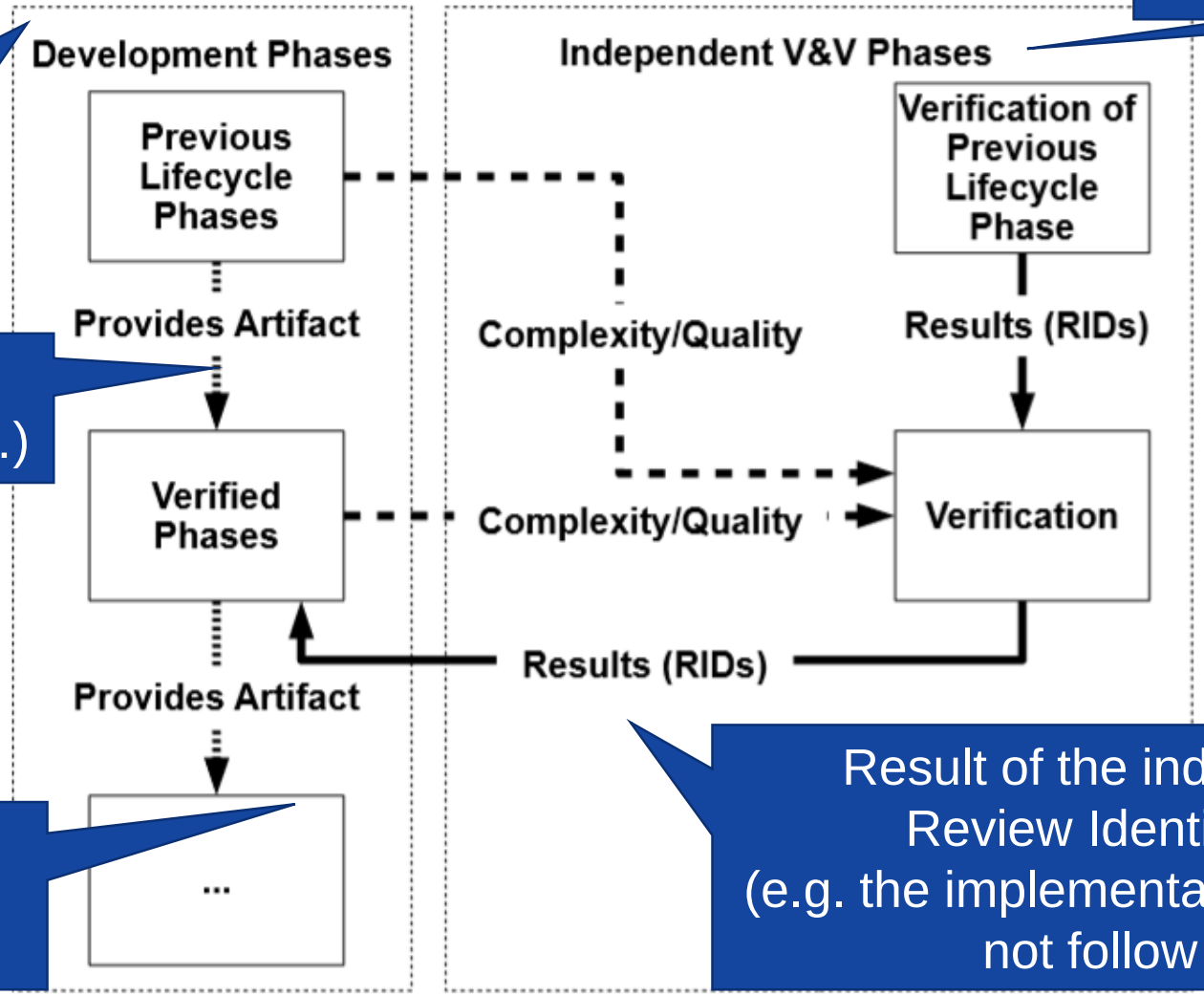
Requirement Modelling

- Goal: unambiguous, manageable specification
- Harmonisation of the terminology
- Traceability:
  - From the requirement to the code (and to the tests)
  - From which requirement does a given element result?
  - Comments / annotations
- Standard: OMG ReqIF (Requirements Interchange Format)
- Tools:
  - (Excel)
  - IBM DOORS
  - Eclipse RMF
  - FormalMind

Certification Process

Independent external checks

Development project may take years



Phases build upon each other (reqs → design → …)

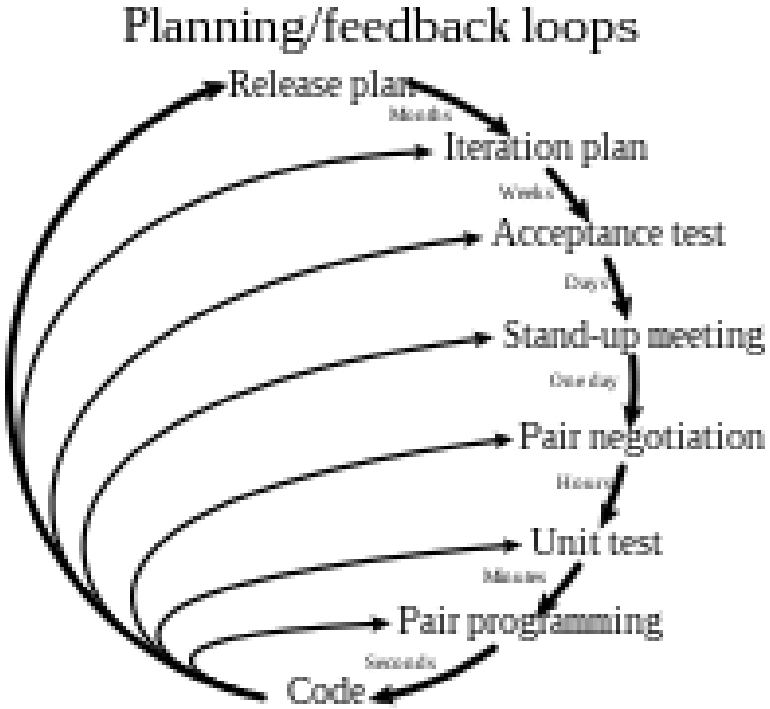
Project members may change

Result of the independent checking:
Review Identified Discrepancy
(e.g. the implementation of the protocol does not follow the standard)

Agile Development

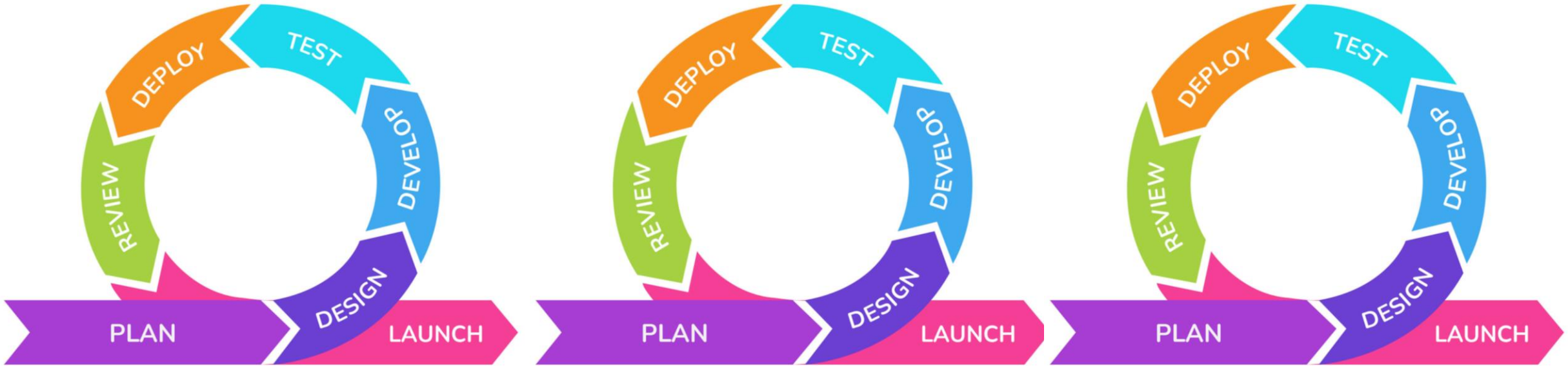
Quality + Flexibility?

- Forerunner
- Extreme Programming (XP) – Beck, 1999
  - „code, test, listen, design”



- Motivation
  - Appearing of the software modelling languages
  - Internet-based services – faster development, changing “products”

Agility – Changes Are Parts of the Development



“Manifesto” – 12 Points

Goal: satisfying the customer, early and continuous delivery	• Have an operating version, even if it will change later
Welcome changing requirements	• Even late in the development
Frequent delivery	• Even bi-weekly
Business people and developers work together	• Daily throughout the project, not only at „ordering/delivery”
Trust individuals	• Support motivated members → they get job done • (cf. “punch clock”)

“Manifesto” – 12 Points

Information exchange – face-to-face conversation	• There is no carved-in-stone specification
Working software is the primary measure of progress	• There are no “indicators for indicators’ sake” (KPIs)
Agile processes promote sustainable development	• The sponsors, developers, and users involved continuously • Maintaining a constant pace indefinitely
Technical excellence and good design	• The good changes are important

“Manifesto” – 12 Points

Simplicity is essential

- The art of maximizing the amount of work not done

Self-organizing teams

- The origin of best architectures, requirements, and designs

Reflecting on effectivity at regular intervals

- Tuning and adjusting the behaviour accordingly

Implementing Agile Projects

- SCRUM methodology
  - “Ceremonies”
  - Roles and “products”
- Technical support
  - Development and teamwork tools
  - Kanban (in the next lecture)

Comparison

Method of Project Evaluation

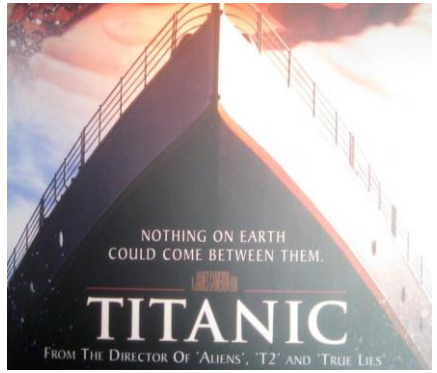
- Alfred Spector, 1986
  - Bridge building (civil engineering) ←→ Software development
- Usually, bridges
  - Do not collapse
  - They get ready on time …
  - … and on budget (???)

Measuring Success

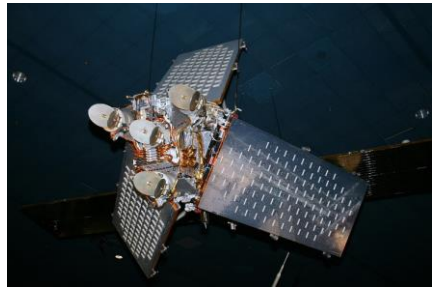
- Basic principles (OTOBOS)
  - On Time
  - On Budget
  - On Scope
- Why are they not enough?
  - Quality, success, maintainability, …

Success – Counterexamples

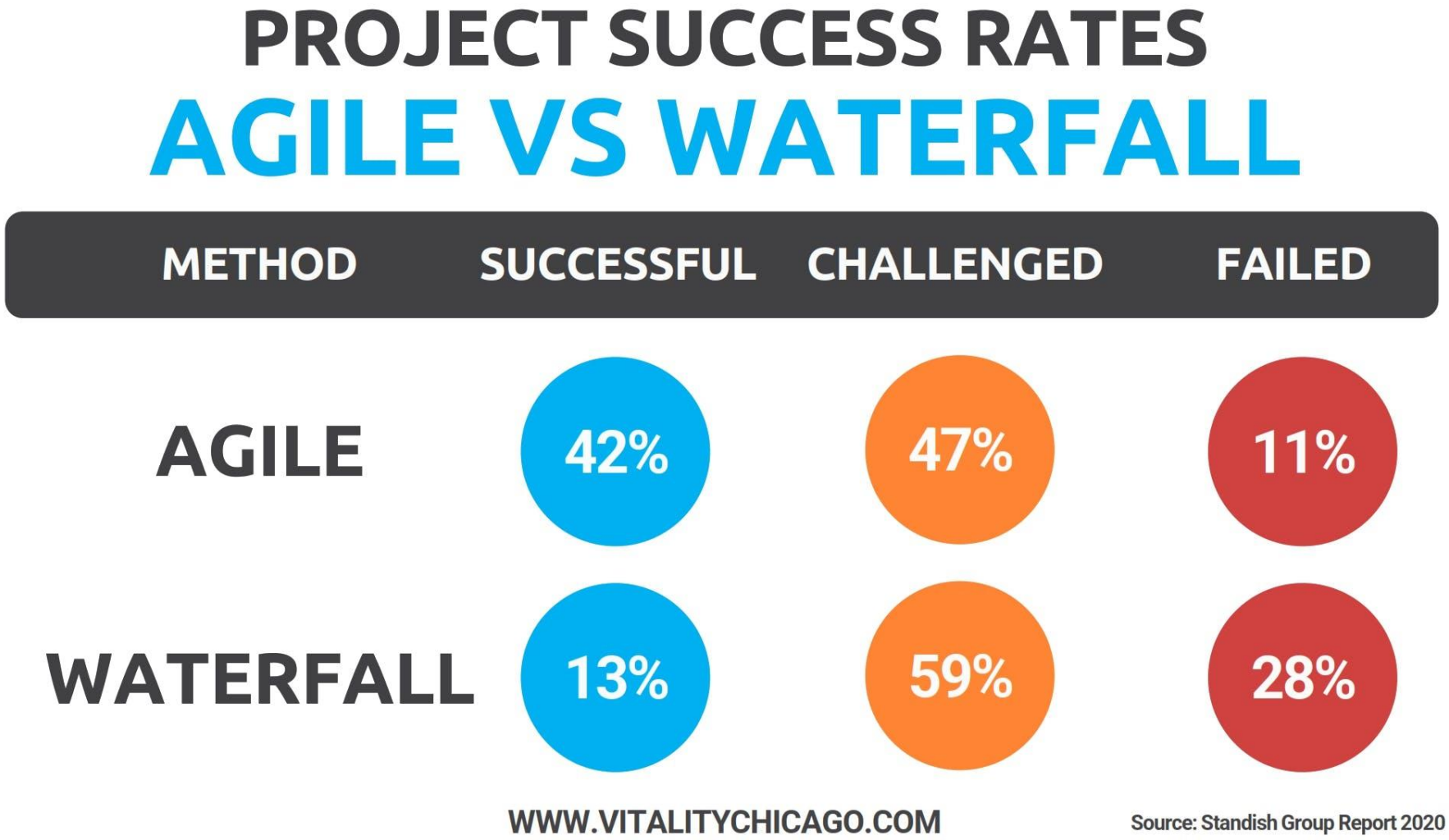
- Successful bad project
  - Titanic movie (1997)
  - Double costs (100→200 M USD)
  - Considerable delay
  - 11 Oscars, 2 billion USD



- Unsuccessful good project
  - Motorola Iridium
  - Satellite phone network
  - 66 satellites
  - 1993-1998, completed on time
  - Phone device: 3000 USD



Comparison of the Agile and Waterfall Models



Summary

Summary

- Software development life cycle
  - Goal: systematic description of the design, development, and operation of the software
- Waterfall model
  - Strictly separated phases, inflexible but simple
- V-Model
  - Developing critical systems
  - Multi-level design and checking, traceability
- Agile development
  - Short development cycles
  - Fast initial results, changing is part of the process