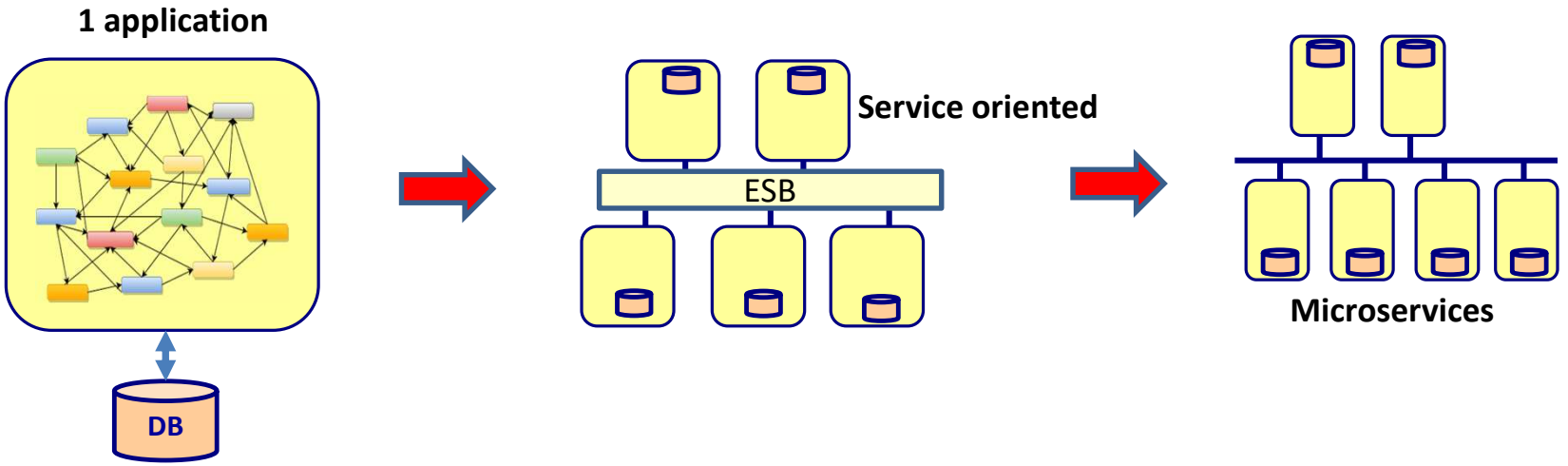
SOFTWARE ARCHITECTURE COURSE OVERVIEW

CS 590 Software Architecture

- Architecture styles
- Architecture patterns
- Architecture principles and best practices



Course agenda

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
April 19 **Lesson 1** Software architecture introduction	April 20 **Lesson 2** Layering & Spring Boot	April 21 **Lesson 3** Domain Driven Design	April 22 **Lesson 4** Databases	April 23 **Lesson 5** Component based design	April 24 **Lesson 6** SOA & integration patterns	April 25
April 26 **Midterm Review**	April 27 **Midterm exam**	April 28 **Lesson 7** Microservices 1	April 29 **Lesson 9** Microservices 2	April 30 **Lesson 9** Microservices 3	May 1 **Lesson 10** Microservices 4	May 2
May 3 **Lesson 11** Microservices 5	May 4 **Lesson 12** Stream based architecture	May 5 **Lesson 13** Finding the right architecture	May 6 **Lesson 14** Architecture analysis	May 7 **Final review**	May 8 **Final exam**	May 9
May 10 **Project**	May 11 **Project**	May 12 **Project**	May 13 **Project**			

LESSON 1
SOFTWARE ARCHITECTURE INTRODUCTION

Why architecture?

More complexity asks for:

- More abstraction and decomposition
- More principles and guidelines
- More communication
- More proces
- More powerful tooling

More complexity asks for more architecture

Why architecture?

- Winchester “mistery” house
- 38 years of construction work– 147 builders 0 architects
- 160 rooms– 40 bedrooms, 6 litchens, 2 basements, 950 doors
- No architecture description
- 65 doors that don’t go anywhere, 13 stairs that don’t go anywhere, 24 skylights where you cannot see the sky

What is software architecture?

- The fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

  - ANSI/IEEE std 1471-2000

What is software architecture?

The things that are difficult to change

A blueprint

Technical leadership

Standards and guidelines

Building blocks

Technical direction

Systems, subsystems, interactions and interfaces

Satisfying non-functional requirements/quality attributes

The big picture

The skeleton/backbone of the product

The system as a whole

Structure (components and interactions)

The things that are expensive to change

What is software architecture?

The important stuff

Whatever that might be

Defining the architecture

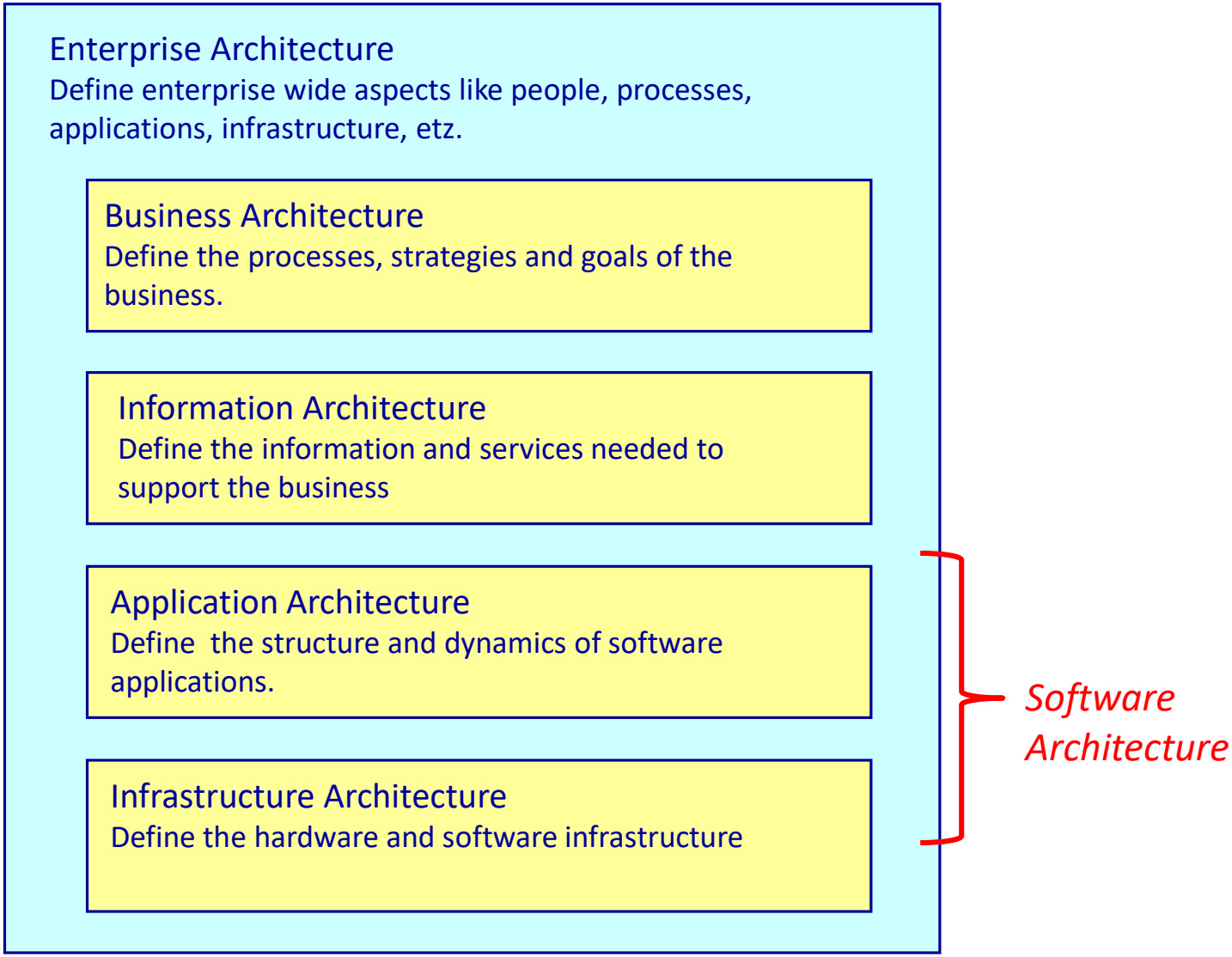
- Define components
- Define component interfaces
- Define platform and language(s)
- Define architectuur styles
- Define architectuur patterns
- Define layers and packages
- Define presentation architecture
- Define persistency architecture
- Define security architecture
- Define transaction architecture
- Define distribution architecture
- Define integration architecture
- Define the deployment architecture
- Define the clustering architecture
- Define the hardware
- Define tools to use

- Decide on solutions for
  - Logging
  - Error management
    - Error detection
    - Error reporting
    - Fault tolerance
  - Event management
  - File handling
  - Printing
  - Reporting
  - Resource management
  - Internationalization
  - Licence management
  - Debugging
  - …

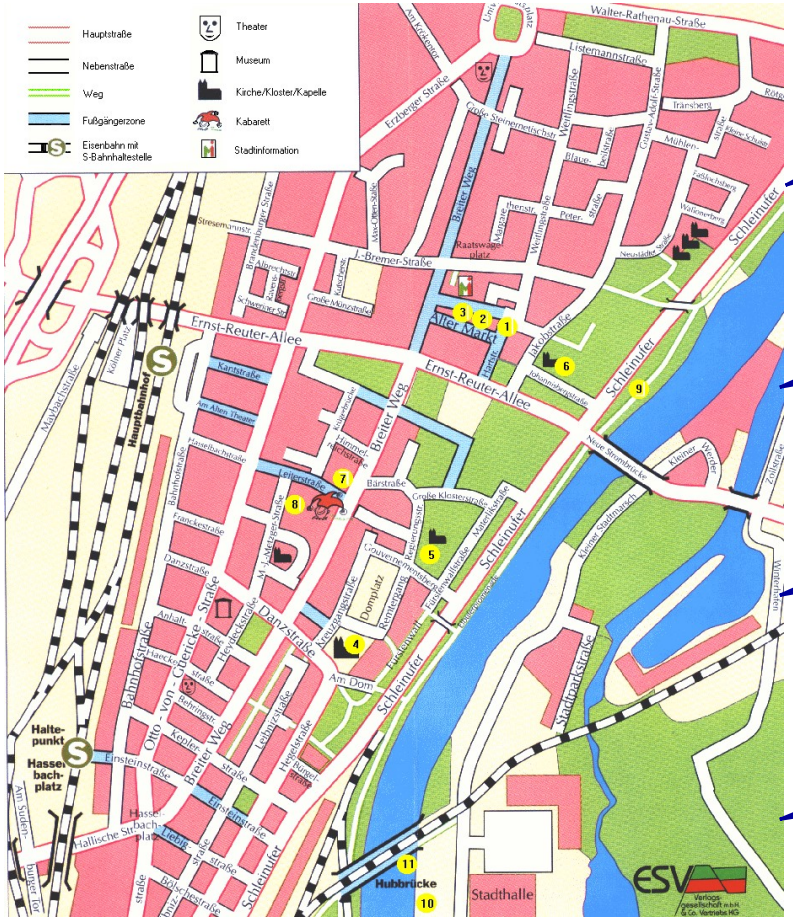
Different kinds of architecture

- Infrastructure
- Security
- Technical
- Solution
- Network
- Data
- Hardware
- Enterprise
- Application
- System
- Integration
- IT
- Database
- Information
- Process
- Business
- Software

Different kinds of architecture



City planning



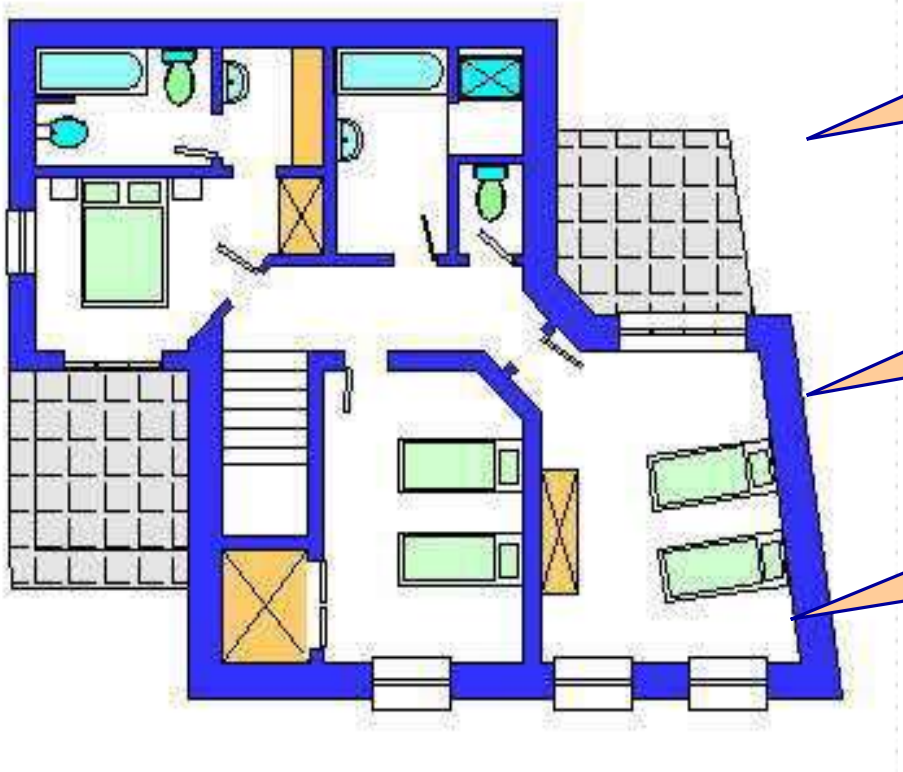
Business Architecture :
Goals of the city and the processes in a city

Information Architecture:
healthcare, education, water, electricity, transport, …

Application Architecture :
hospital, police, library, schools, …

Infrastructure Architecture :
roads, railroads, harbour, airport, …

House planning



Information Architecture:
water, electricity, heating, …

Application Architecture :
Living room, bedrooms, kitchen, …

Infrastructure Architecture :
Electrical wires, water pipes, driveway,…

Warship Vasa

- Customer
  - King of Sweden
    - Gustav II Adolf
- Requirements:
  - 70 m long
  - 300 soldiers
  - 64 guns
  - 2 decks
- Architect
  - Hendrik Hybertson

Software architecture is hard!

- Complexity
  - No physical limitations
  - Huge state-space
- Constant change of
  - Business
  - Technology
- The architecture is never ideal

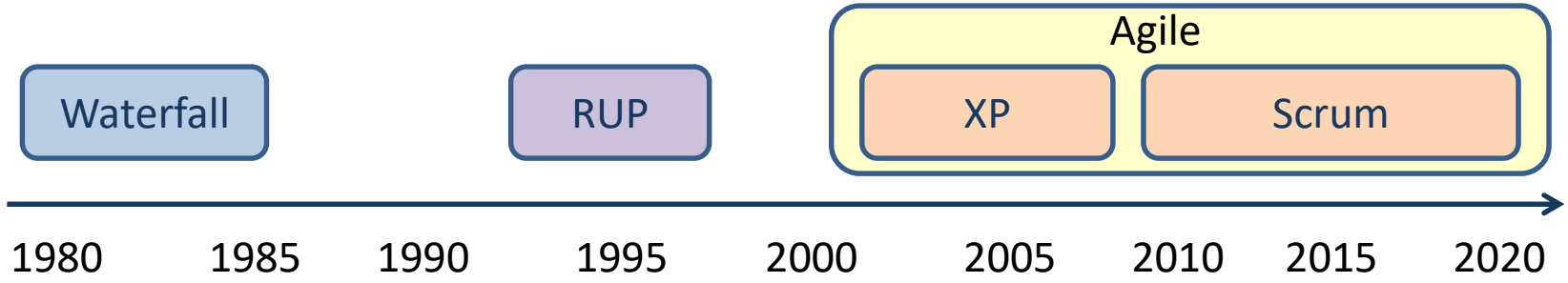
The work of an architect: Make non-optimal decisions in the dark.

Main point

- Software architecture is defining all the important stuff in a software development project.
- The human physiology has the same structure as the structure of the Veda and Vedic literature who are expressions of the structure of pure consciousness.

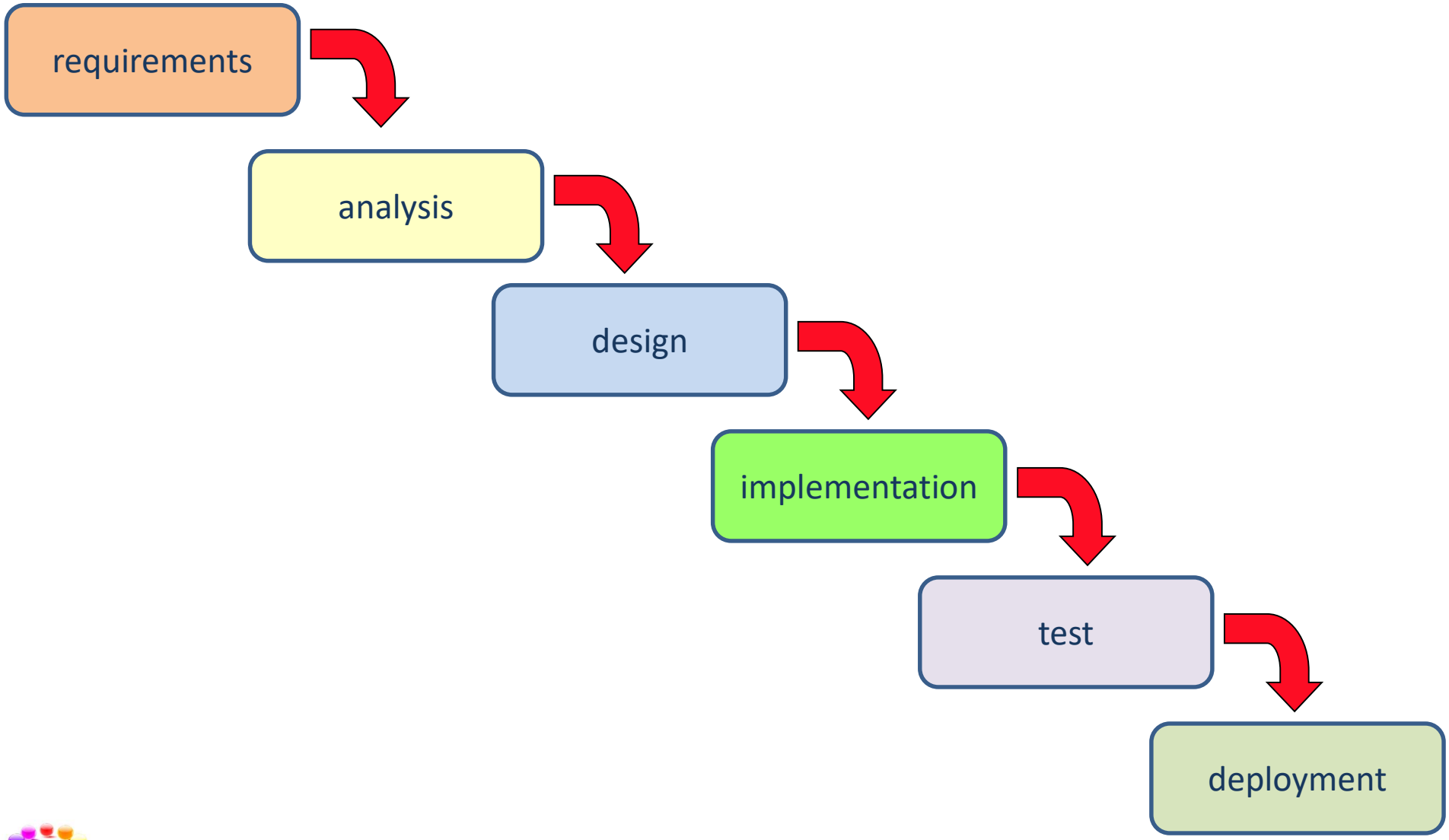
HOW DOES SOFTWARE ARCHITECTURE FIT IN THE PROCESS

Software development methods



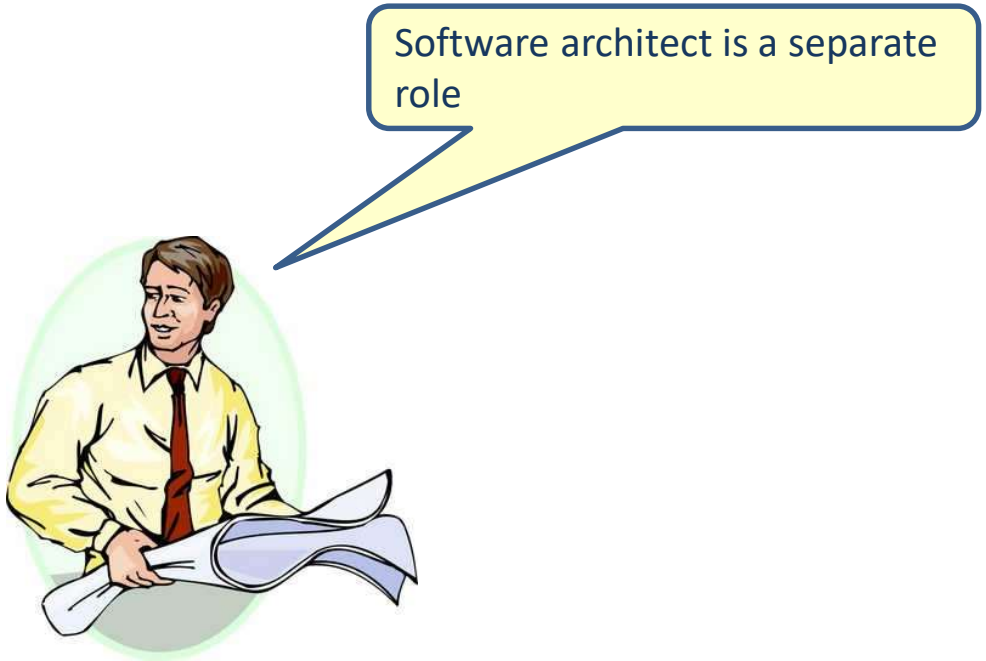
- Linear
- Different roles
- Document driven
- Customer is outside the project
- Large projects(time, nr. of people)
- Req. statements

Waterfall



Core roles in waterfall

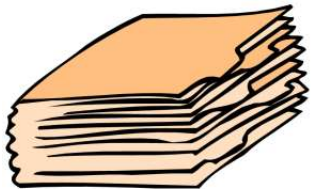
- Project manager
- Analyst
- Developer
- Tester
- Architect



Software Architect

Waterfall software architecture

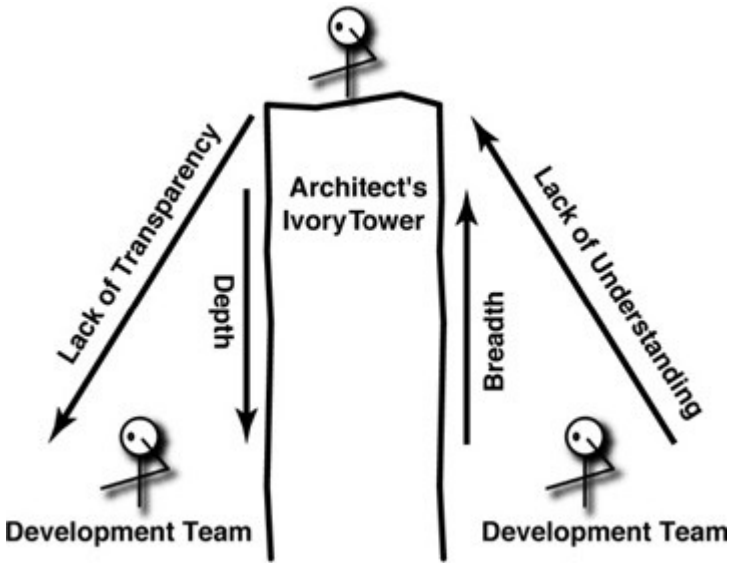
- The architect designs the system



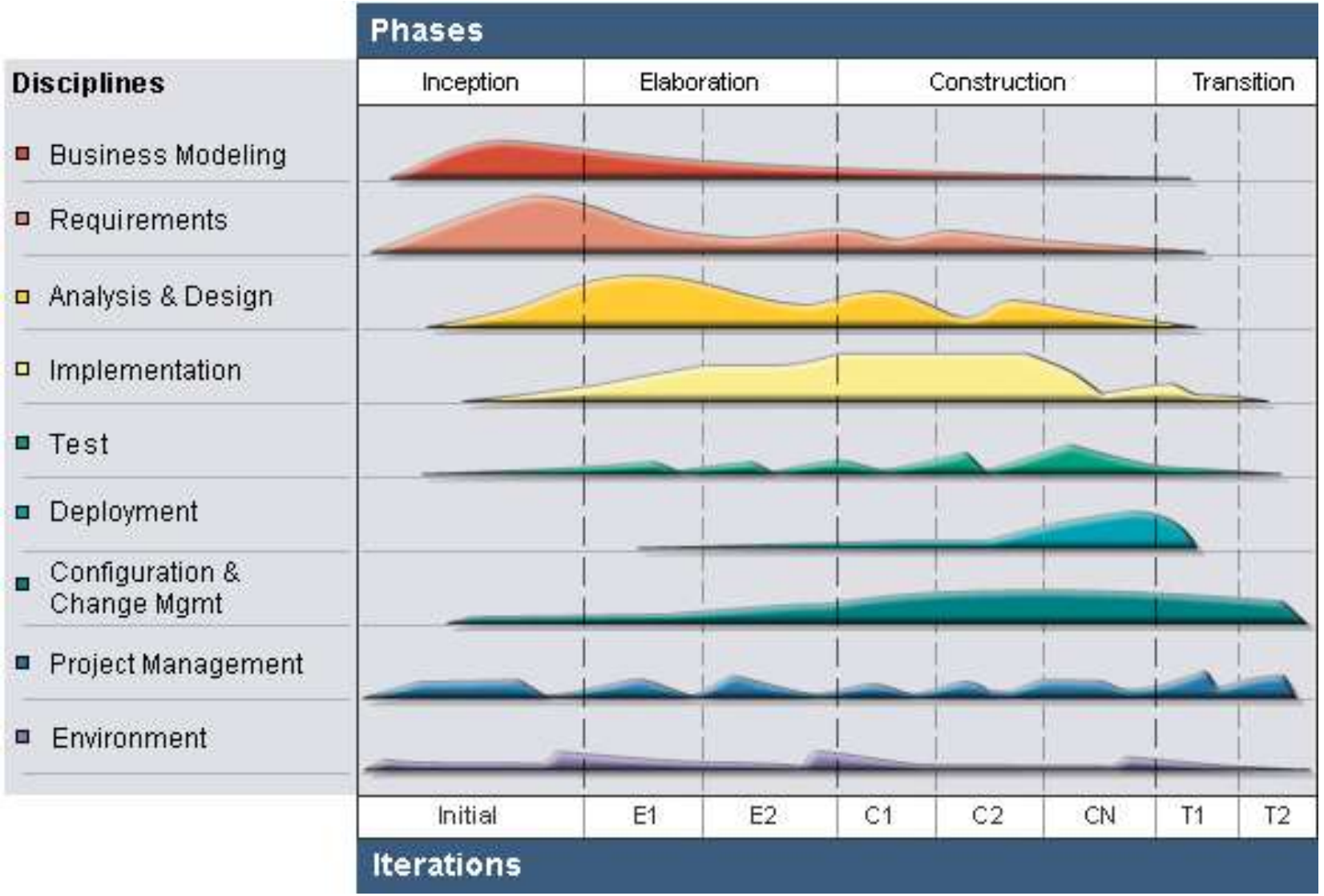
  - Big upfront architecture
    - Large Software Architecture Document (SAD)
  - Ivory tower architect
    - The architecture is not understood and implemented by the developers
    - The architect does not understand the current technology
  - The architect is only available in the beginning of the project

Ivory tower architect

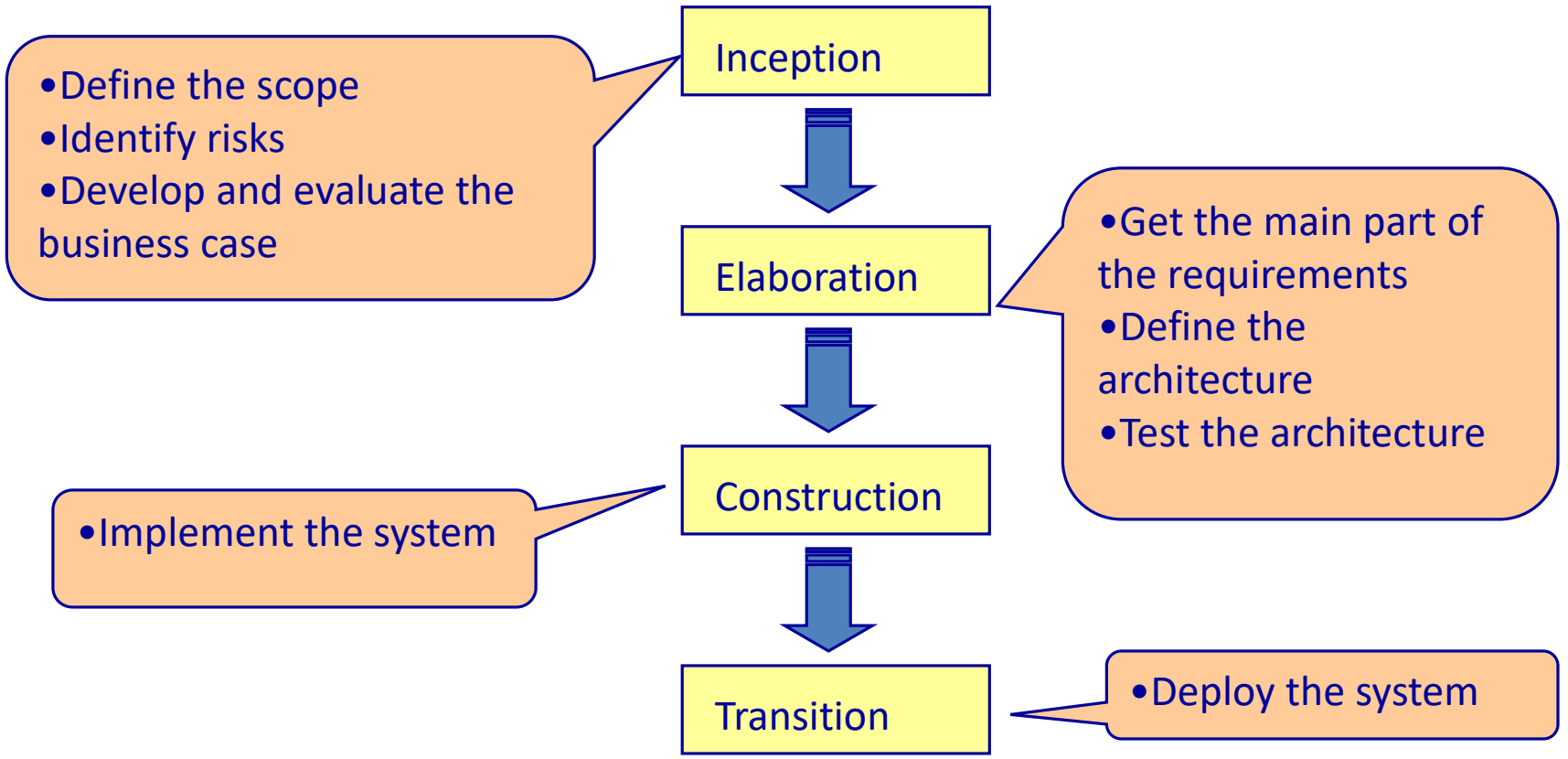
- It is very hard to truly know the best solutions for design problems if you are not working (coding) on the project
- It takes many iterations of a solution before it finally works - so you can’t suggest a solution and then leave



RUP

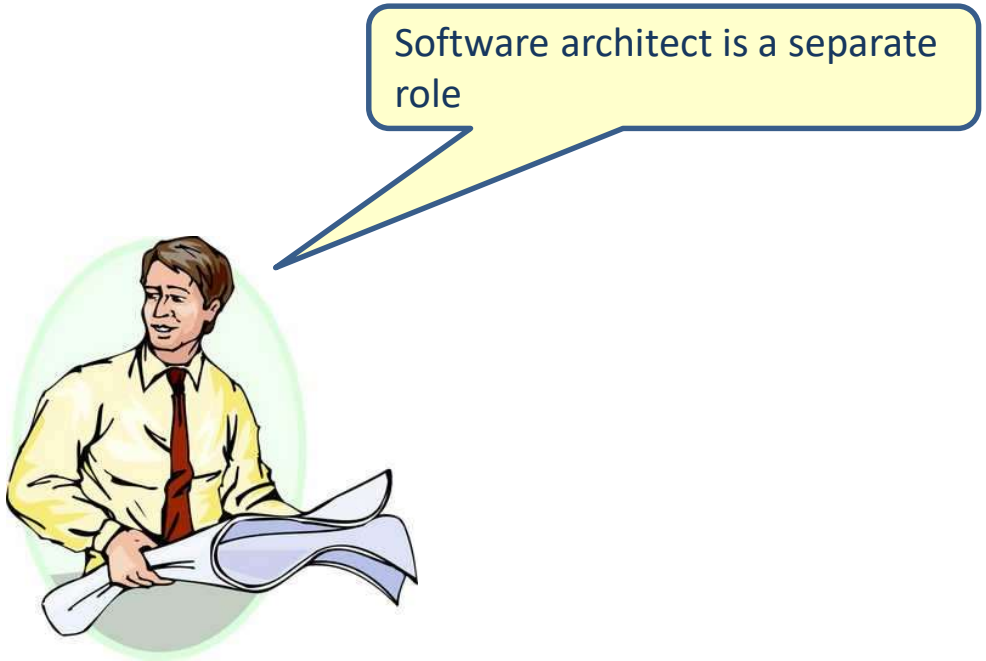


RUP phases



Core roles in RUP

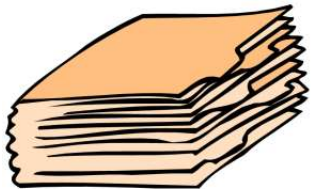
- Project manager
- Analyst
- Developer
- Tester
- Architect



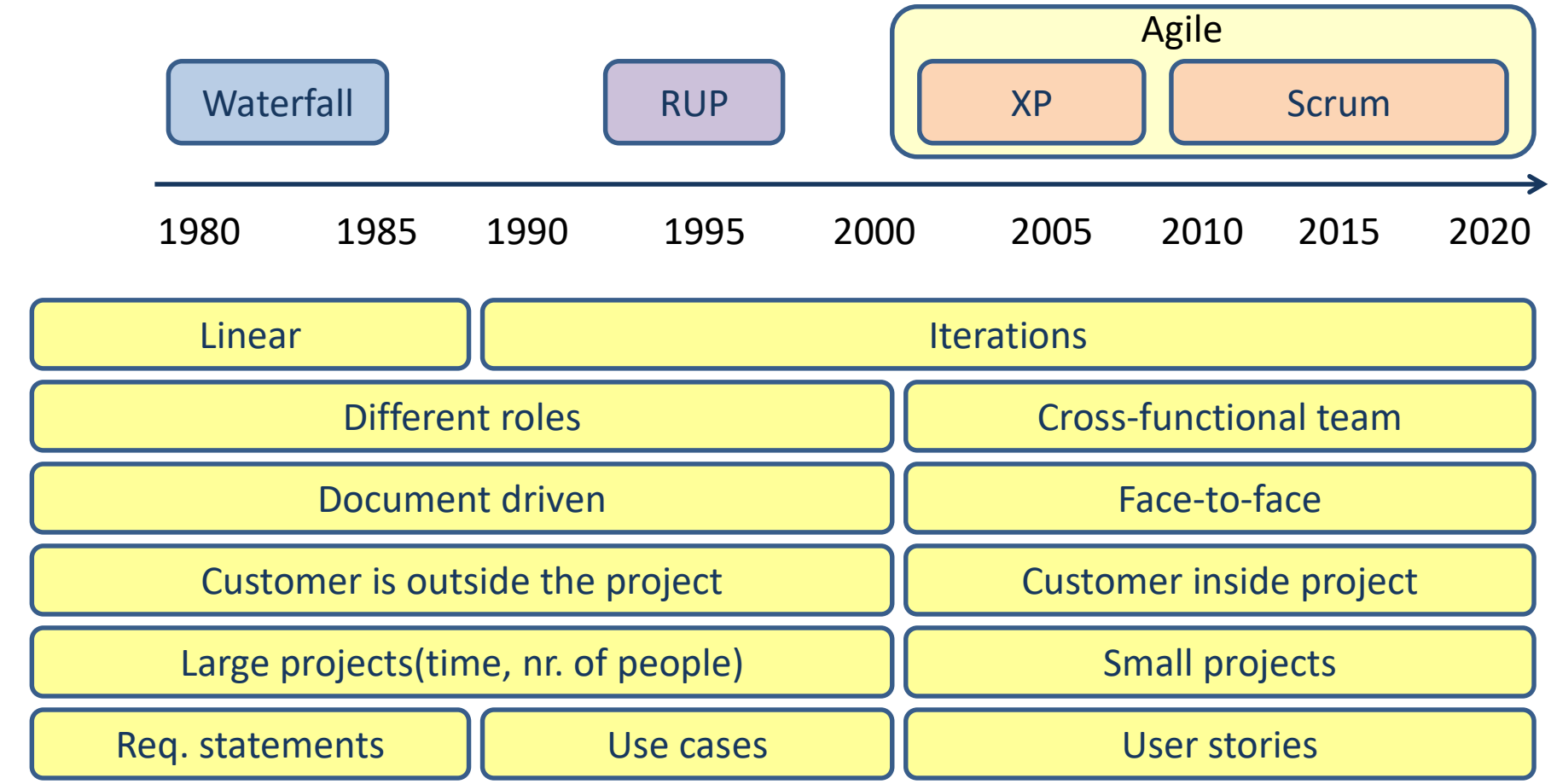
Software Architect

RUP software architecture

- The architect designs the system
  - Big upfront architecture
    - Large Software Architecture Document (SAD)
  - Ivory tower architect
    - The architecture is not understood and implemented by the developers
    - The architect does not understand the current technology
  - The architect is only available in the beginning of the project



Software development methods



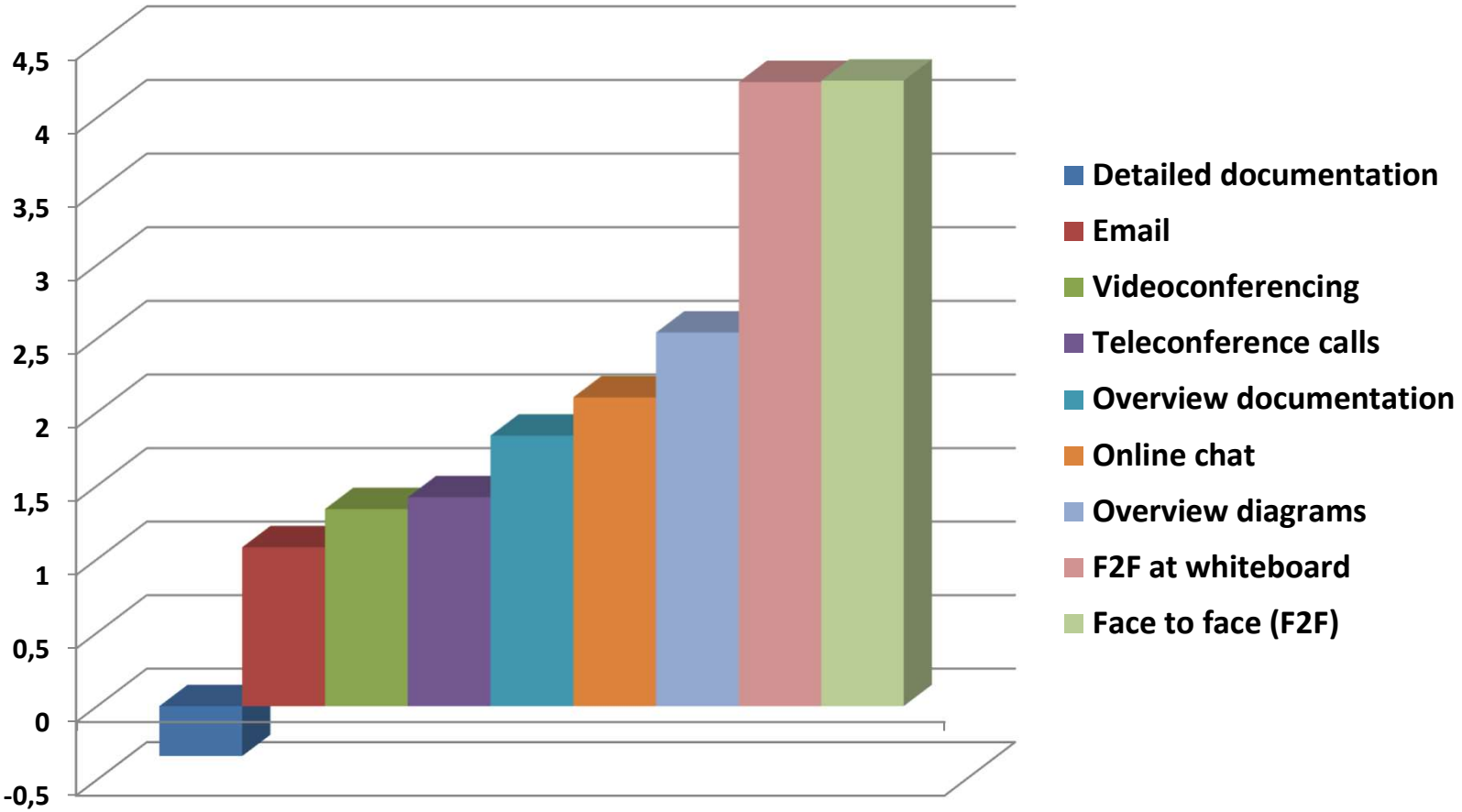
The Agile Manifesto

Individuals and interactions	over	Processes and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

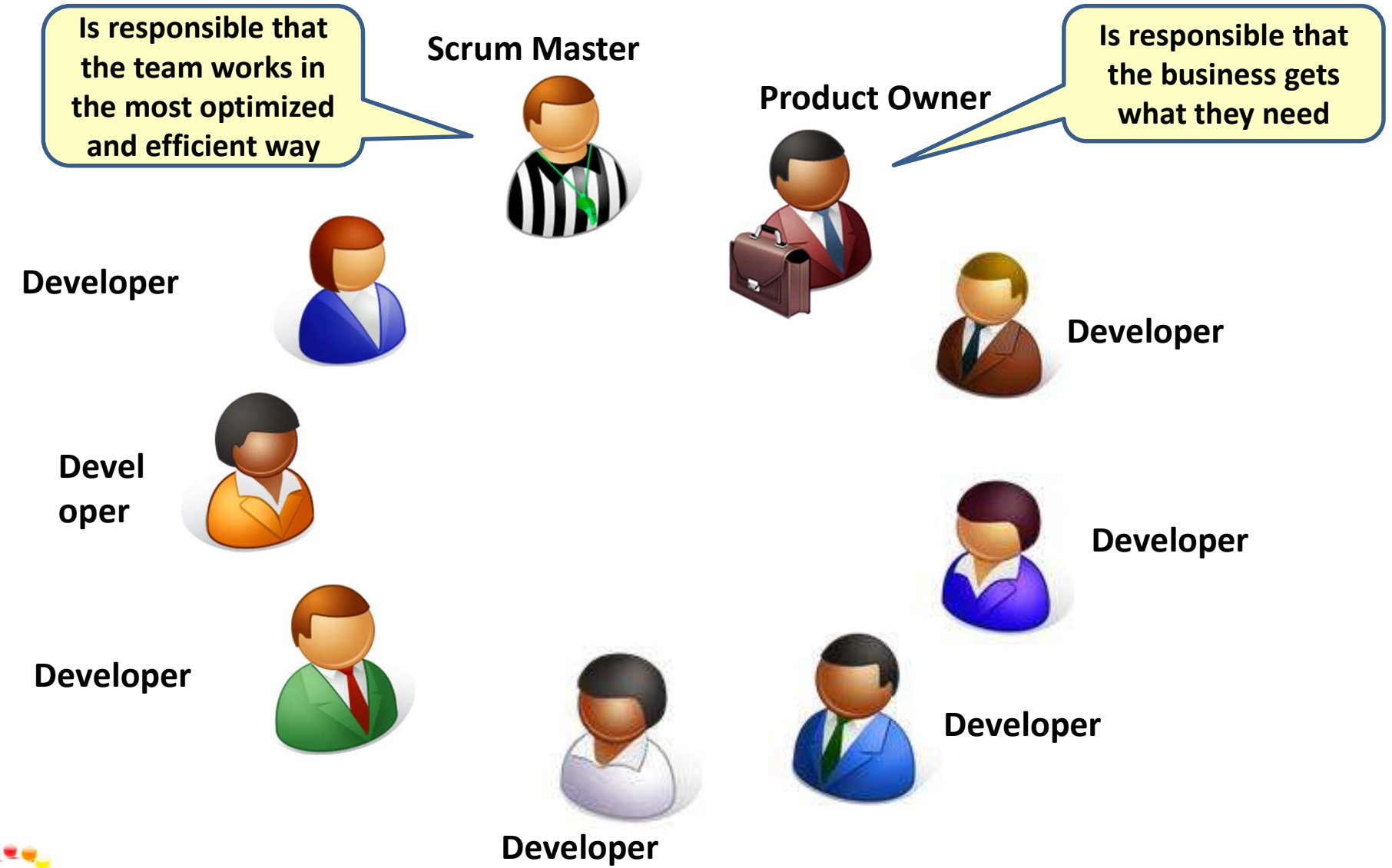
Agile principles

- Early and continuous delivery of valuable software.
- Welcome changing requirements.
- Business people and developers must work together daily.
- Give the team the environment and support they need, and trust them to get the job done.
- Prefer face-to-face conversation.
- Working software is the primary measure of progress.
- Continuous attention to technical excellence and good design
- Simplicity is essential.
- Self-organizing teams.

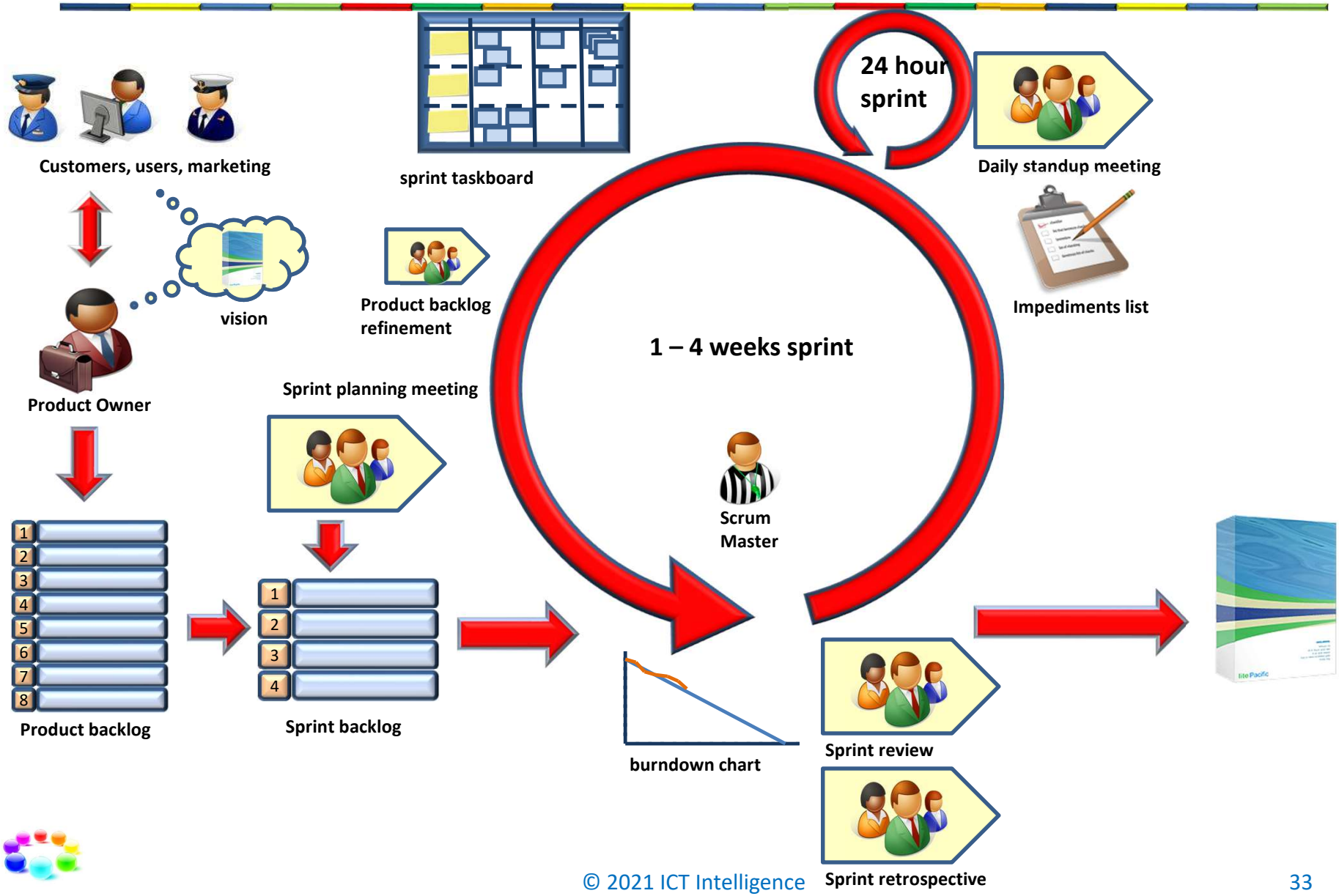
Effectiveness of communication



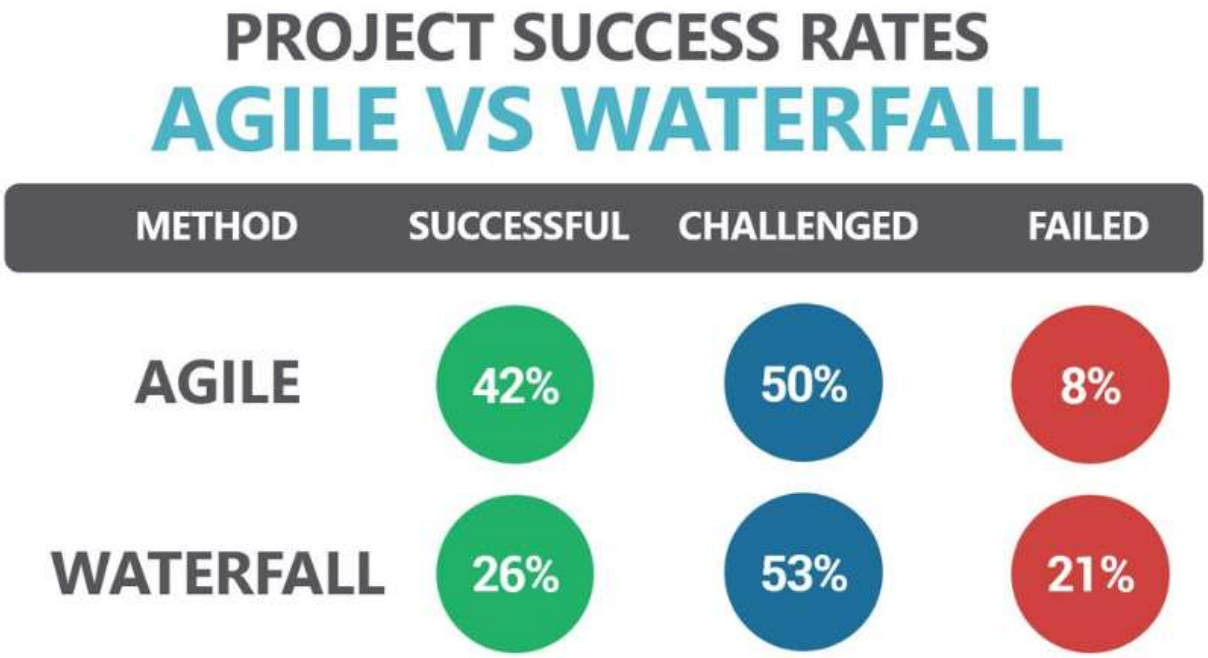
Scrum team



Scrum in action



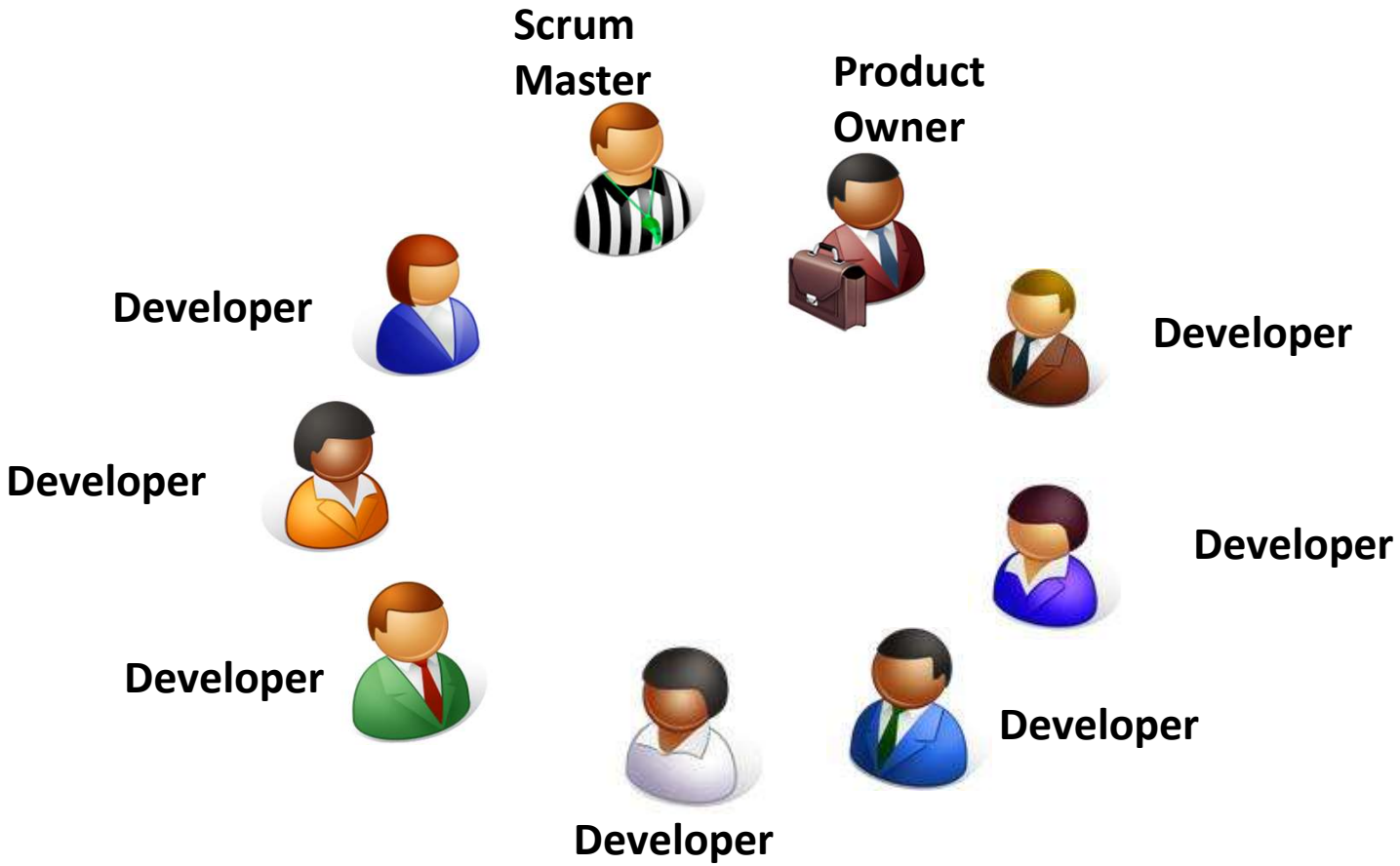
Comparing waterfall and agile



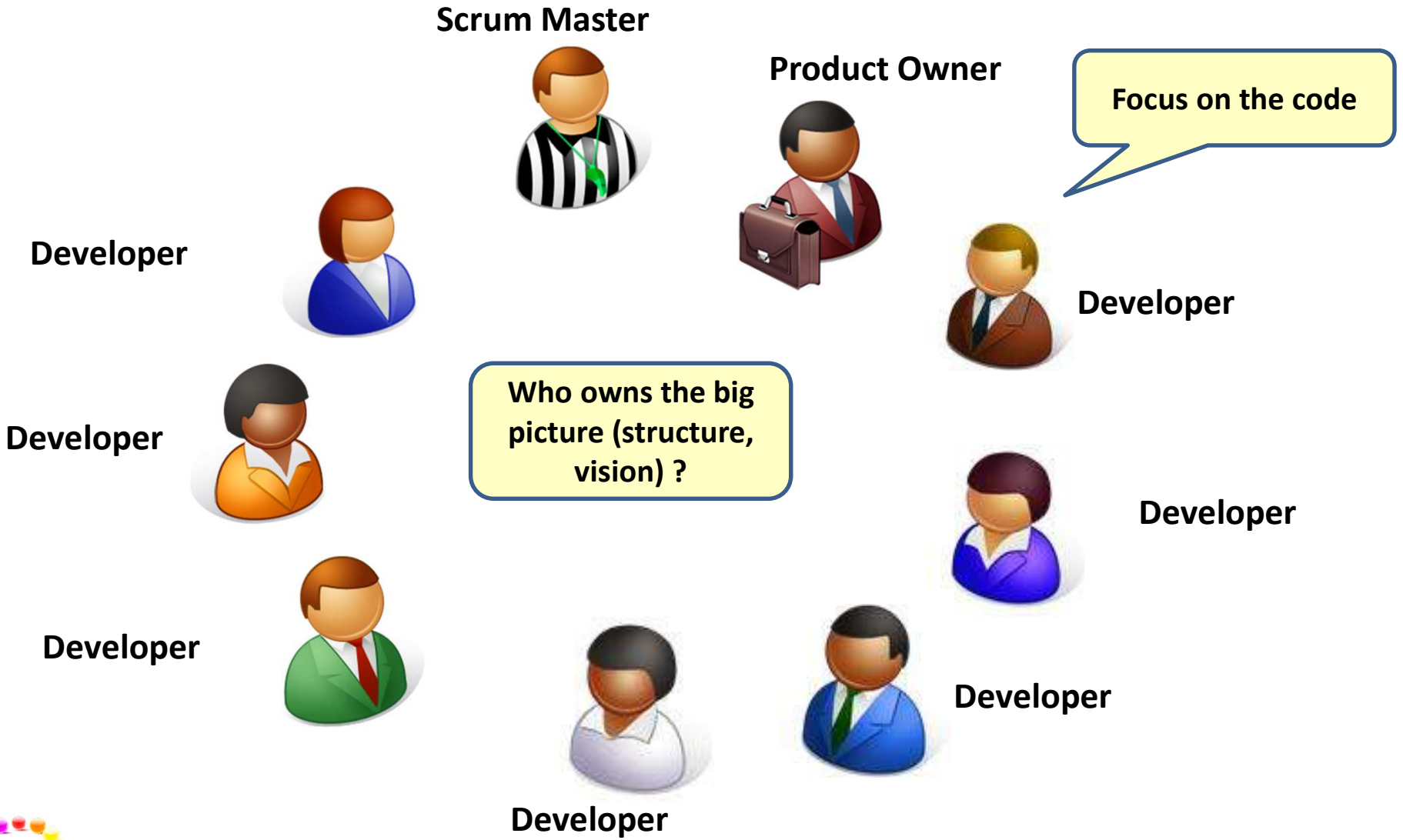
The chaos manifesto 2017

Agile architecture

- Architecture is a task, not a role
  - Team is responsible for architecture



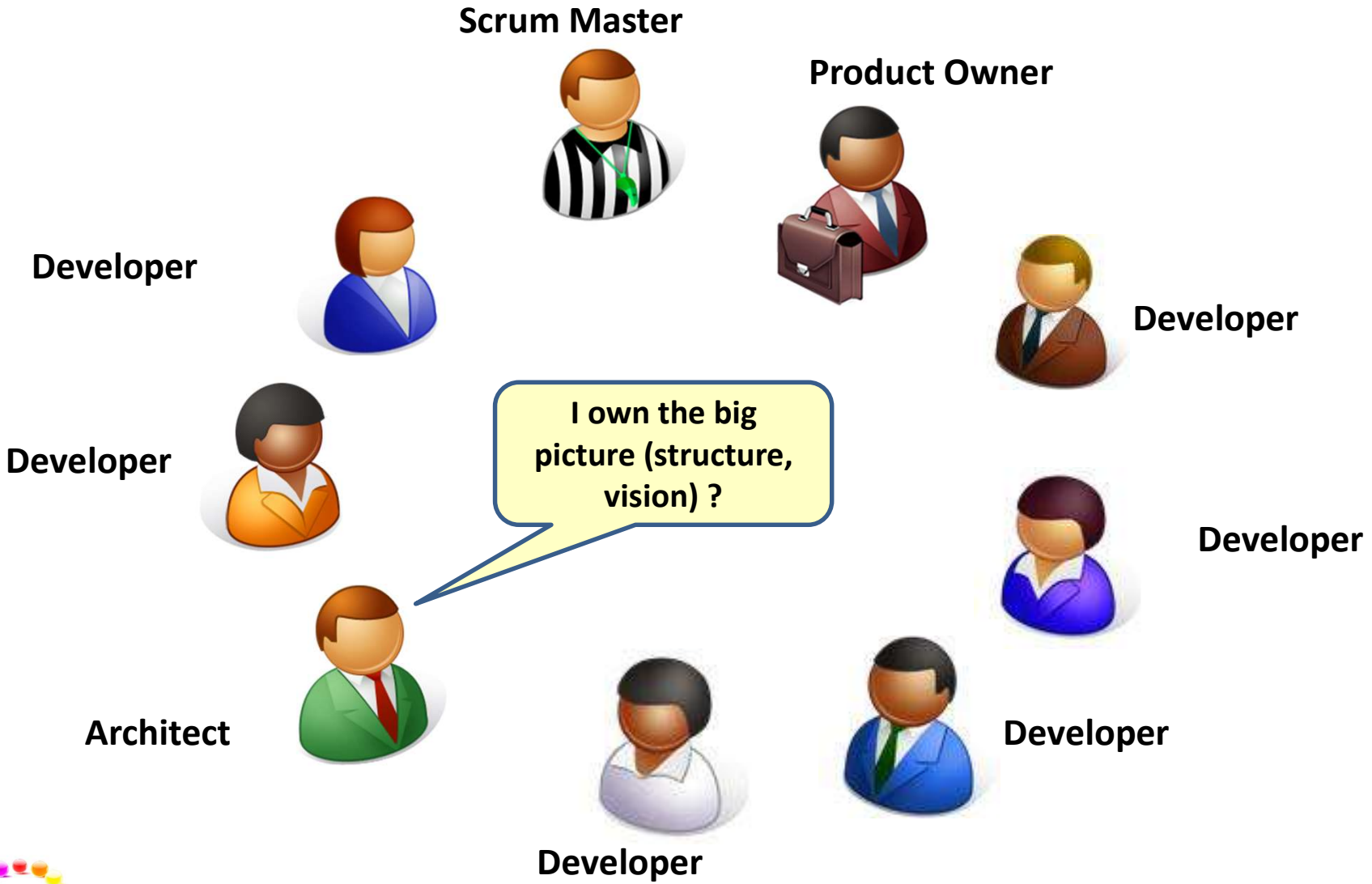
Scrum team


Scrum Master
Product Owner
Focus on the code
Developer
Developer
Developer
Who owns the big picture (structure, vision) ?
Developer
Developer
Developer
Developer

Why software architecture in agile?

- We need a clear vision and roadmap for the team to follow.
- We need to identifying and mitigating risk.
- We have to communicate our solution at different levels of abstraction to different audiences.
- We need technical leadership
- We have to make sure our architecture is consistent, correct and fits within the context

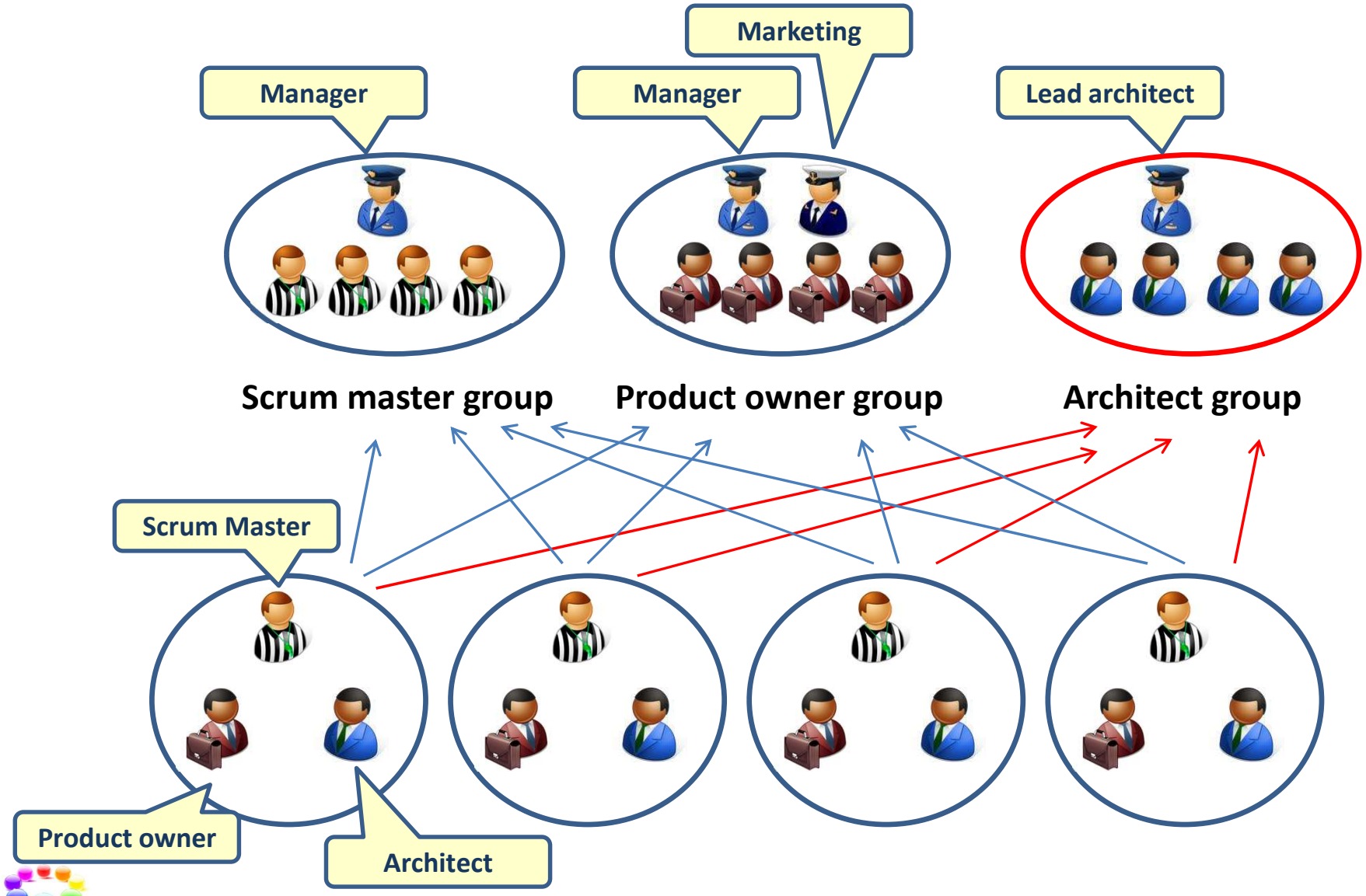
You need an architect



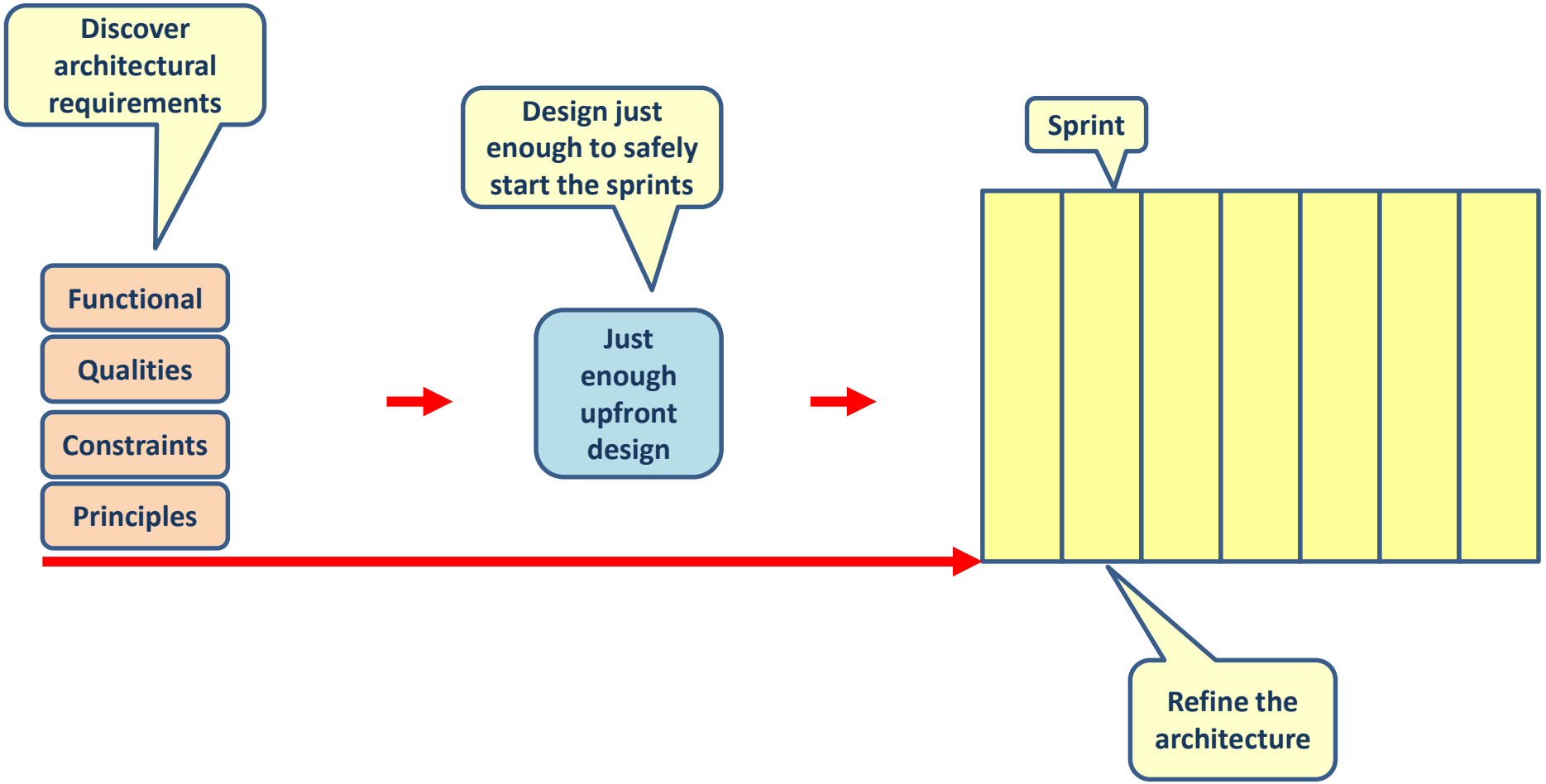
Agile architecture

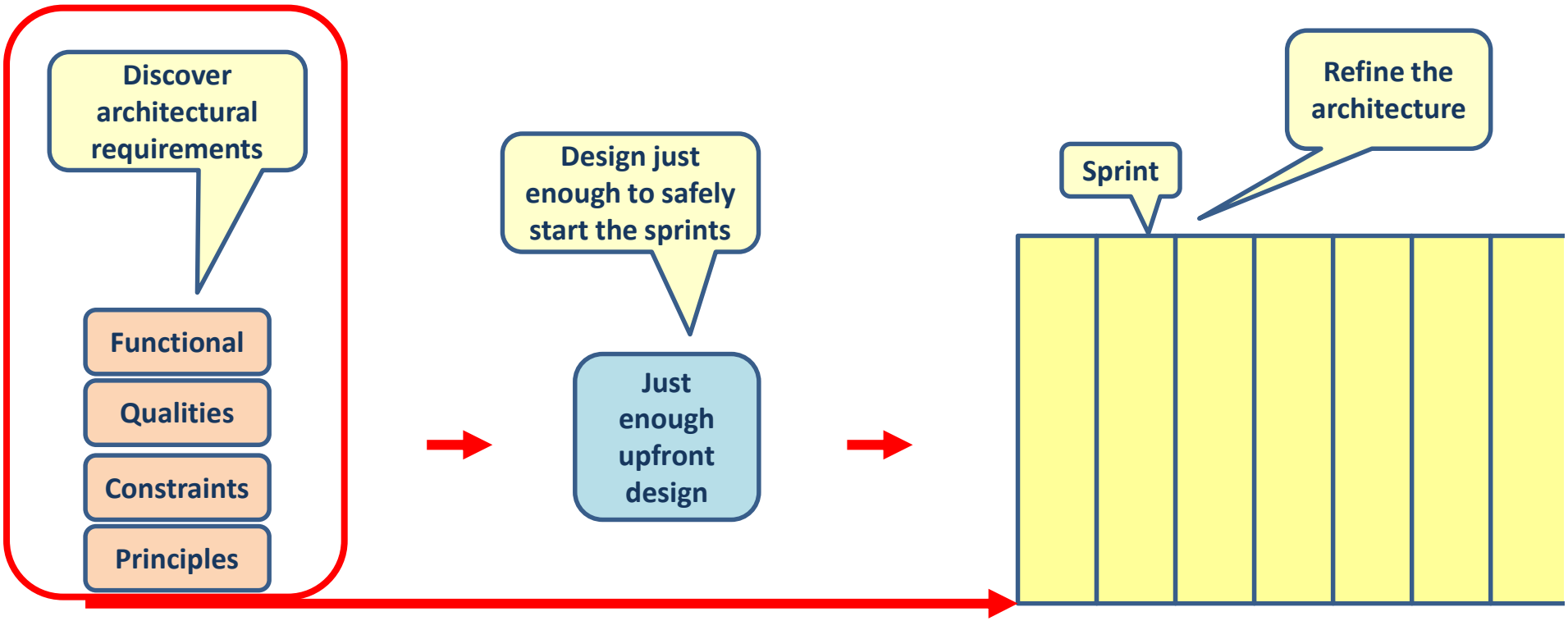
- Just enough upfront architecture
  - Refine the architecture later
- Keep your architecture flexible
- The architect is available during the whole project
- The architect also writes code
  - But not all the time
  - The architecture is grounded in reality
  - Works together with the developers
  - Architects should be master builders
- Proof the architecture in the first iteration(s)

Architect group



Agile architecture





ARCHITECTURAL REQUIREMENTS

Functional requirements

- What should the system functionally do?
  - Use cases
  - User stories

As a customer

I can view my account history

so that I know all transactions on my account

- **Functional**
- **Qualities**
- **Constraints**
- **Principles**

Architectural constraints

- Constraints from the business (or enterprise architecture
  - We do everything in .Net
  - We always use an Oracle database
  - Our maintenance engineers all know Java
  - All applications talk with the Oracle ESB
- Budget
- Deadlines

SOFTWARE QUALITIES

NFR characteristics

- Which qualities are available?
- We need to balance the qualities
- A quality itself is not precise enough
- Stakeholders have different interests

Wikipedia software qualities

- accessibility
- accountability
- accuracy
- adaptability
- administrability
- affordability
- agility (see Common subsets below)
- auditability
- autonomy [Erl]
- availability
- compatibility
- composability [Erl]
- configurability
- correctness
- credibility
- customizability
- debuggability
- degradability
- determinability
- demonstrability
- dependability (see Common subsets below)
- deployability
- discoverability [Erl]
- distributability
- durability
- effectiveness
- efficiency
- evolvability
- extensibility
- failure transparency
- fault-tolerance
- fidelity
- flexibility
- inspectability
- installability
- integrity
- interchangeability
- interoperability [Erl]
- learnability
- localizability
- maintainability
- manageability
- mobility
- modifiability
- modularity
- observability
- operability
- orthogonality
- portability
- precision
- predictability
- process capabilities
- producibility
- provability
- recoverability
- relevance
- reliability
- repeatability
- reproducibility
- resilience
- responsiveness
- reusability [Erl]
- robustness
- safety
- scalability
- seamlessness
- self-sustainability
- serviceability (a.k.a. supportability)
- securability (see Common subsets below)
- simplicity
- stability
- standards compliance
- survivability
- sustainability
- tailorability
- testability
- timeliness
- traceability
- transparency
- ubiquity
- understandability
- upgradability
- usability
- vulnerability

SEI quality model

Qualities noticeable at runtime

Performance	**Responsiveness of the system**
Security	**Ability to resist unauthorized usage**
Availability	**Portion of time the system is available**
Functionality	**Ability to do intended work**
Usability	**Learnability, efficiency, satisfaction, error handling, error avoidance**

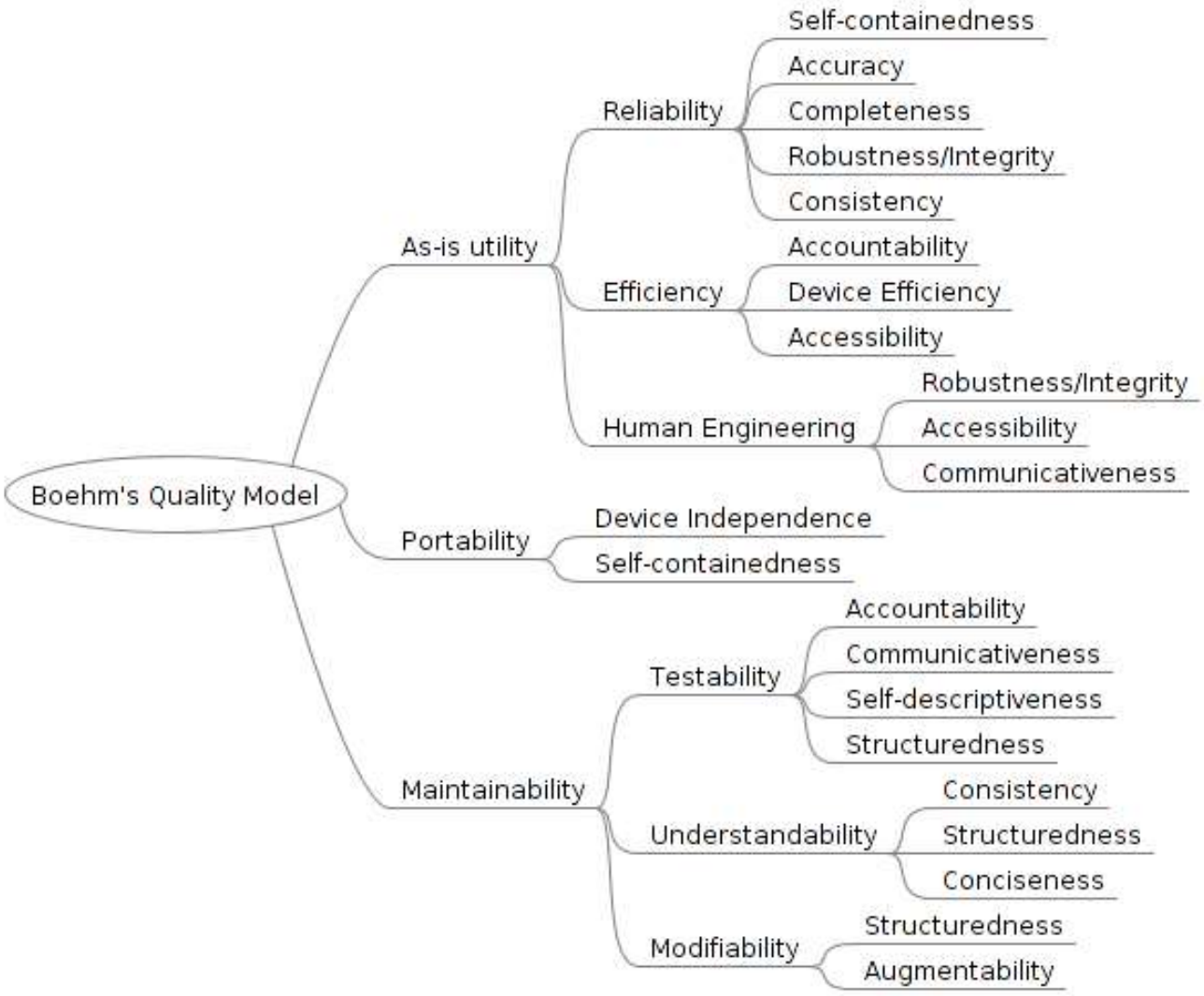
Qualities not noticeable at runtime

Modifiability	**Cost of introducing change**
Portability	**Ability to operate in different computing environments**
Reusability	**Ability to reuse components in different applications**
Integrability	**Ability that components work correctly together**
Testability	**Ability to systematic testing to discover defects**

FURPS model

- **Functionality** - evaluate the feature set and capabilities of the program, the generality of the functions delivered and the security of the overall system
- **Usability** - consider human factors, overall aesthetics, consistency, and documentation
- **Reliability** - measure the frequency and severity of failure, the accuracy of outputs, the ability to recover from failure, and the predictability
- **Performance** - measure the processing speed, response time, resource consumption, throughput and efficiency
- **Supportability** - measure the maintainability, testability, configurability and ease of installation

Boehm



ISO 25010

NFR characteristics

- Which qualities are available?
- We need to balance the qualities
- A quality itself is not precise enough
- Stakeholders have different interests

Balance the qualities

- More security through encription lowers performance
- More scalability through clustering lowers performance
- More scalability through clustering increases the cost

Find the top 5(+/- 2) qualitities

NFR characteristics

- Which qualities are available?
- We need to balance the qualities
- A quality itself is not precise enough
- Stakeholders have different interests

Quality scenario’s

- A quality on itself has little meaning
- Create scenario’s for the top qualitities
- Make scenario’s measurable
  - The should be able to scale to 1000 concurrent users
  - The system should be available 24/7
  - All user actions should give a response within 3 seconds.
- Prioritize the scenario's
- Write acceptance tests for NFR scenario’s

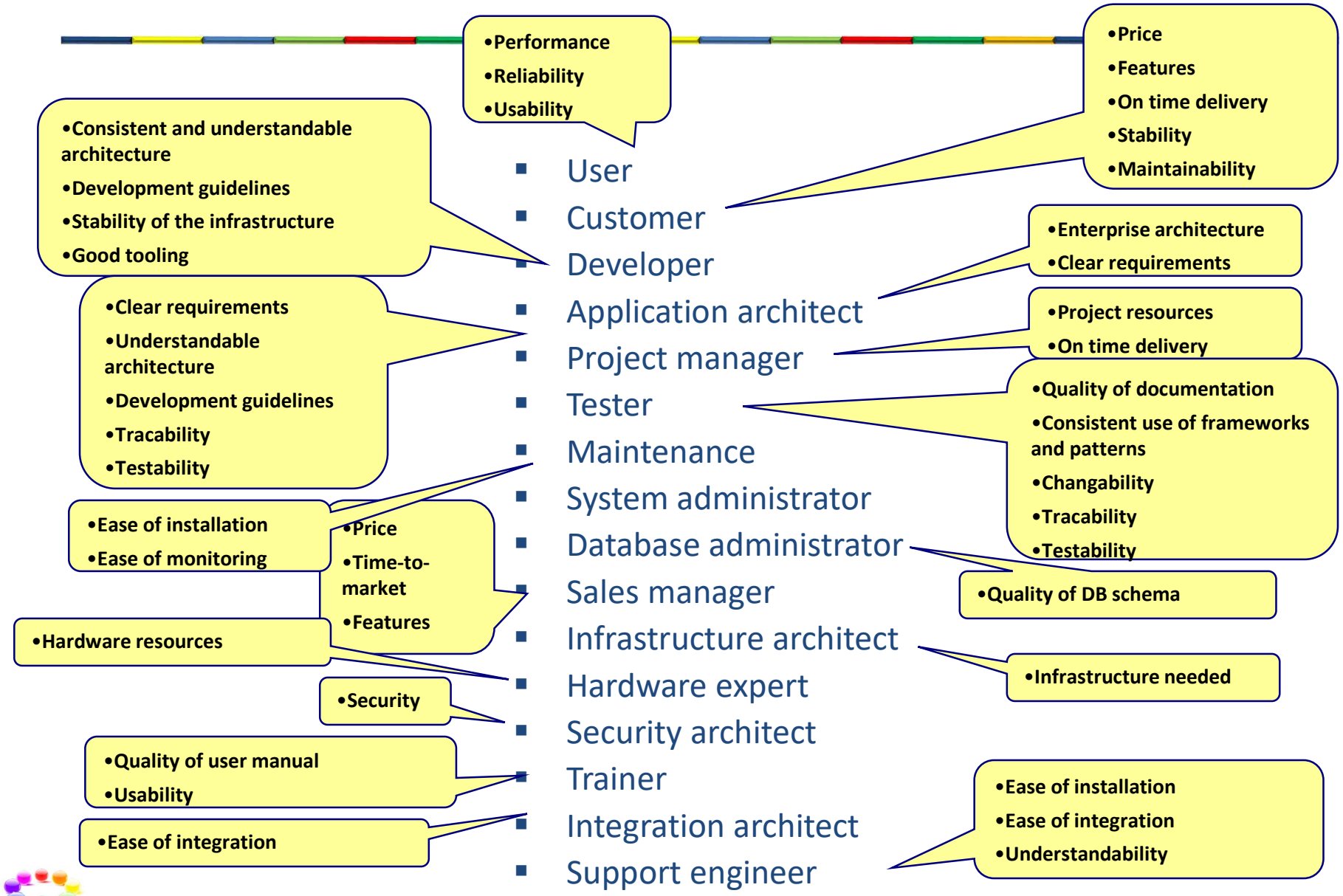
NFR characteristics

- Which qualities are available?
- We need to balance the qualities
- A quality itself is not precise enough
- Stakeholders have different interests

Stakeholders

- User
- Customer
- Developer
- Application architect
- Project manager
- Tester
- Maintenance
- System administrator
- Database administrator
- Sales
- Infrastructure architect
- Hardware expert
- Security architect
- Trainer
- Integration architect
- Support engineer

Stakeholders and their interest

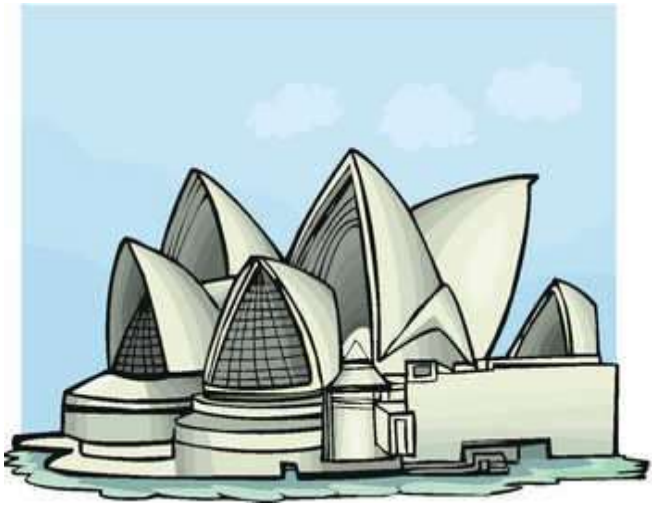


ARCHITECTURE PRINCIPLES

Architecture (design) principles

- Keep it simple
- Keep it flexible
- Loose coupling
  - High cohesion, low coupling
- Separation of concern
- Information hiding
- Principle of modularity
- Open-closed principle

Keep it simple



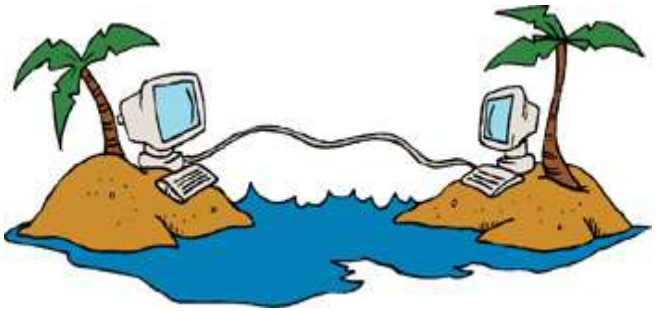
- The more complexity, the more change on failure
- Simple applications, frameworks, tools, etc. remain to be used
  - Complex ones will be replaced by something simple
- Gold plating

Keep it flexible

- Everthing changes
  - Business
  - Technical
- More flexibility leads to more complexity

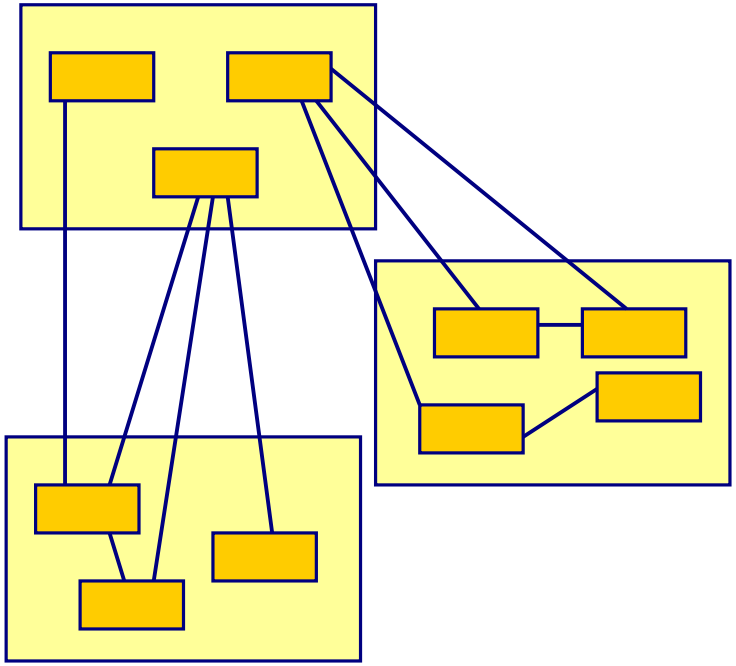
Loose coupling

- Different levels of coupling
  - Technology
  - Time
  - Location
  - Data structure
- You need coupling somewhere
  - Important is the level of coupling

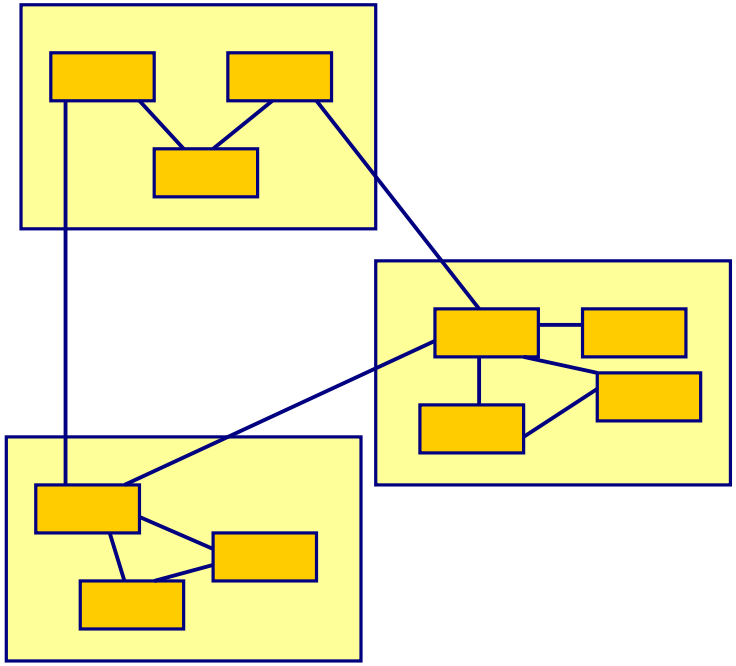


High cohesion, low coupling

- High coupling, low cohesion



- High cohesion, low coupling



Separation of concern

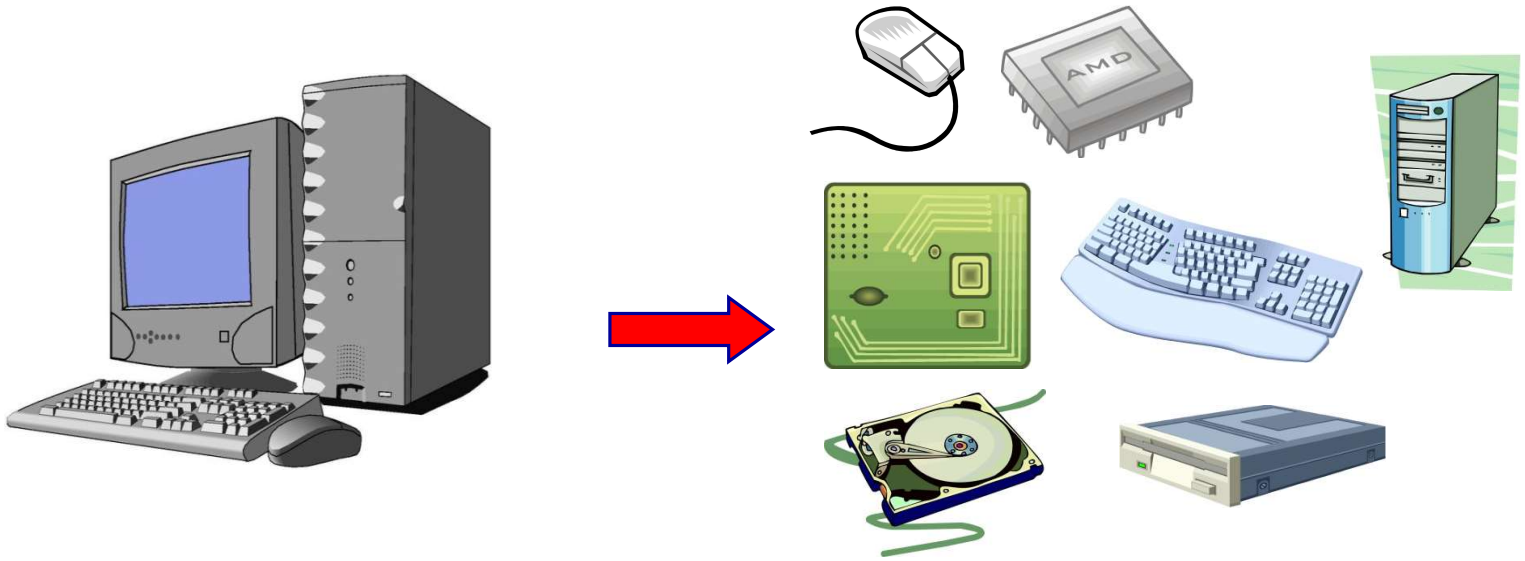
- Separate technology from business
- Separate stable things from changing things
- Separate things that need separate skills
- Separate business process from application logic
- Separate implementation from specification

Information hiding

- Black box principle
- Hide implementation behind an interface
- Hide the data structure behind stored procedures
- Hide the data structure behind business logic

Principle van modularity

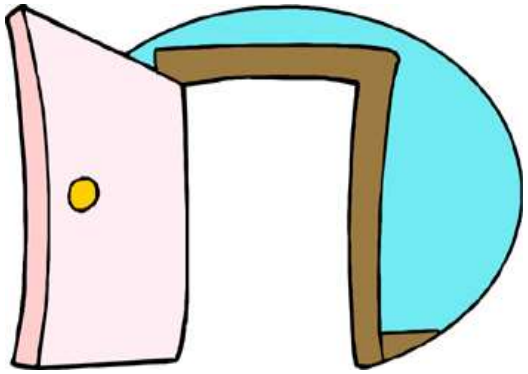
- Decomposition
- Devide a big complex problem is smaller parts
- Use components that are
  - Better understandable
  - Independent
  - Reusable
- Leads to more flexibility
- Makes finding and solvings bugs easier



Open- closed principle

- The design should be “open” for extension, but “closed” for change.

- You want to add new functionality instead of changing existing, working and tested code.

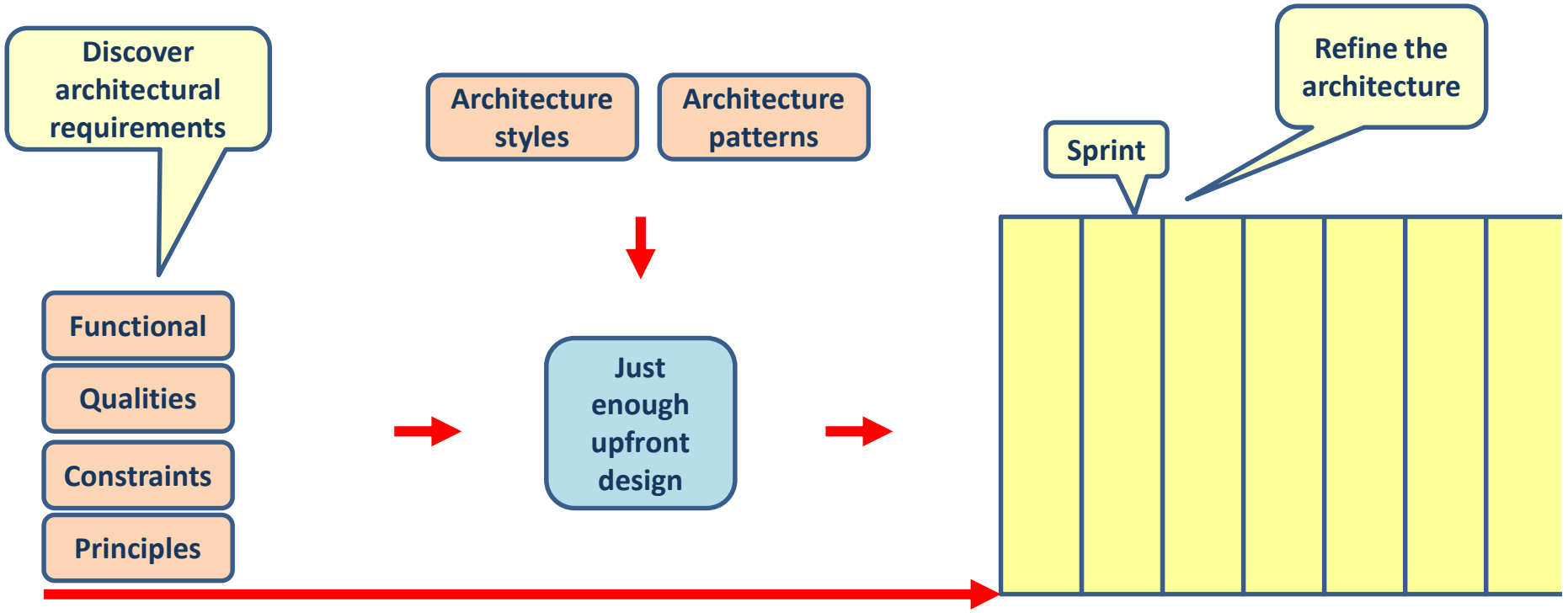


Most important architecture principles

- Keep it simple
- Keep it flexible
- Loose coupling
  - High cohesion, low coupling
- Separation of concern
- Information hiding
- Principle of modularity
- Open-closed principle

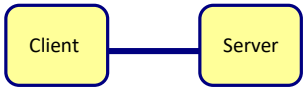
Main point

- Software architecture is never ideal. We have to find the right balance between the different software qualities and architecture principles
- Nature always takes the path of least resistance so that the perfection of the unified field can express itself in the relative creation

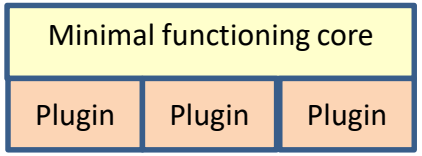


ARCHITECTURAL STYLES

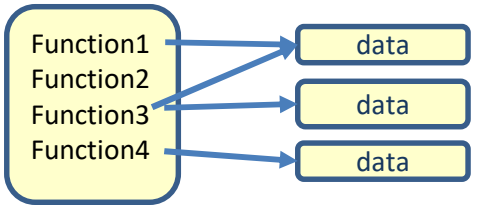
Architecture styles within an application



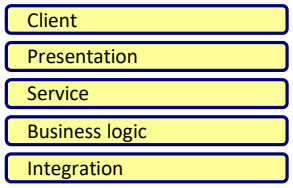
Client-server



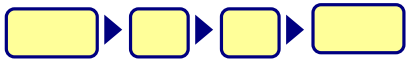
Microkernel



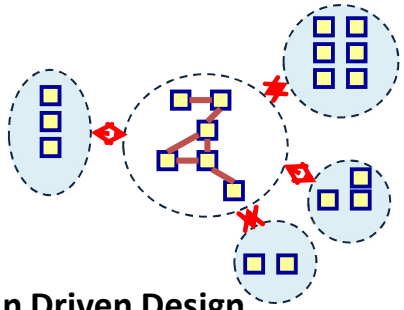
Procedural



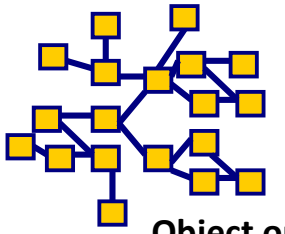
Layering



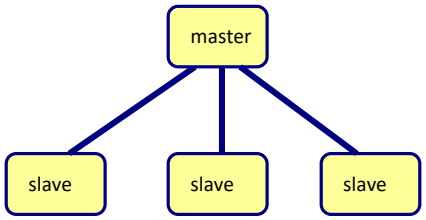
Pipe-and-Filter



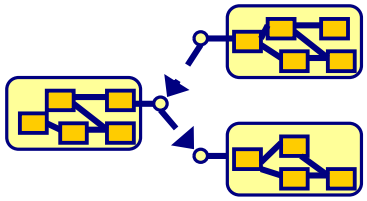
Domain Driven Design



Object oriented

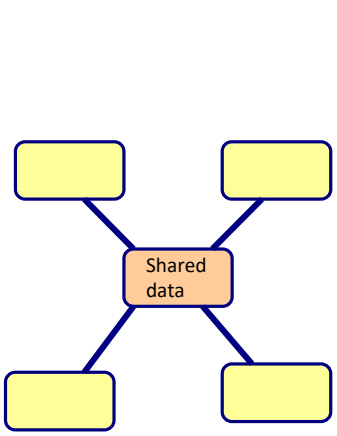


Master-Slave

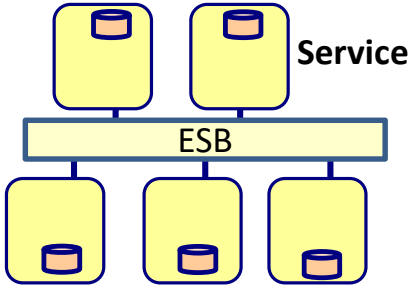


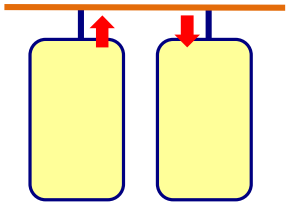
Component based

Architecture styles to connect applications

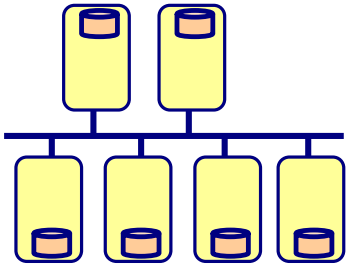


Blackboard

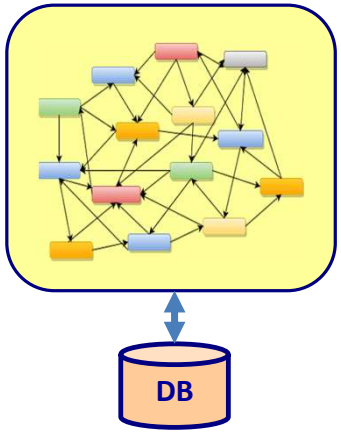




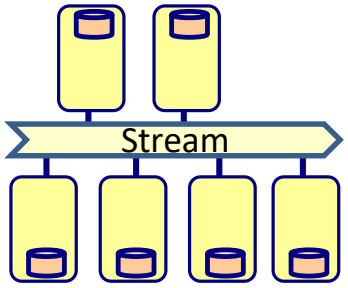
Event Driven



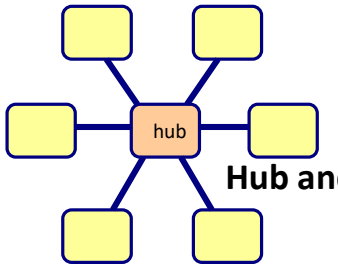
Microservices



Monolith



Stream based



Hub and Spoke

SUMMARY

What is software architecture?

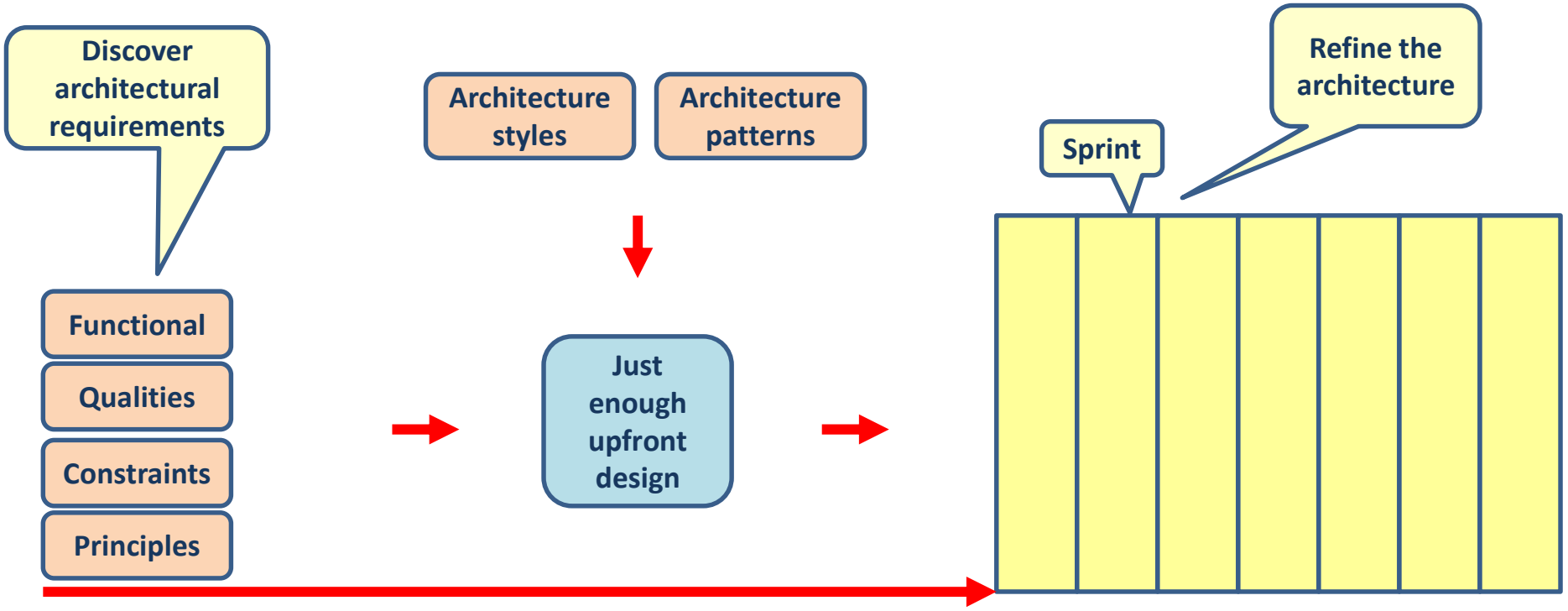
The important stuff

Whatever that might be

Agile architecture

- Just enough upfront architecture
  - Refine the architecture later
- Keep your architecture flexible
- The architect is available during the whole project
- The architect also writes code
  - But not all the time
  - The architecture is grounded in reality
  - Works together with the developers
  - Architects should be master builders
- Proof the architecture in the first iteration(s)

Agile software architecture



SOFTWARE ARCHITECTURE KEY PRINCILES

Key principle 1

- Software architecture is about making tradeoffs
  - Every decision has advantages and disadvantages
    - There is no silver bullet
    - The architecture is never ideal
  - You cannot read this is a book
    - There is no fixed template you can follow
  - The answer is always: It depends

Key principle 2

- The 2 most important quality attributes are
  1. Keep it simple
  2. Loose coupling

Connecting the parts of knowledge with the wholeness of knowledge

1. Software architecture defines all important aspects of a system.
2. The architecture decisions are based on the functional requirements, the qualities, the business constraints and the architecture principles

---

3. **Transcendental consciousness** is the natural experience pure consciousness, the home of all the laws of nature.
4. **Wholeness moving within itself:** In unity consciousness, one appreciates and enjoys the underlying blissful nature of life even in all the abstract expressions of pure consciousness.