Tampereen yliopisto

Large Scale Software Design

Introduction

Slides: Hannu-Matti Järvinen, David Hästbacka, …

Tampereen yliopisto

Introduction

- What is a software architecture?
  - Implement using P2P topology?
  - Use agile methods in development
  - Use a particular programming language as implementation technology
  - Use open source library OpenSSL
  - Publish using GPL license

What constitutes software architecture

- Basic structure of the software (components of the software)
- Relations between software components
- Relations between the software and its environment
- The principles guiding the design and evolution the software

Additional highlights on SW architecture

- Architecture is not a static structure - It contains also functionalities and dynamic structures of the software.
- Architecture is not only the structure and their relations, but it gives also reasons and justifications for them.
- There may be rules and principles how to develop systems using a given architecture.

Architecture definitions in standards

- The fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. (IEEE Standard 1471-2000)

- The fundamental conception of a system in its environment embodied in elements, their relationships to each other and to the environment, and the principles guiding its design and evolution. (ISO/IEC 42010)

Other definitions

- Barry Boehm's definition on what software system architecture comprises:
  - a collection of software and system components, connections, and constraints
  - a collection of system requirements
  - a rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system requirements
- D'Souza & Wills Architecture as a design principle:

  "Architecture is the set of design decisions about any system that keeps its implementers and maintainers from exercising needless creativity."
- Architecture = the law of the system
  - Given fundamental solutions, How to use given technology, How to use given data structures, How to use design patterns, How components communicate with each other, Rules to handle exceptions, Etc.

Does every software have an architecture

- Even though every system has an architecture it may not be explicit and it does not necessarily follow the architecture known
- An architecture can exist independently of its description or specification, which raises the importance of architecture documentation and architecture reconstruction.

Two different architecture views

- Defined architecture
  - Definition of the system, specification, <u>will of the designer</u>
- Architecture of a specified system
  - Property of the system, <u>defined by the system</u>
- Problems:
  - Only in ideal world these are the same thing
  - It is seldom defined which one is meant
  - Defined architecture not documented
  - Architecture of an implementation is hard to see

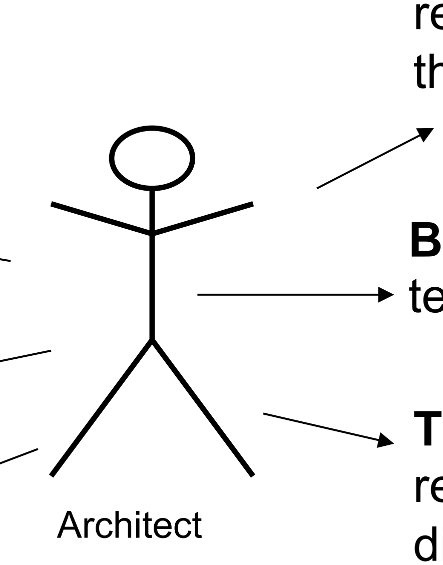
Things affecting the decisions of an architect

Resources: “I have two Scrum teams”

Experience: “I have had bad experiences using Struts”

Personal preference or company policy: “I prefer to use open source components over commercial ones”

..**and many other concerns.**



Requirements: “The system should have response time under 1 s when there are 10000 users logged in the service.

Business goals: “Company should take technology X in use within two years.”

Time-to-market: “Product should be released next week, I just do this quick-n-dirty solution”

Software architecture and quality requirements

The architecture is defined mostly by its quality requirements, not by functional requirements.

Architecture is a way to take into account quality requirements in software development process.

Here quality refers to the quality how the systems performs its logical functions

Eg. Response time with normal load is 5 ms, components should be exchangeable, …

Quality requirements

Any system can be implemented by any ”architecture”, if only functionality is concerned. From the point of view of logical functionality, the architecture is not very significant.

Why is the software architecture important 1?

It gives abstraction level for solving the main problems associated with the development of the system.

It is the central mean of communication during the lifespan of the software: it defines and names the main components, solutions and concepts.

Why is the software architecture important 2?

Architecture sets limits and eases (or makes possible) to build the system, test it, maintain it and reuse.

The first representation of the system that can be analysed: the system can be ”tested” before its implementation.

Pause for a second...

What consequences can result from bad architectural decisions?

Consequences of failed architecture

The system can’t be implemented

The system is not finished in time

The system does not scale

The system is powerless

The system is hard to test and maintain

The system can’t be reused

The system can’t be moved to another environment

Reasons for a failed architecture

Bad communication

Essential requirements have been neglected

The architect is inexperienced or weak-willed

Development process does not support the architecture

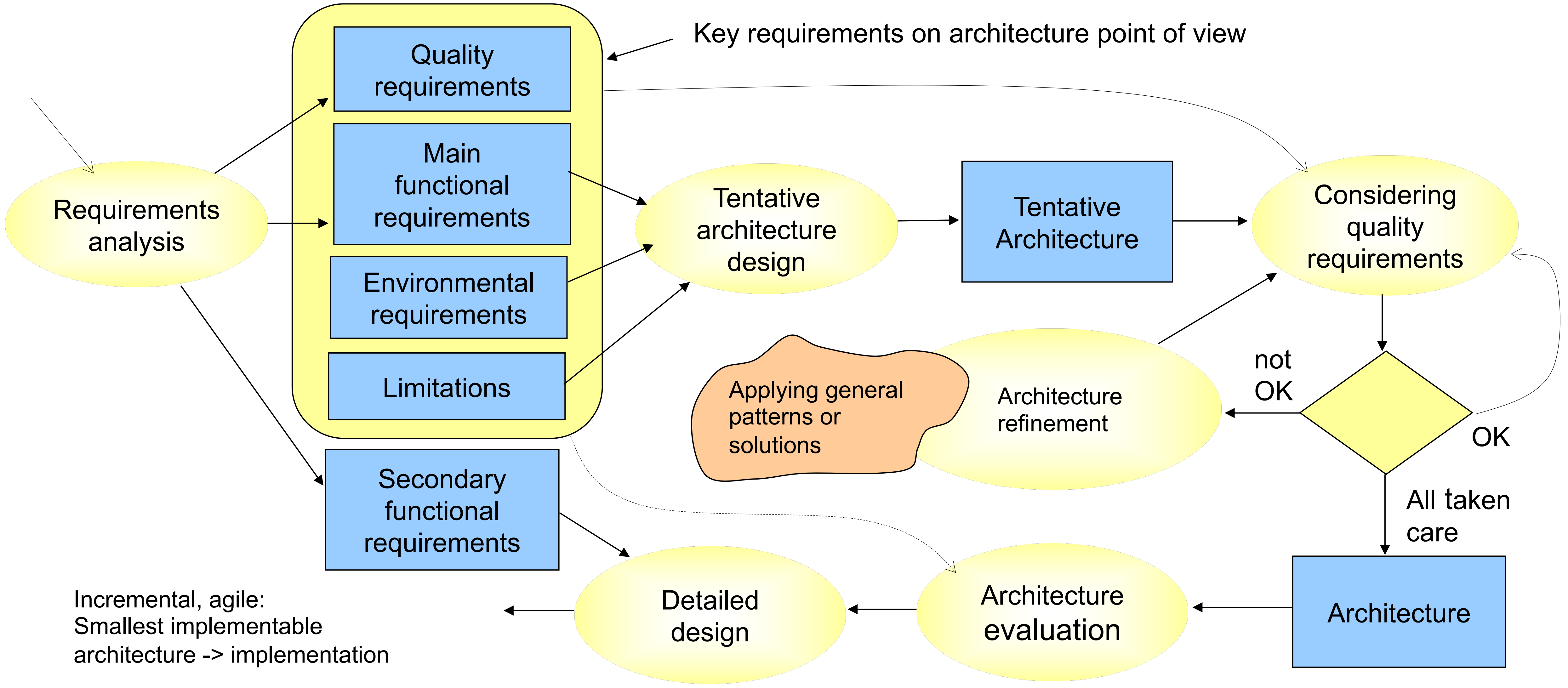
The architect does not know the target subject

Others?

Software architecture and software engineering process

The next section ties the software architecture design to software engineering process

Developing the architecture


Key requirements on architecture point of view
Quality requirements
Main functional requirements
Environmental requirements
Limitations
Requirements analysis
Secondary functional requirements
Tentative architecture design
Tentative Architecture
Considering quality requirements
Applying general patterns or solutions
Architecture refinement
not OK
OK
All taken care
Architecture
Architecture evaluation
Detailed design

Incremental, agile:
Smallest implementable
architecture -> implementation

Main requirements affecting the architecture 1

Main functional requirements (what to do)

- **Often the starting point for designing the architecture**

Quality requirements (how is it done)

- **Typically great influence on the architecture (non-functional requirements)**

Tampereen yliopisto

Main requirements affecting the architecture 2

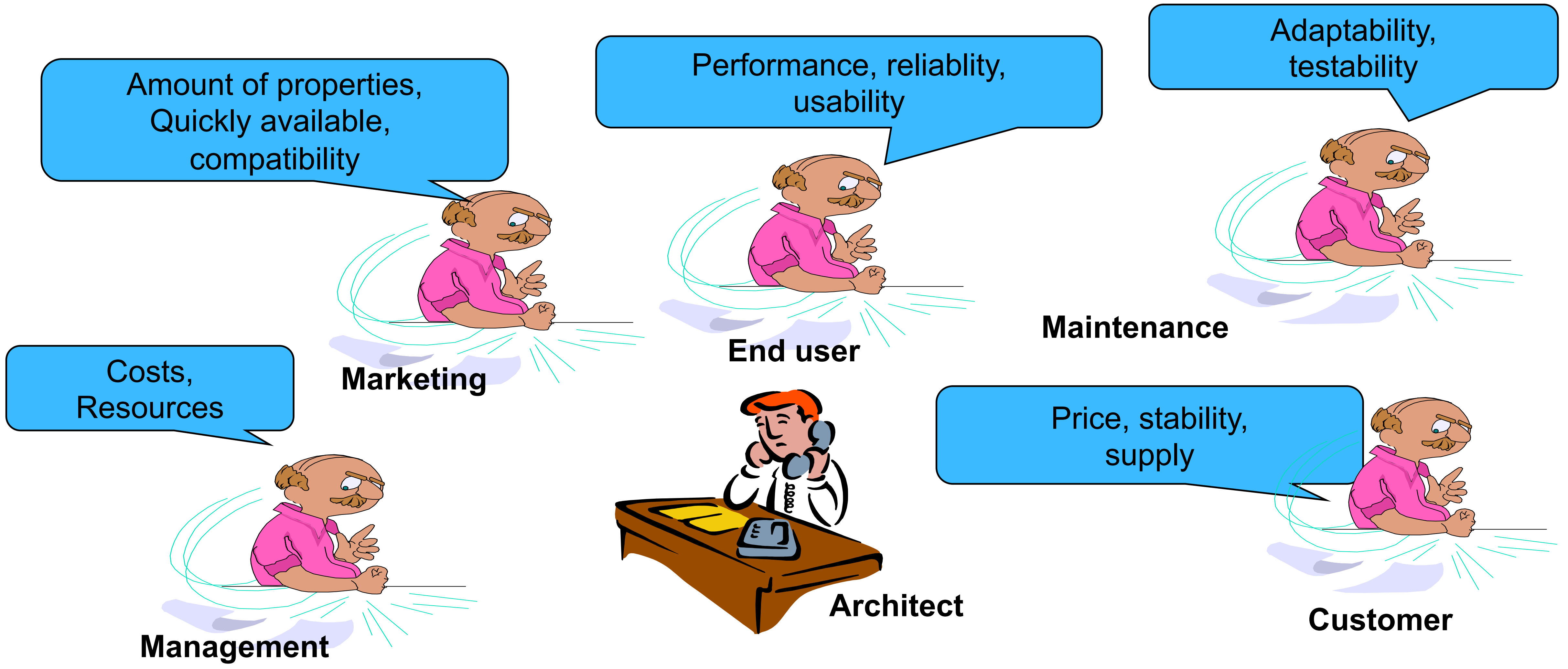
Environmental requirements

- The development environment
  - Can it be done by the available tools
  - Distributed development?
- The execution environment of the system
  - Devices of an embedded system (differ from development environment)

Limitations

- E.g. the used technology

Conflicting quality requirements


Amount of properties, Quickly available, compatibility
Marketing
Performance, reliablity, usability
End user
Adaptability, testability
Maintenance
Costs, Resources
Management
Architect
Price, stability, supply
Customer

Tampereen yliopisto

Architecture evaluation

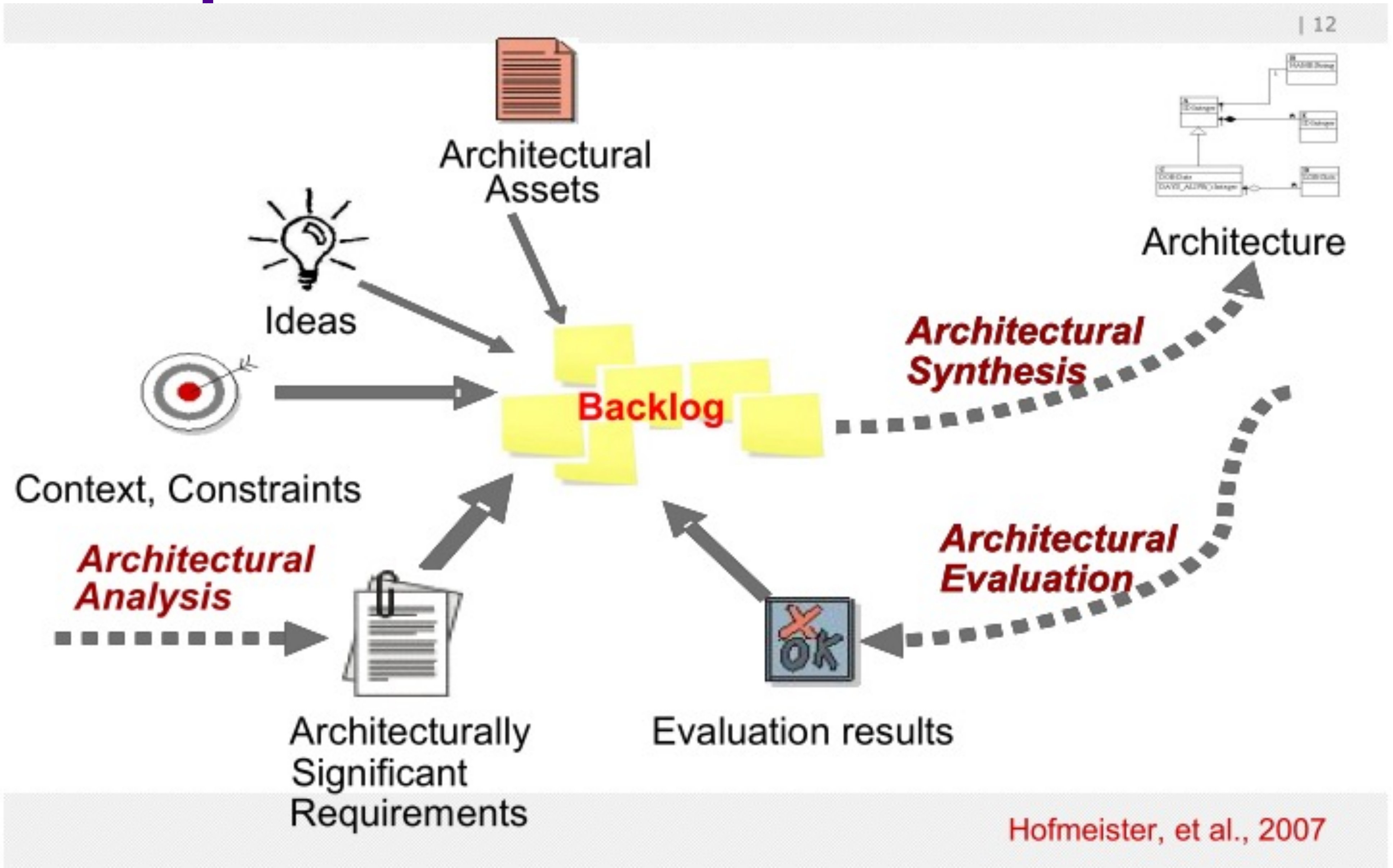
Many of the software quality properties could be deduced from the architecture description

- e.g. strict tier architecture -> performance problems

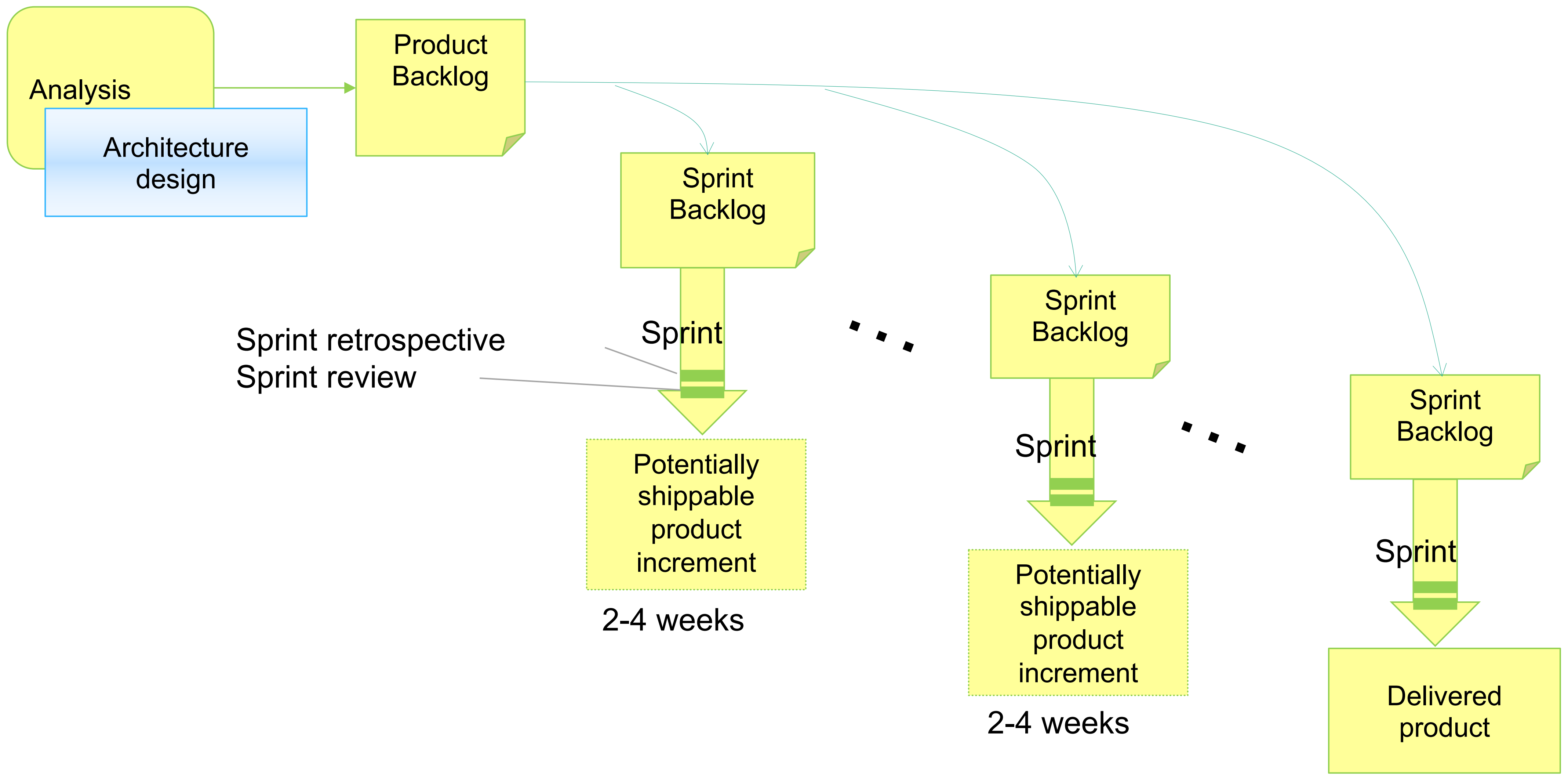
Architecture evaluation = refinement of quality requirements + evaluation against the refined requirements (e.g. ATAM)

Evaluation of the architecture is testing the software using its first precise description

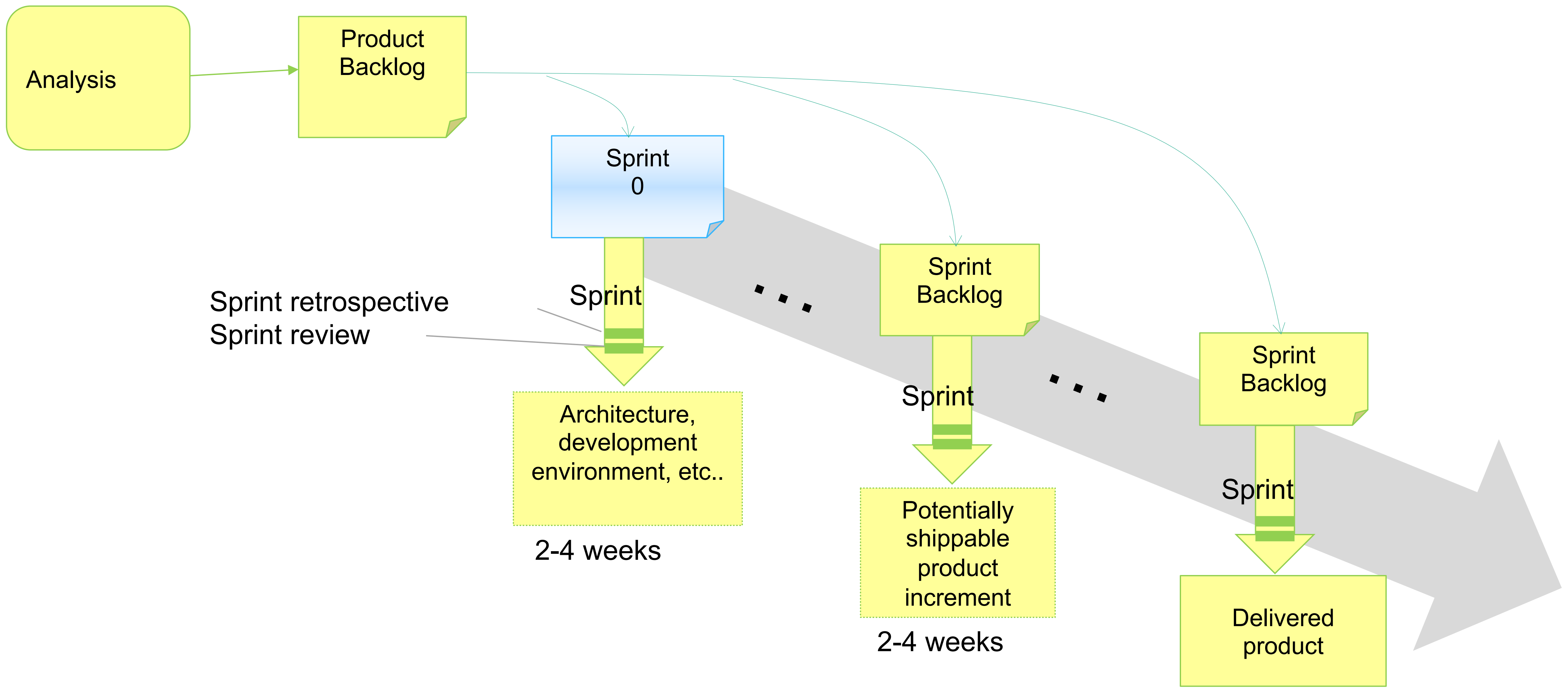
Development of architecture



Making architecture up-front

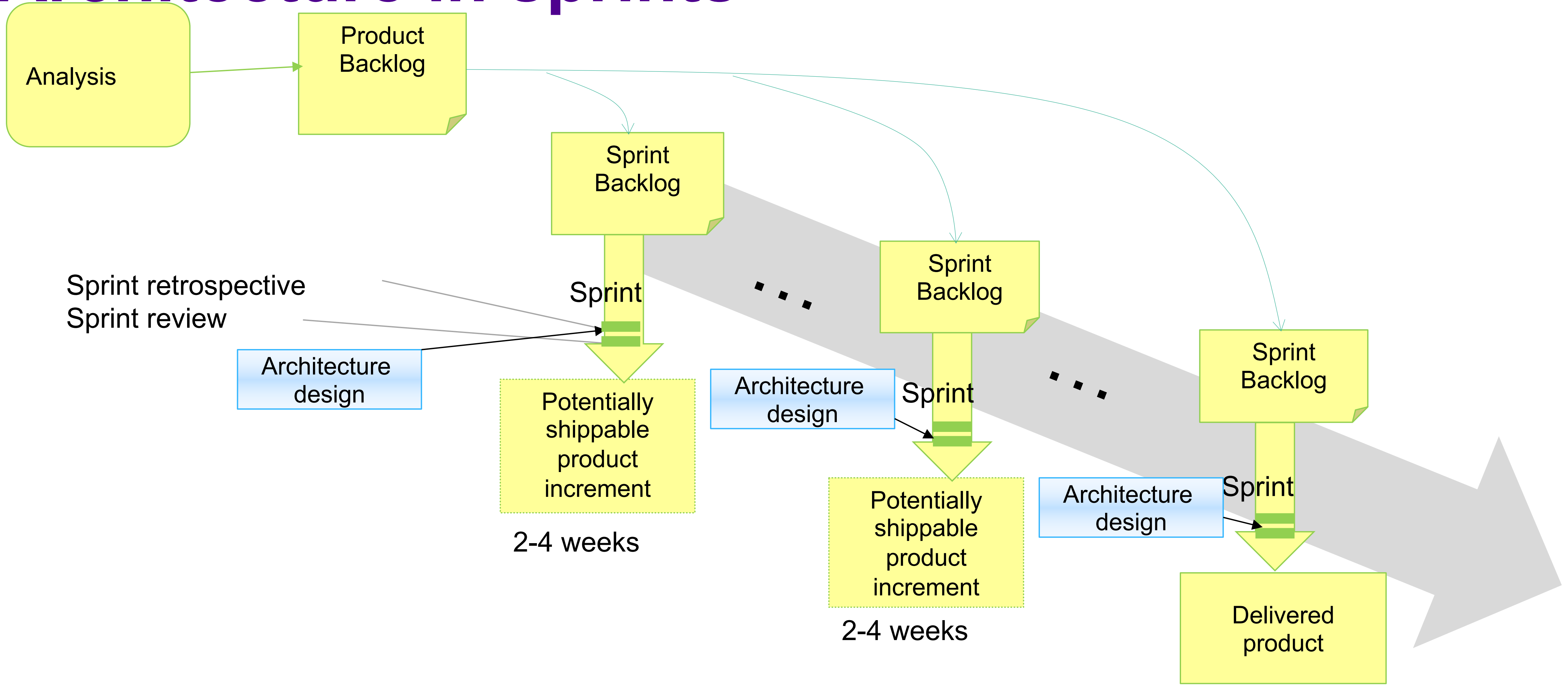


Sprint 0


Analysis
Product Backlog
Sprint 0
Sprint
Sprint retrospective
Sprint review
Architecture, development environment, etc..
2-4 weeks
. . .
Sprint Backlog
Sprint
Potentially shippable product increment
2-4 weeks
. . .
Sprint Backlog
Sprint
Delivered product

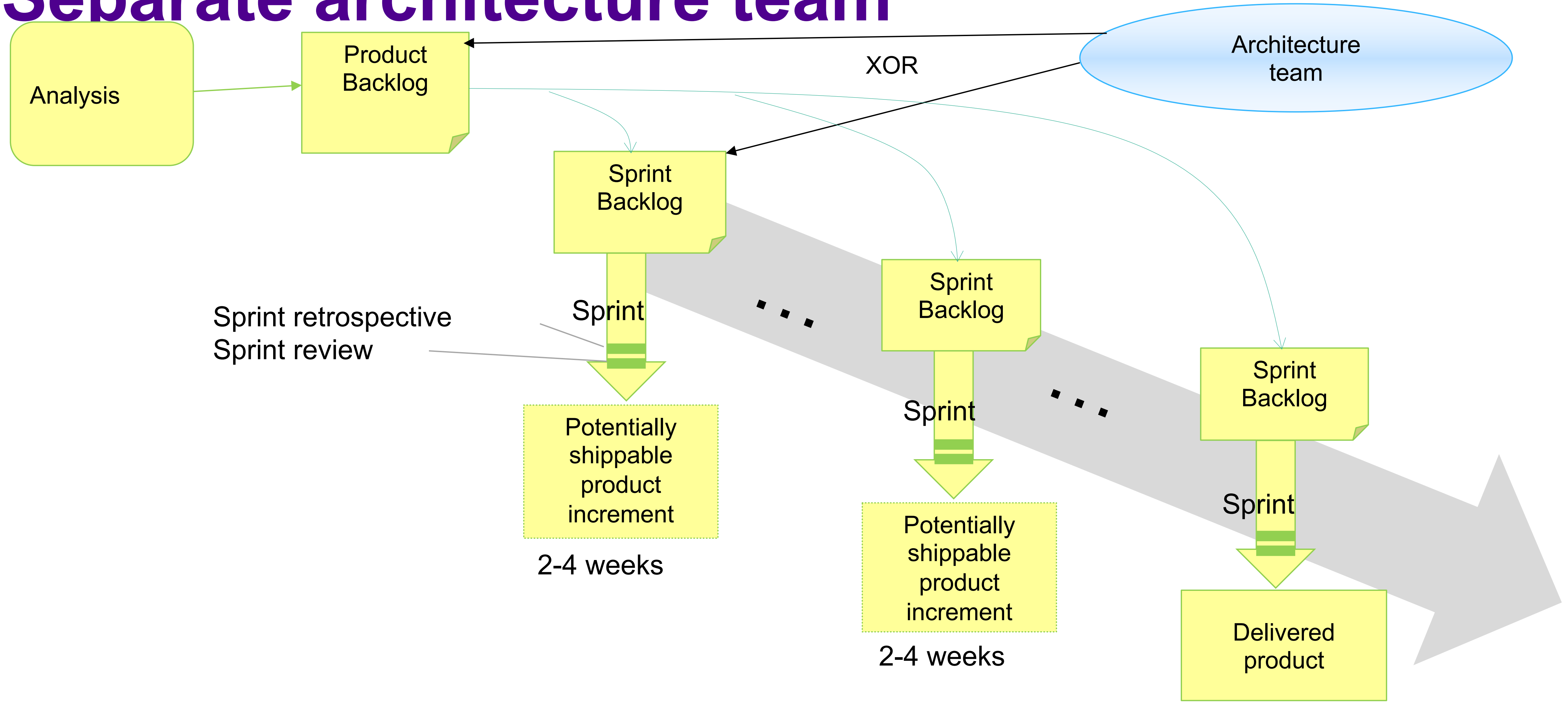
Tampereen yliopisto

Architecture in sprints



Tampereen yliopisto

Separate architecture team



Requirements

Essential requirements are normally prioritised

- **Typically one or two quality properties dominate the architecture.**

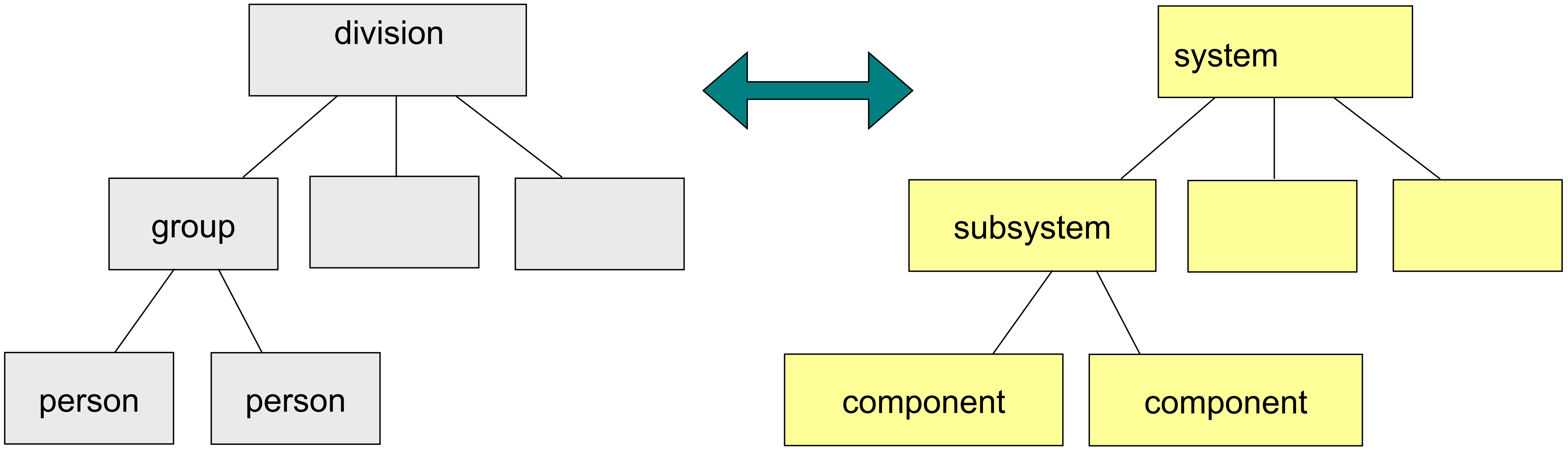
Preserving connection between architecture design and requirements is essential

- **How the system fulfils especially quality requirements.**

Requirements are changing during the system's lifespan

- ***Walking on water and developing software from specifications are easy tasks – when both are frozen*** **(E. Bevard)**

Software architecture and organisation



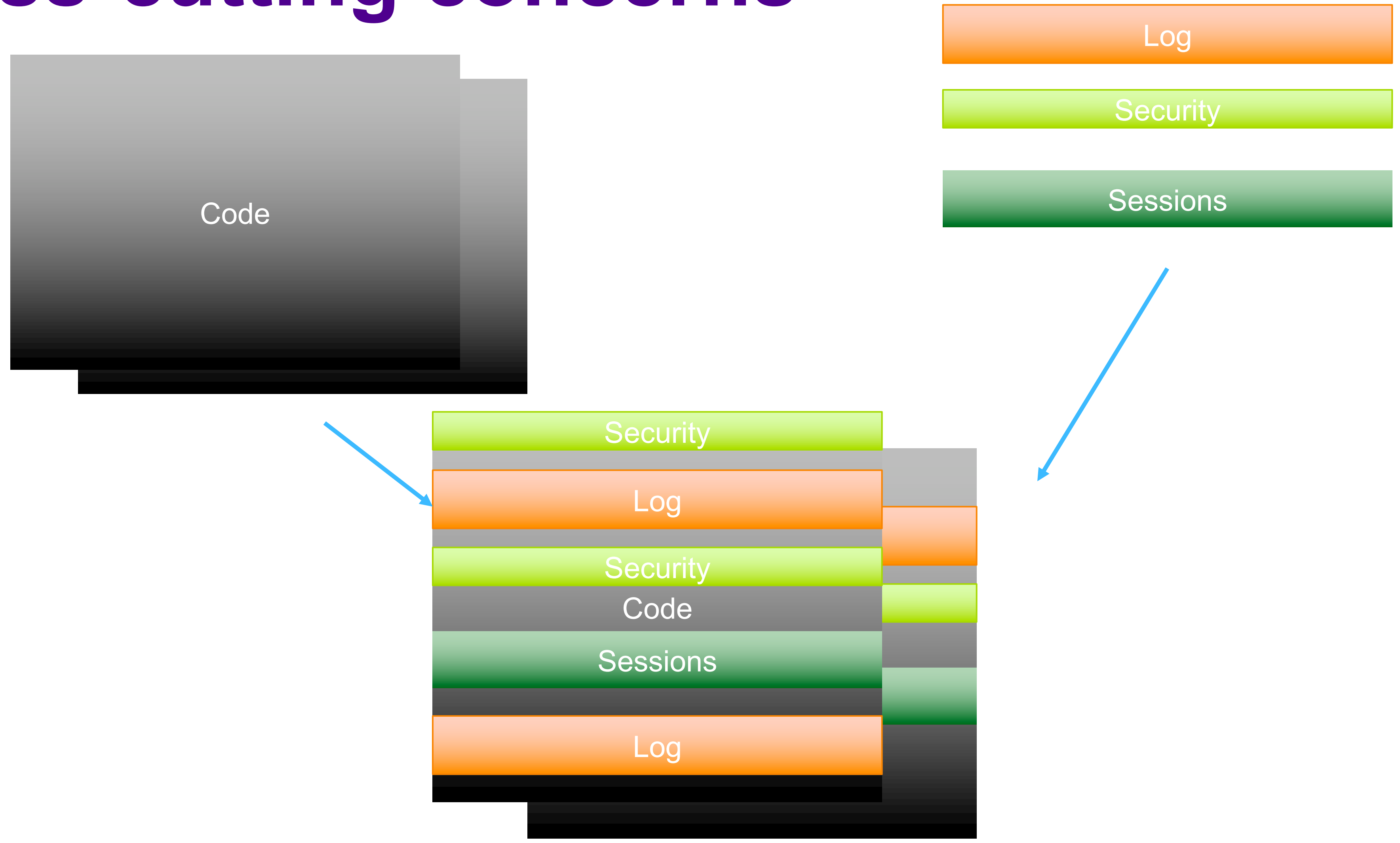
- Conway’s law: the structure of the architecture is the structure of the organisation

Software partitioning

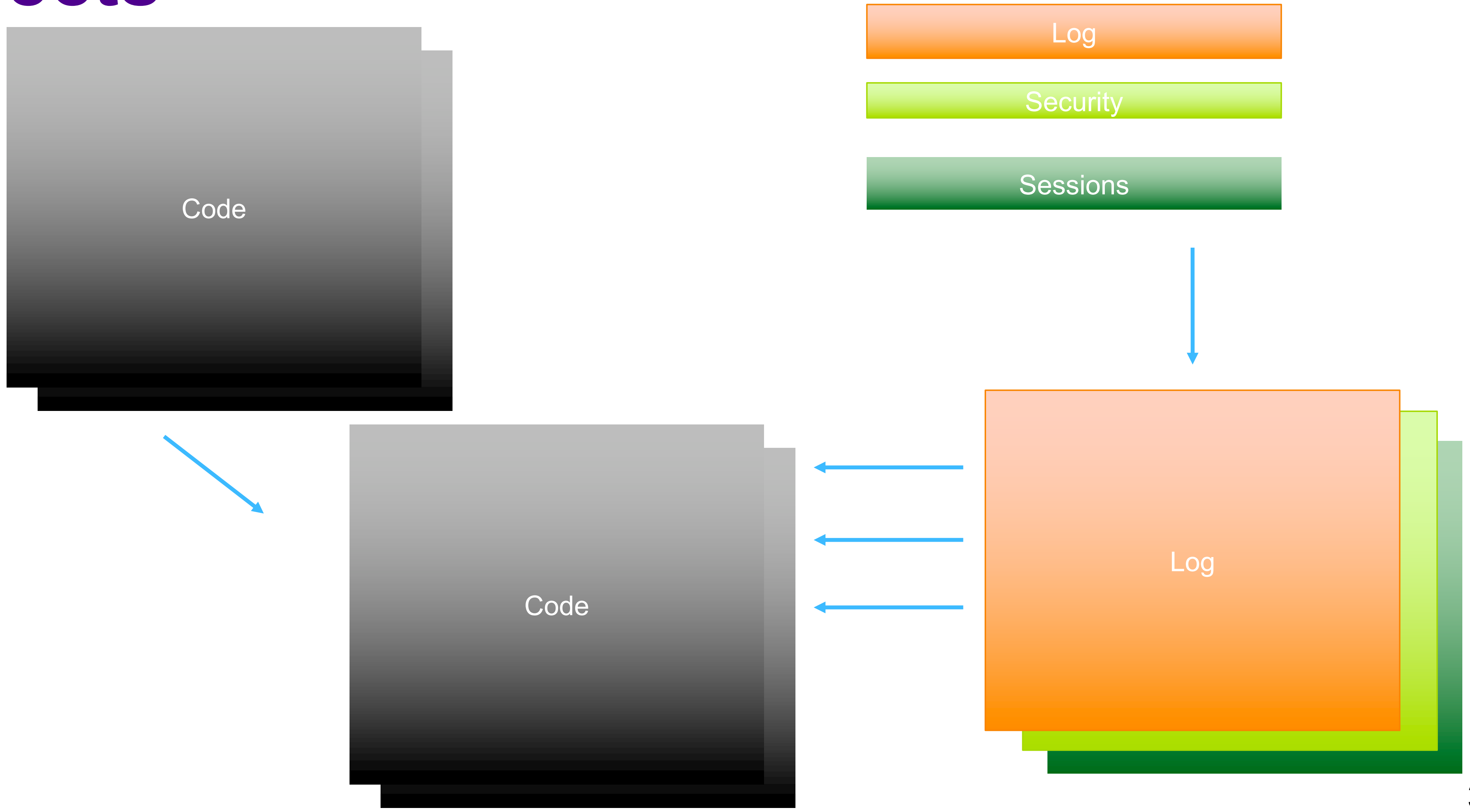
Can be based on:

- **Functionality**
- **Generality**
- **Distribution**
- **Sensitivity to change**
- **Interest**
- **Concern**
- **…**

Cross-cutting concerns



Aspects



Tampereen yliopisto

Architecture of the implementation platform

Implementation platform often enforces a given architecture for applications.

The application is built on this architecture.

Main question of application development is how to implement the requirements on the platform

Example: How to implement on

- **J2EE**
- **.NET**
- **QT**
- **…**
- **Internet communication protocol, P2P network protocol, blockchains, …**

Towards architecture-level implementation tools

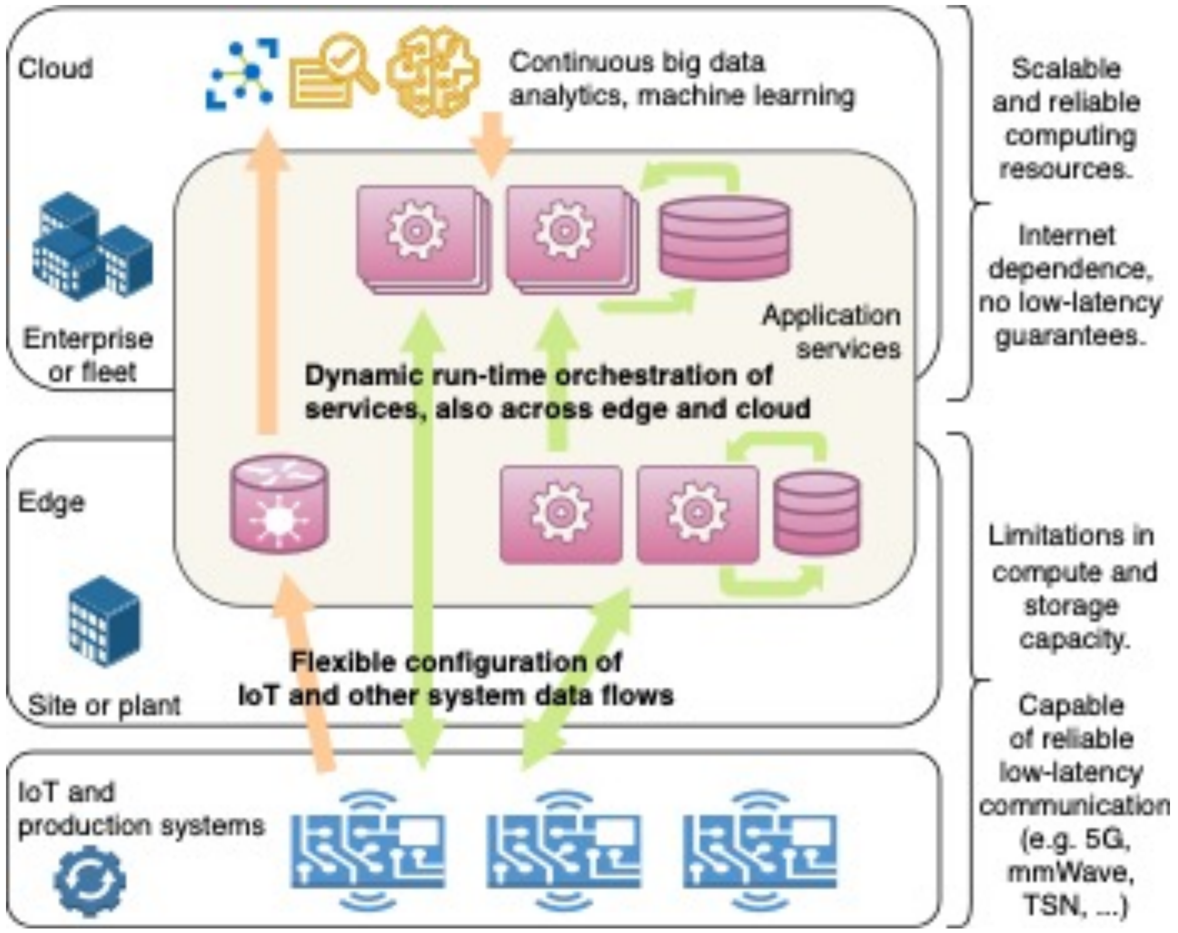
Implementation tools:

- **Variables**
- **Procedures and functions**
- **Classes**
- **Components**
- **Platforms**
- **Ecosystems**

Programming language

Software architecture

Example of distributed software components


Cloud
Continuous big data analytics, machine learning
Enterprise or fleet
Application services
Dynamic run-time orchestration of services, also across edge and cloud
Edge
Site or plant
Flexible configuration of IoT and other system data flows
IoT and production systems
Scalable and reliable computing resources.
Internet dependence, no low-latency guarantees.
Limitations in compute and storage capacity.
Capable of reliable low-latency communication (e.g. 5G, mmWave, TSN, ...)

Example connecting software systems across ecosystems

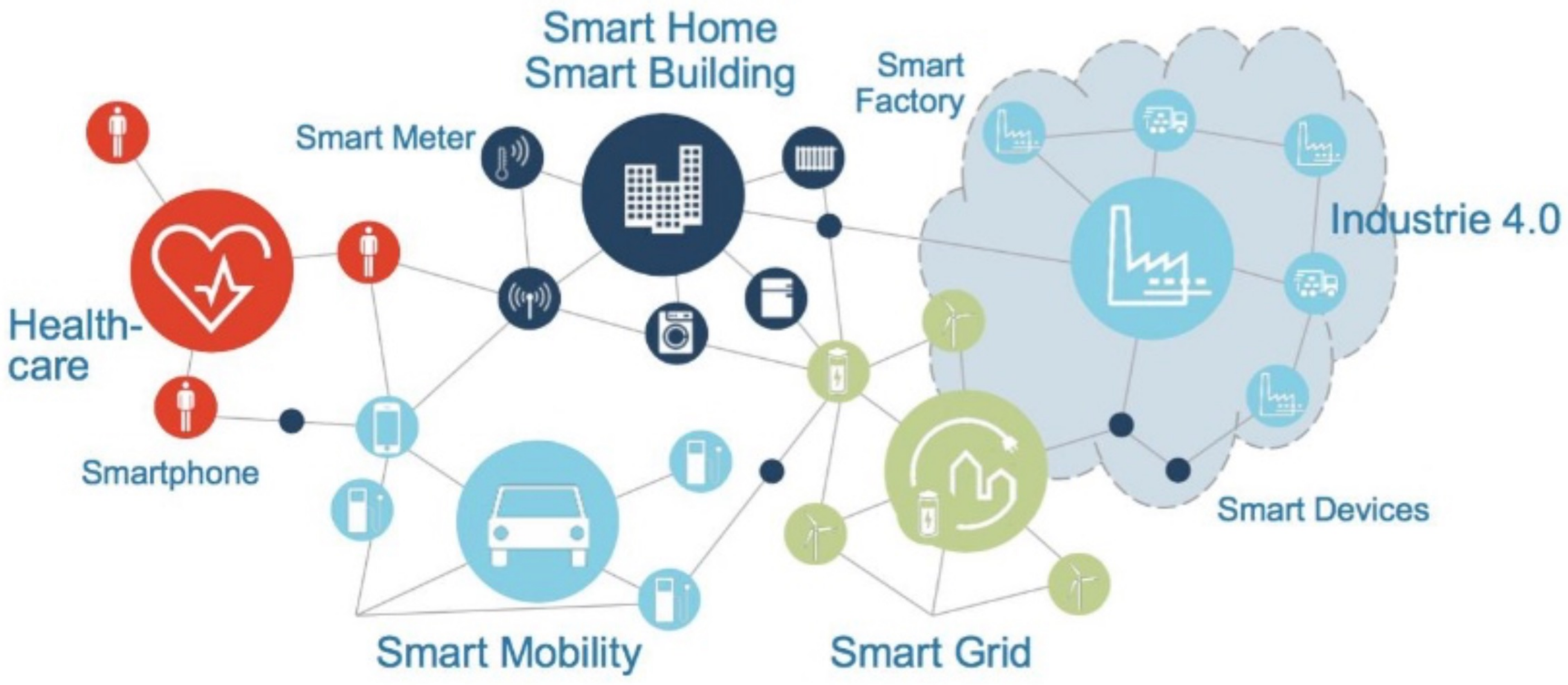


Image curtesy of Bosch Rexroth AG

Tampereen yliopisto

Requirements set for the software architecture

When the software architecture is part of the description of the implementation tools, it has to be:

- **Generic: suitable for many applications**
- **Understandable by a regular programmer**
- **Described precisely and comprehensively**
- **Described on the view of the application programmer.**

Conclusions

Description of a software enables mastering of the system during its lifespan.

There is a solid connection between software architecture and the quality of the software: great part of the architecture is supporting quality properties.

Software development has become architecture-centric

- **Using given technology requires understanding of their architecture**