Software Architectures and Quality

Introduction to Software Architectures

Javier González Huerta

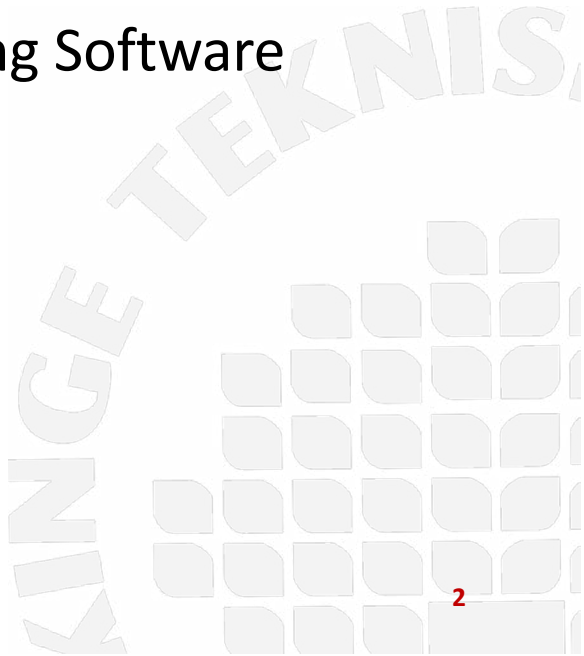
javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

in real life

Objectives

- Introduce the concept of Software Architectures (SA)
- Discuss Software Architecture Influencing Factors
- Discuss what is the relationship between Software Architectures and Quality
- Discuss the importance and complexity of documenting Software architectures



Software Engineering Challenges

- Reduce Development Costs
- Increase System Qualities
- Decrease Maintenance Costs
- Reduce Time-To-Market
- Deliver on time within budget
- Deliver with the right qualities

Software Engineering Challenges

- Reduce Development Costs
- Increase System Qualities
  - Performance
  - Security
  - Reliability
  - Maintainability
  - …
- Decrease Maintenance Costs
- Reduce Time-To-Market

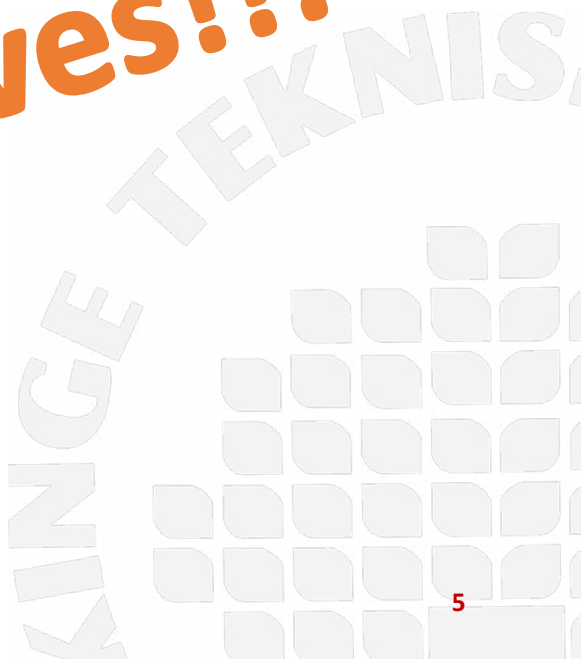
- Deliver on time within budget
- Deliver with the right quality

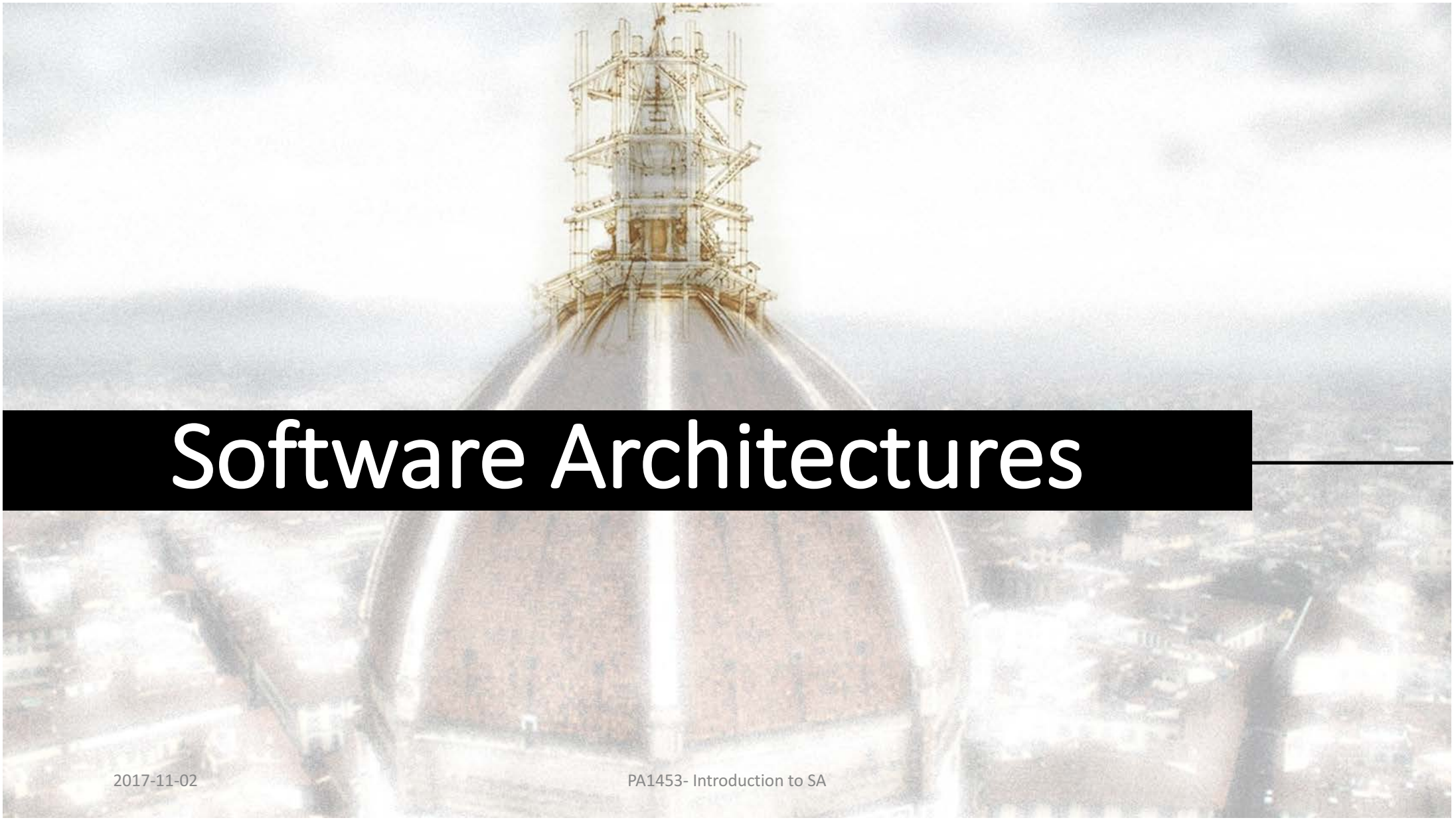
Software Engineering Challenges

- Reduce Development Costs
- Increase System Qualities
  - Performance
  - Security
  - Reliability
  - Maintainability
  - …
- Decrease Maintenance Costs
- Reduce Time-To-Market
- Deliver on time within budget
- Deliver with the right quality

Big Problem!!!

Multiples Perspectives!!!

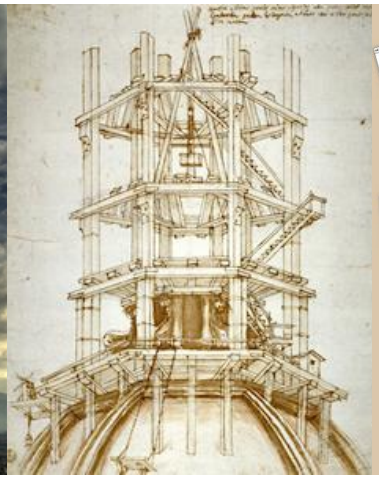
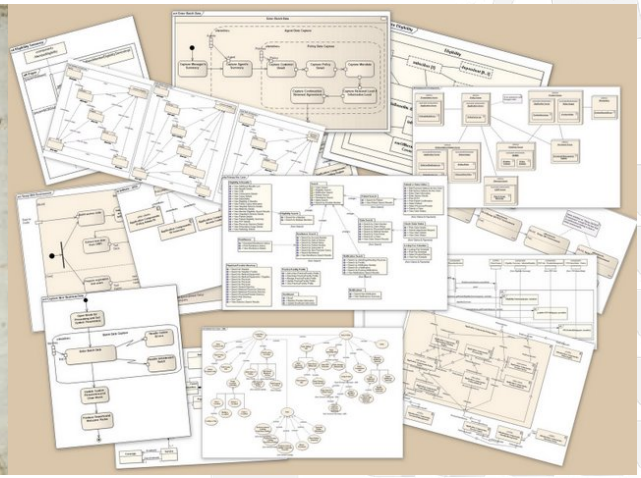




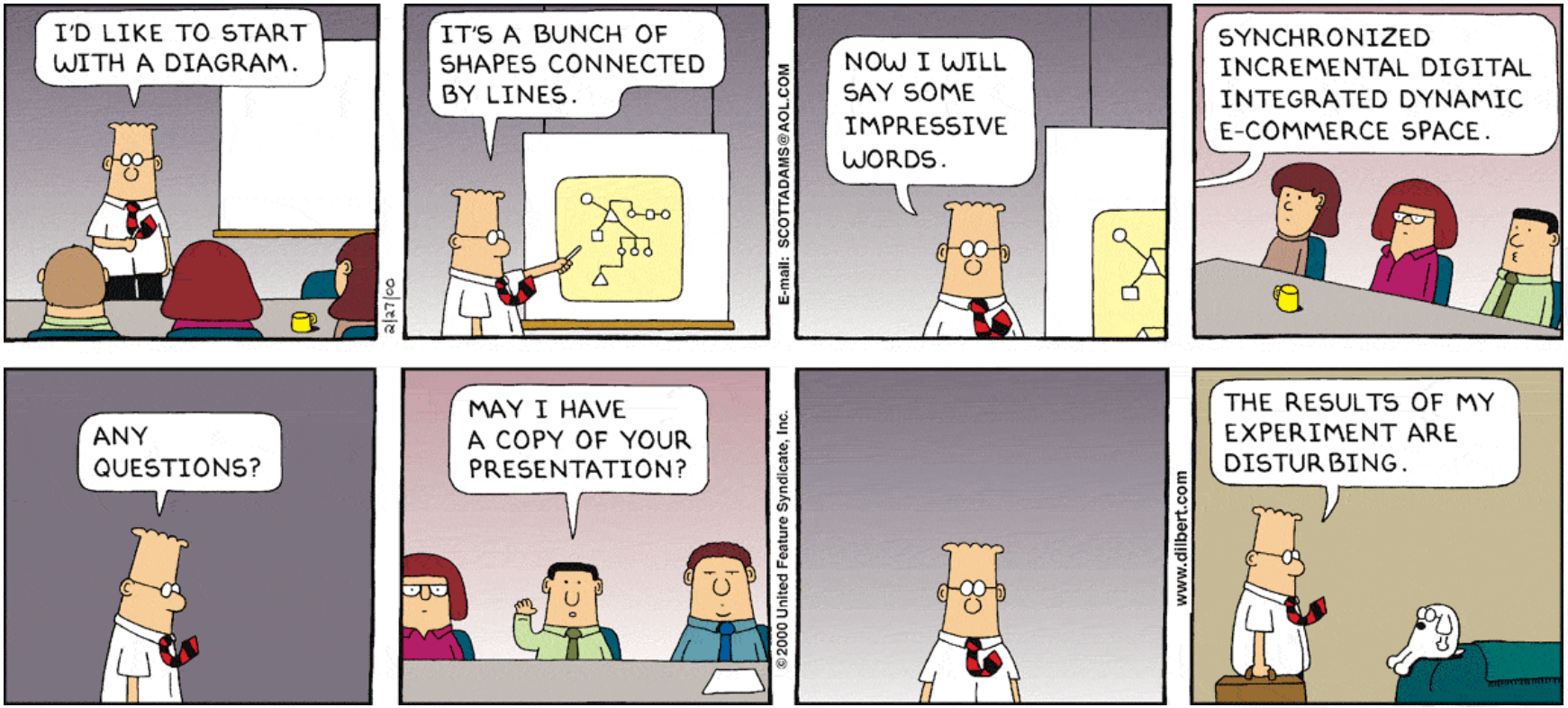
Software Architectures

Generic Concept of Architectures

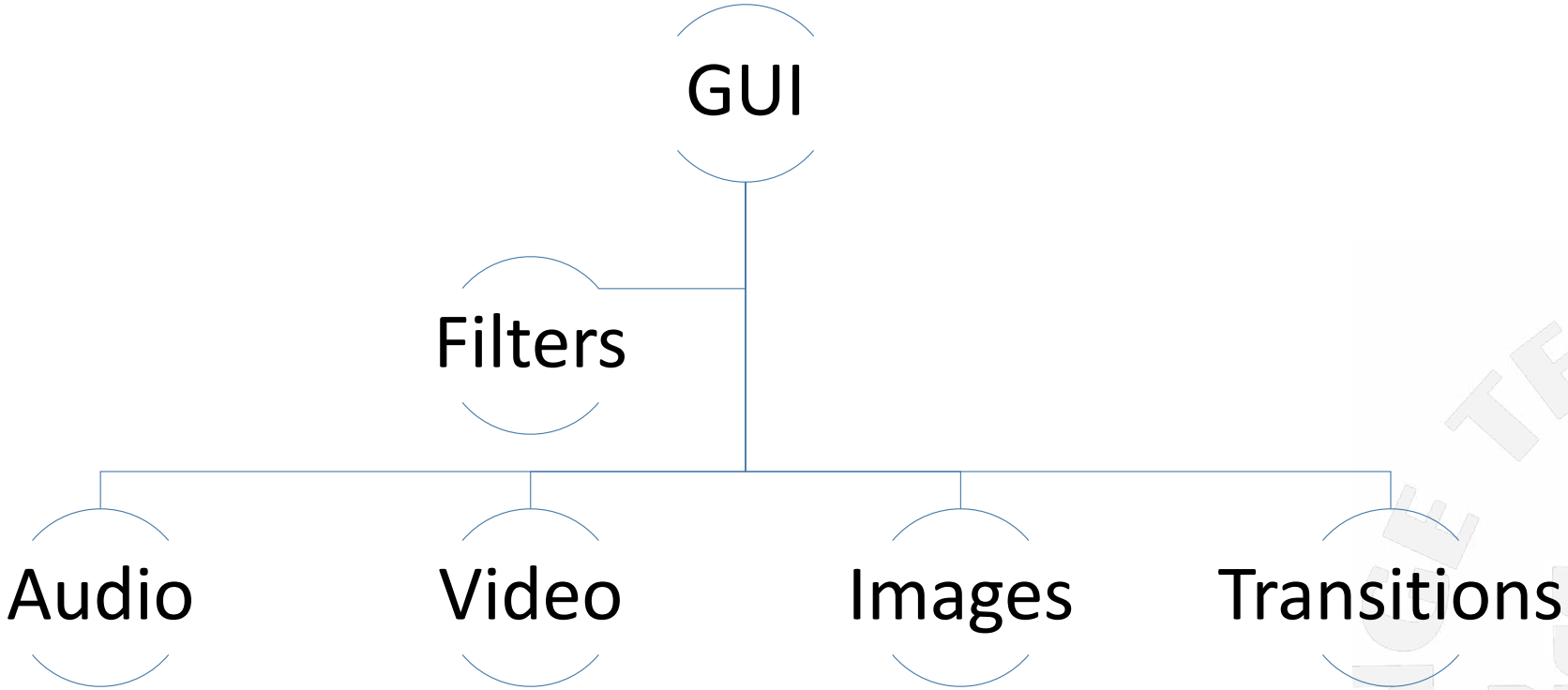
- Architecture is the partitioning of a whole into parts, with specific relations among these parts.
  - This partitioning is what allows groups of people—often separated by organizational, geographical, and even time-zone boundaries—to work cooperatively and productively together to solve a much larger problem than any of them could solve individually.

Software Architecture: Beyond Lines and Boxes

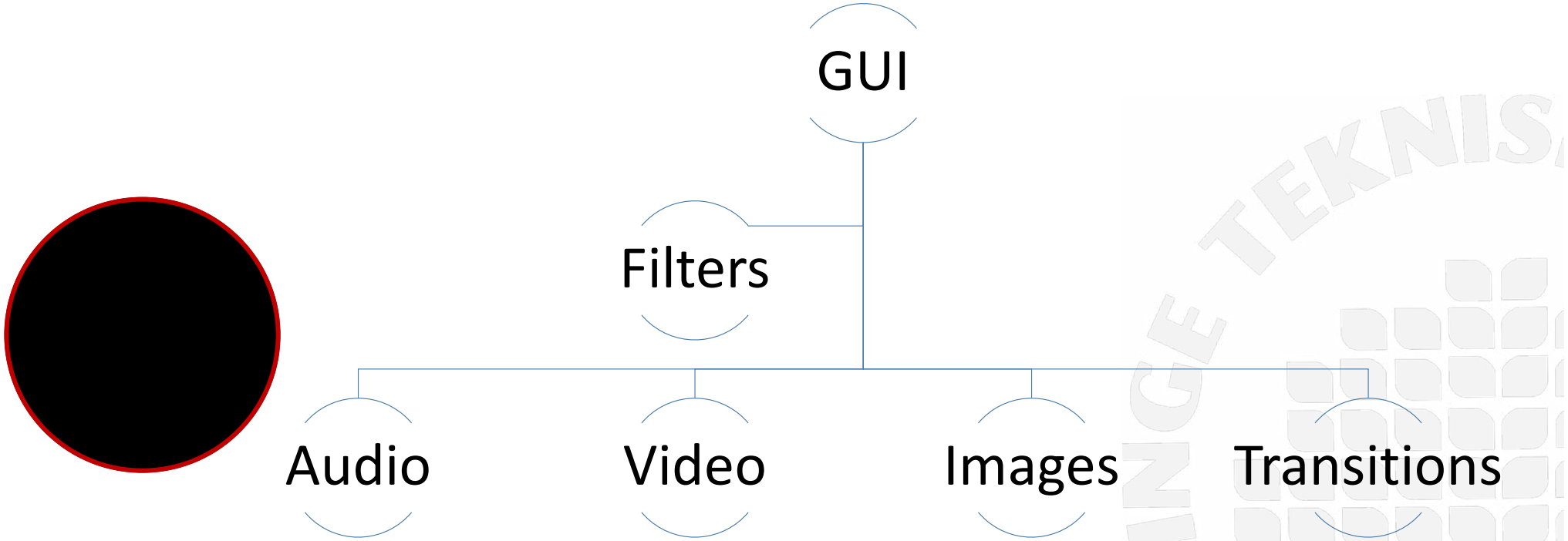


Software Architecture: Beyond Boxes and Lines



Reflection

Discuss during 2 minutes 2-3 persons: What can we say about this system?



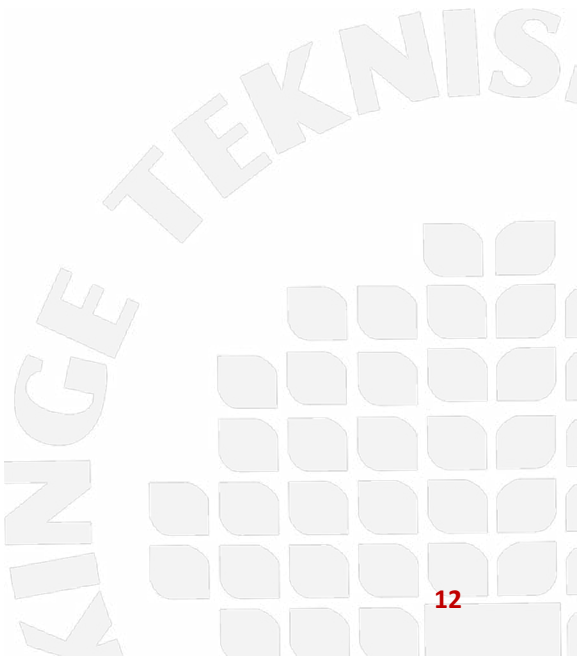
Software Architecture: Beyond Boxes and Lines

- The system seems to have six elements
- Four elements 

seem to have something in common (they are at the same level)
- Apparently all the elements have some sort of relationship, since everything is connected with almost everything

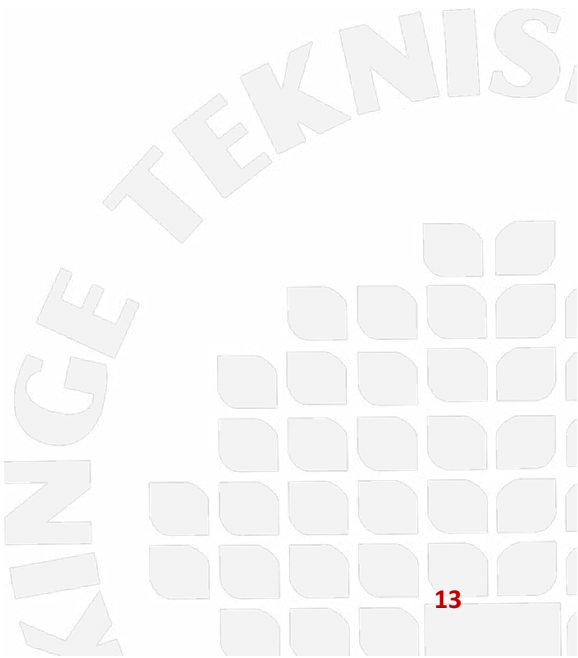
Software Architecture: Beyond Boxes and Lines

- What is the nature of the elements?
  - What is the significance of their separation?
  - Do they run on separate processors?
- What are the responsibilities of the elements?
- What is the significance of the connections?
  - Means data exchange? Control flow?
- What is the significance of the layout?
  - What do the levels mean?
  - Why is one of them in between two levels?

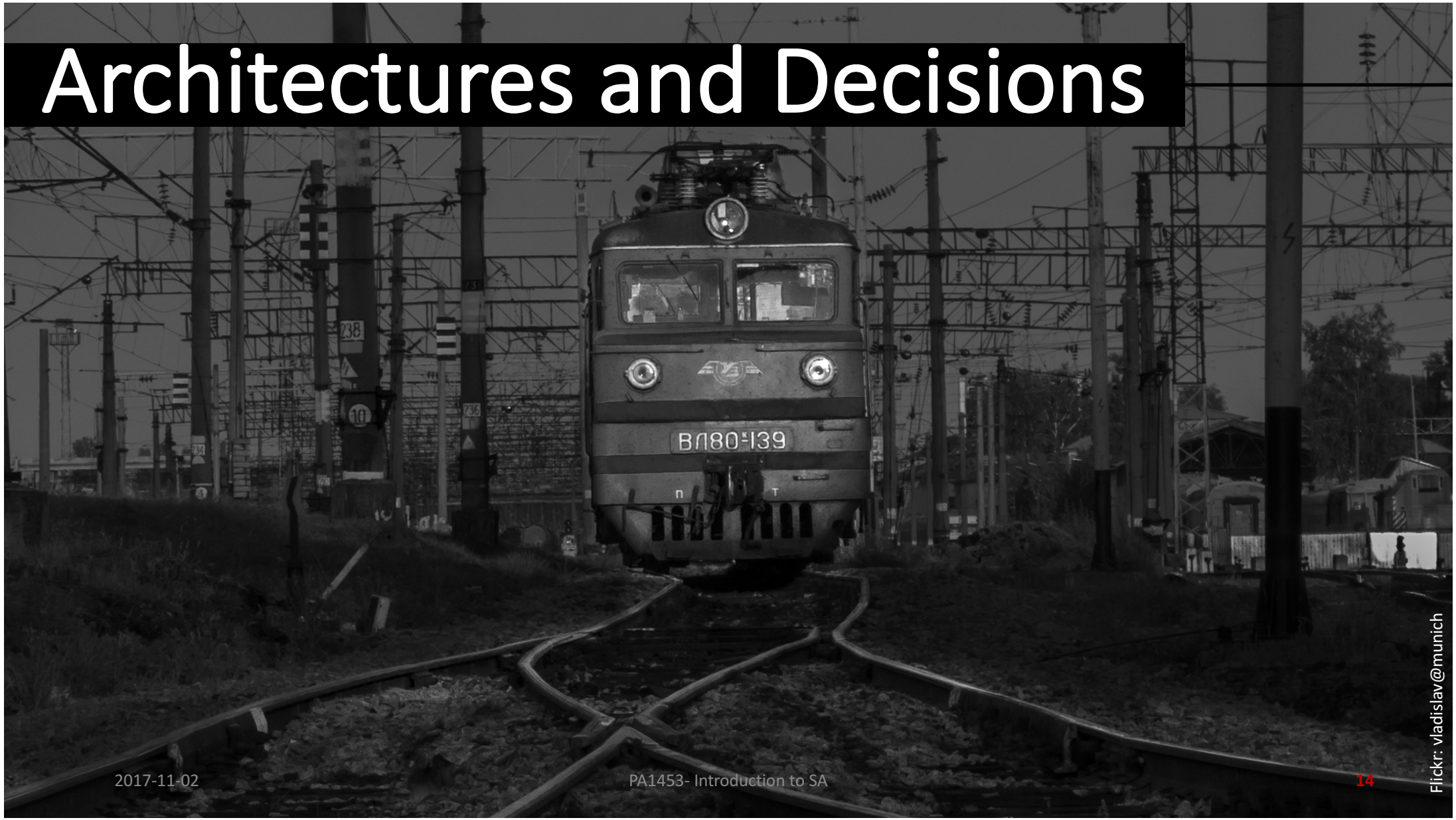


Software Architecture: Beyond Boxes and Lines

- Unless we already know precisely what the elements are and how they cooperate, the diagram is useless
- This diagram does not show the architecture in a useful way, is a starting point



Architectures and Decisions



Flickr: vladislav@munich

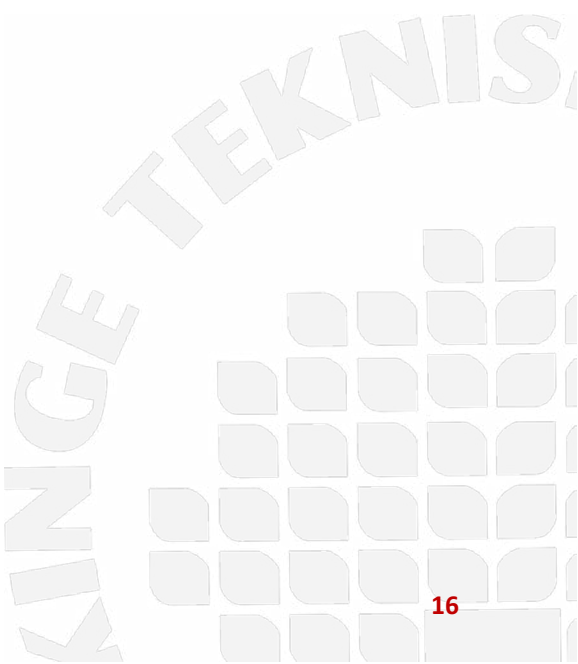
Architectural decisions

“The life of a software architect is a long (and sometimes painful) succession of suboptimal decisions made partly in the dark“

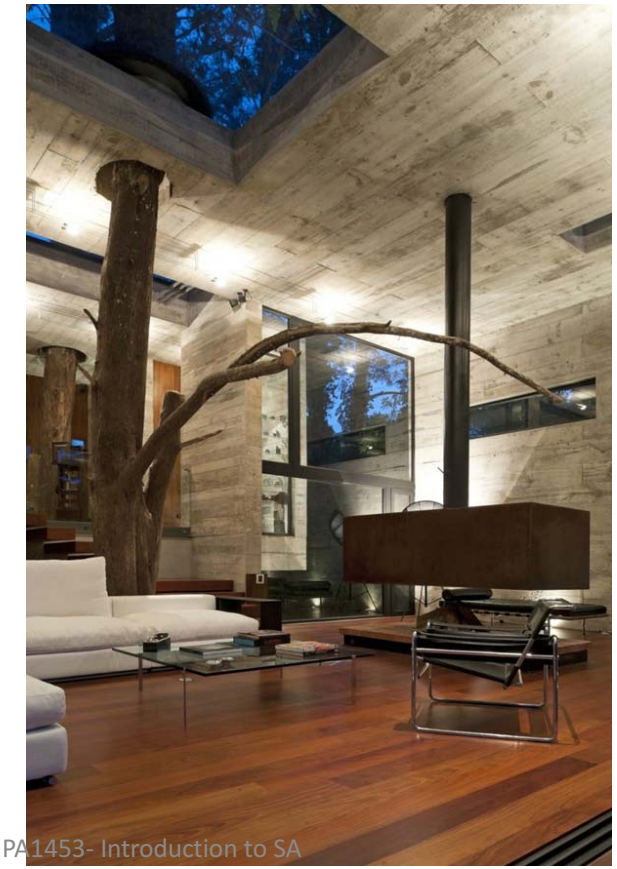
P. Kruchten

Architecting a System: Decisions

- Architecting a system consists on:
  - Making technical and business decisions to balance stakeholder needs
  - Identifying the necessary decisions that satisfy all stakeholders, also meeting the organizational constraints

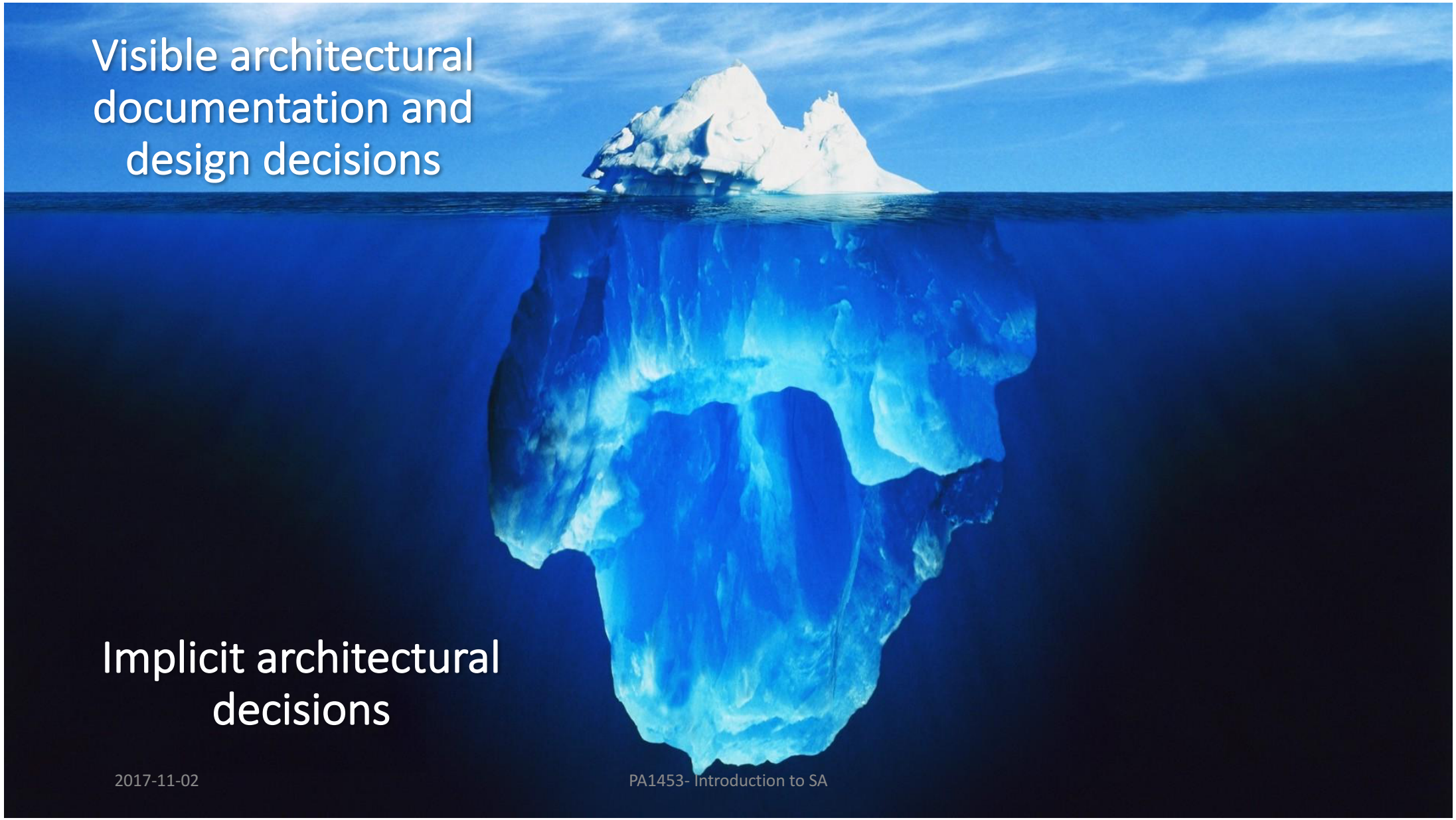


Architecture and Decisions



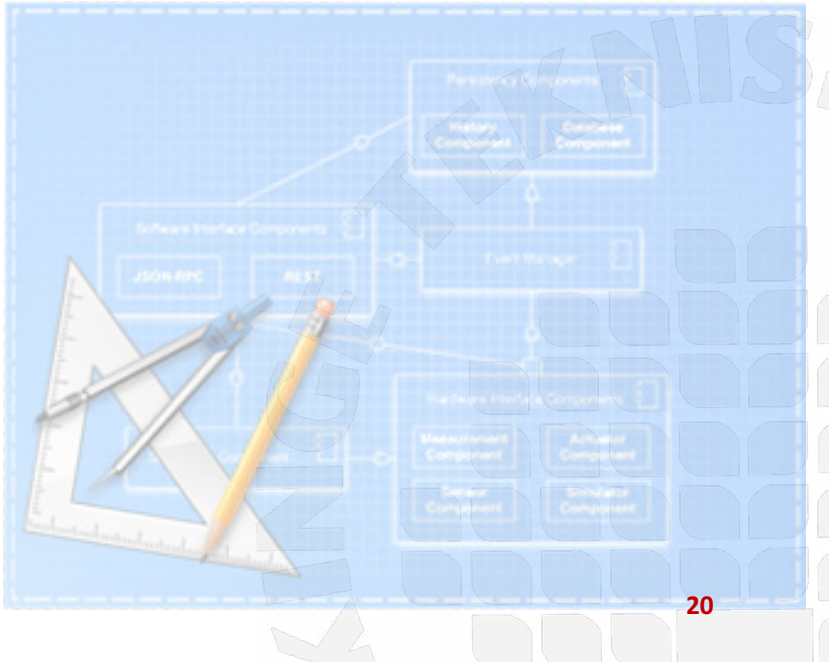
Architecting a System: Decisions

- Architecting a system consists on
  - Making technical and business decisions to balance stakeholder needs
  - Identifying the necessary decisions that satisfy all stakeholders, also meeting the organizational constraints
  - Document them…
- These decisions *are* the architecture
  - The rest is the architectural documentation and its instantiation


Visible architectural documentation and design decisions
Implicit architectural decisions

What is a Software Architecture? Academic Definition

- A software architecture is the set of structures needed to reason about a (typically complex) software system
- These structures are described by:
  - Elements
  - Relationships between elements
  - Visible properties of elements and relationships



But… What Is Software Design?

- Defining the overall structure of the software and data organization
- Definition – "*the process of defining the architecture, components, interfaces, and other characteristics of a system or component*" and "*the result of [that] process*"

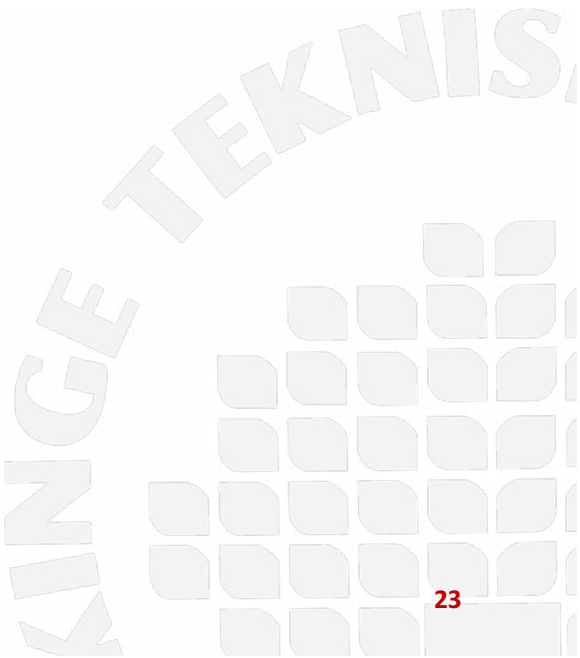
[1] ISO/IEC/IEEE 24765:2010 Systems and Software Engineering—Vocabulary, ISO/IEC/IEEE, 2010.

Architecture vs Design: Where to draw the line?

- Architecture is about satisfying the quality, behavioral requirements and business goals
- Architecture describes the constrains and rules to implement and design the system
- All the fine grained decisions that go beyond that are **non-architectural design decisions** (AKA detailed design)
  - For example which data structure to use or design details about structure of interfaces or algorithms used

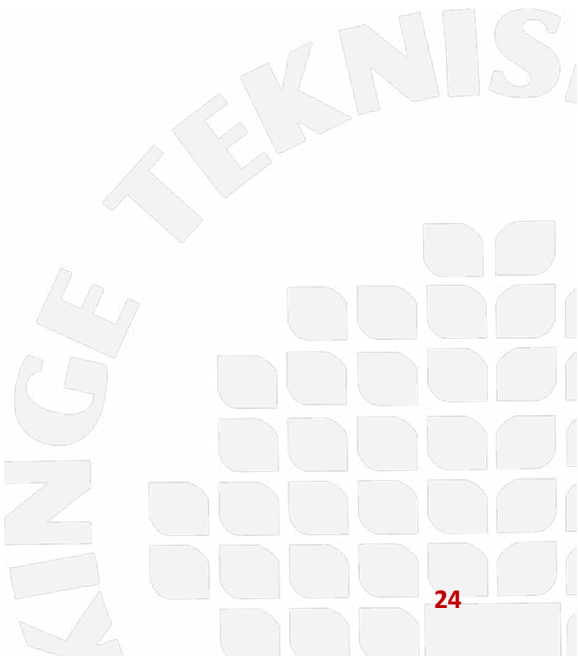
Software Architectures and Quality

- The software architecture enables or inhibits systems quality attributes
- Systems are often replaced not because they are functionally deficient but because they lack of some qualities
  - They are difficult to maintain
  - Slow
  - No compatible with other systems
  - Not portable to other platforms



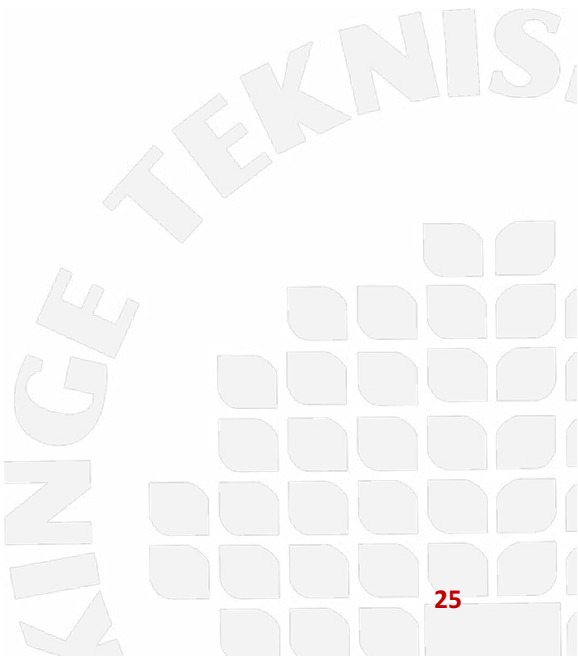
Software Architectures and Quality

- But obviously the architecture alone cannot guarantee the functionality and quality
  - *Architecture giveth and the implementation taketh away*

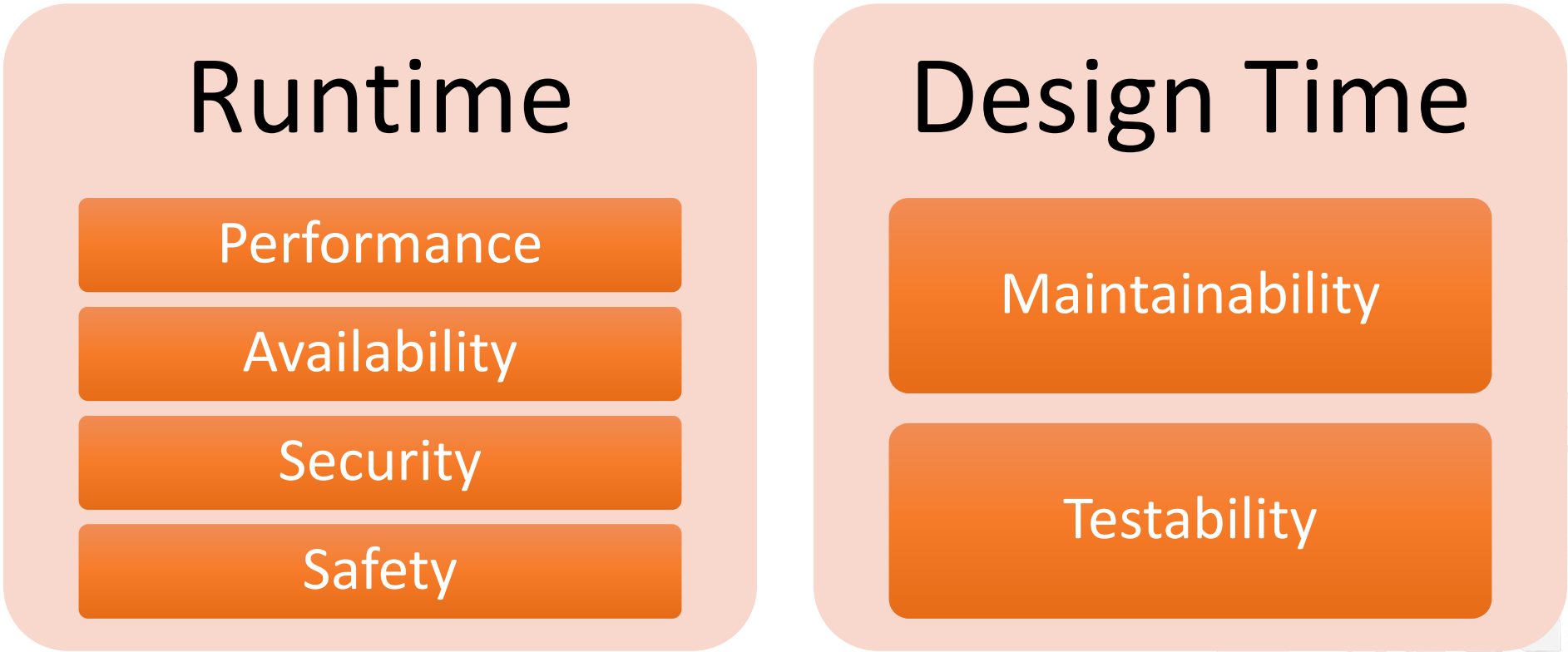


Software Architectures and Quality

- But obviously the architecture alone cannot guarantee the functionality and quality
  - *Architecture giveth and the implementation taketh away*

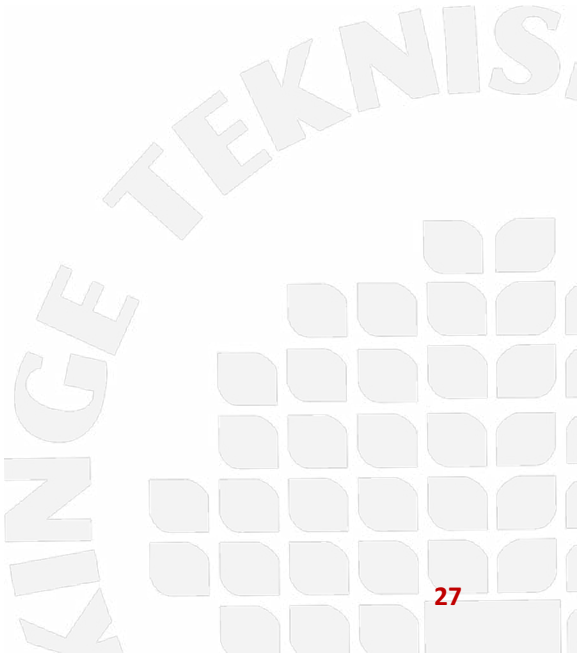


Qualities: Run-time vs Design-time



Importance of Software Architecture

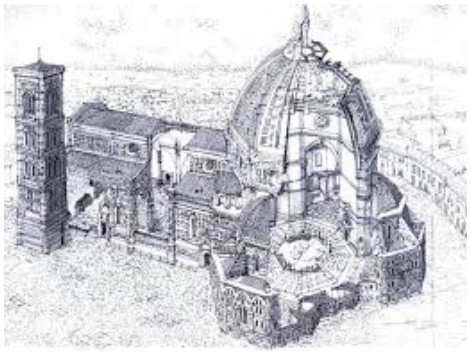
- SA Enables/inhibits the qualities of the system both at runtime and at development levels
- Supports impact analysis
- Supports evaluation / prediction
- Defines policies, guidelines and constraints

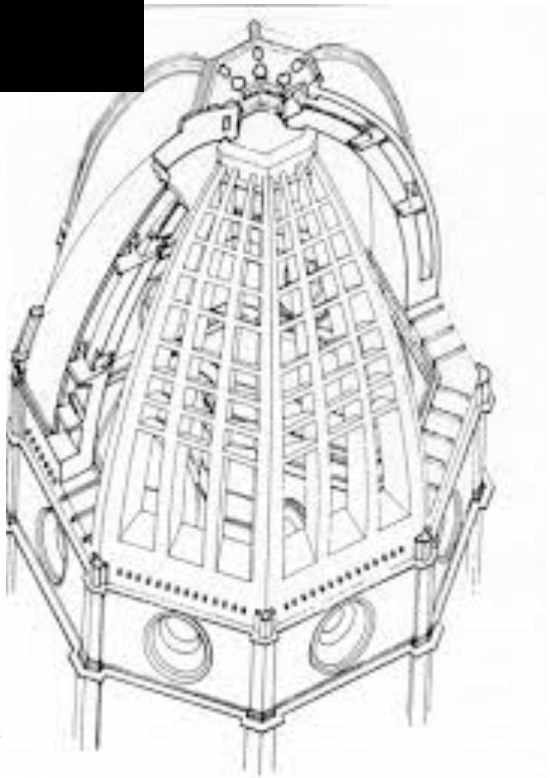


Why documenting Software Architectures?

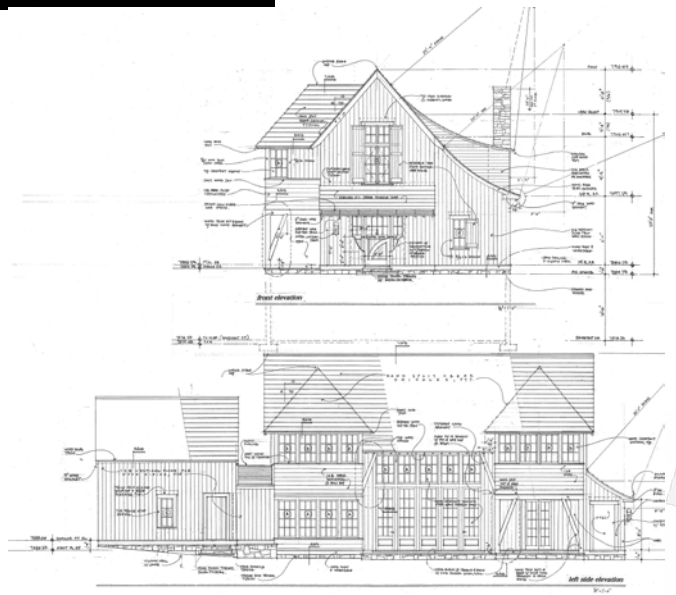
- A good architecture (even the best one) will be useless if the stakeholders cannot understand it
  - Or even worse that, if introduces misunderstandings that makes them apply it incorrectly
- Architectures need to be communicated in a way that allows stakeholders to understand it and use it properly in their jobs
- Architecture documentation should be produced because is relevant (or essential), not because we need to do it

Documenting Software Architectures





Architecture and views



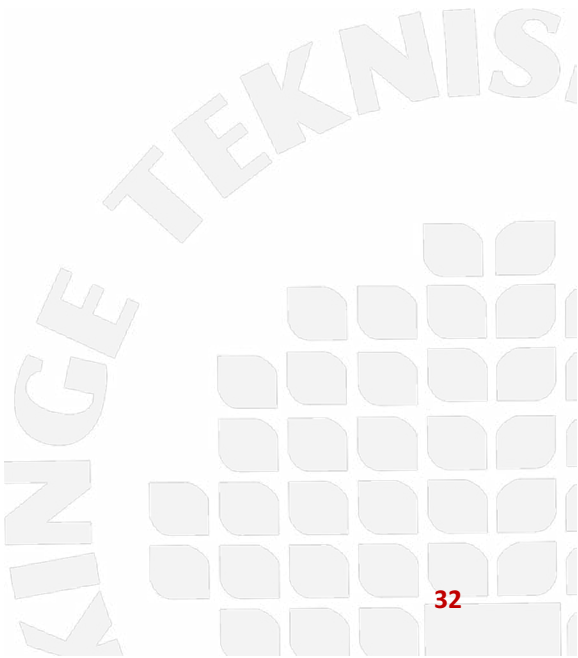
2ND FLOOR PLAN

Architecture Documentation: Views

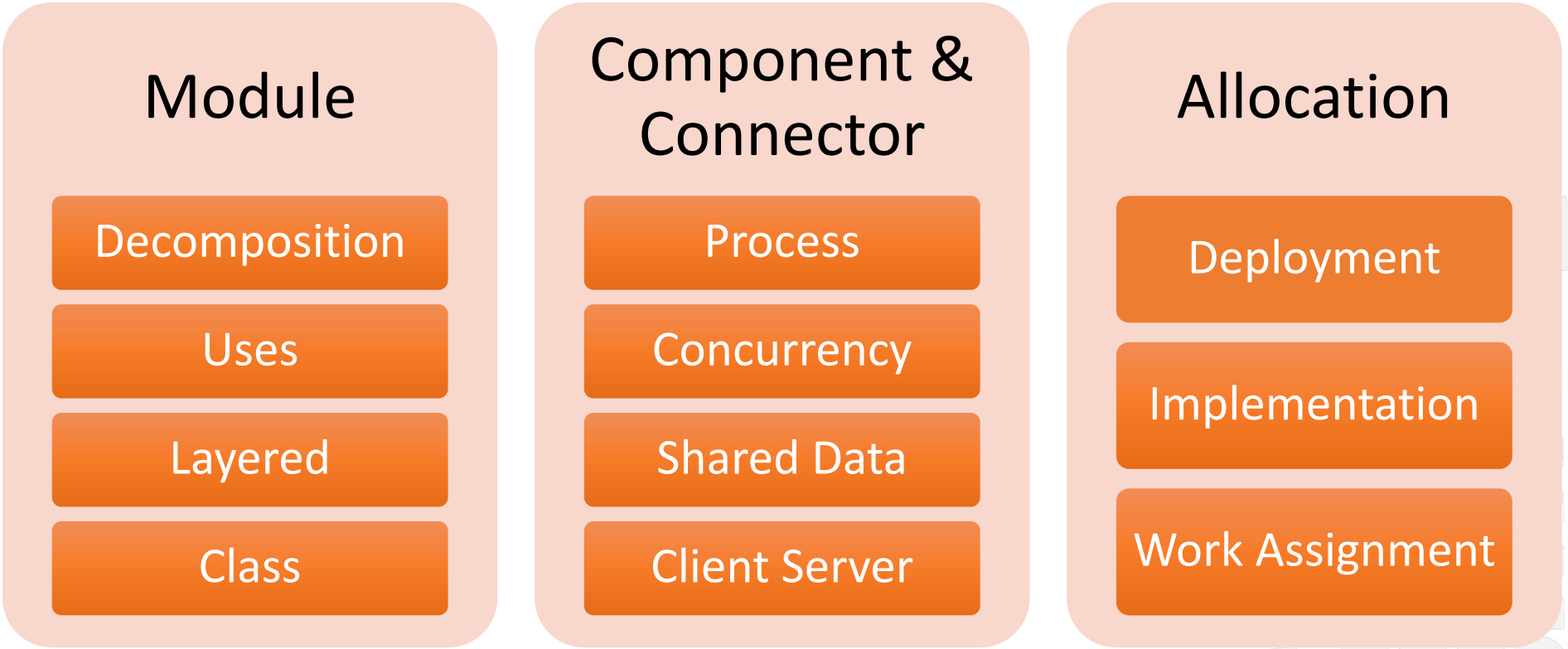
- We defined architecture is a set of structures needed to reason about a [large] piece of software
- These structures are described in terms of:
  - Elements
  - Relations between elements
  - Properties of elements and relationships

Structures and Views

- We can find several different approaches to describe the architectures in terms views:
  - Views and Beyond
  - Kruchten 4 + 1
  - Hofmeister et al
  - ISO

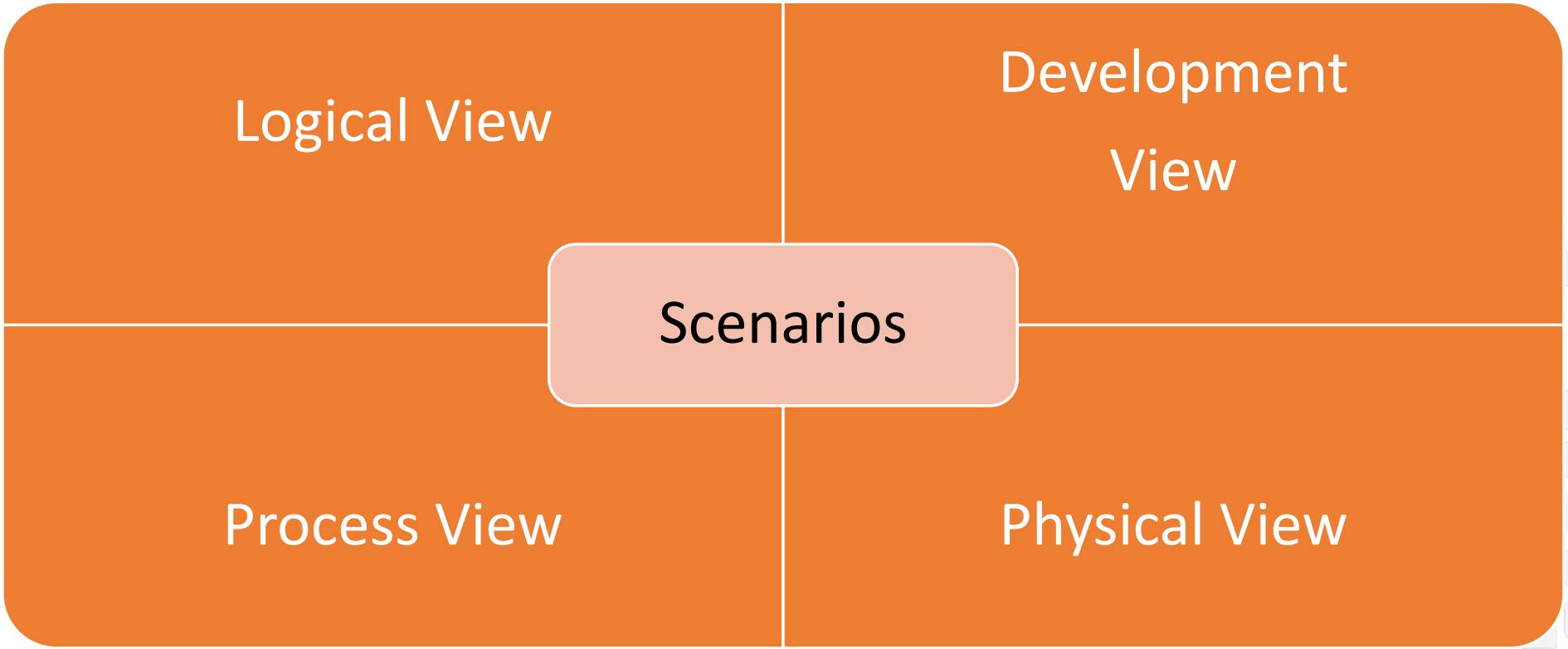


Structures and Views: Views and Beyond Method

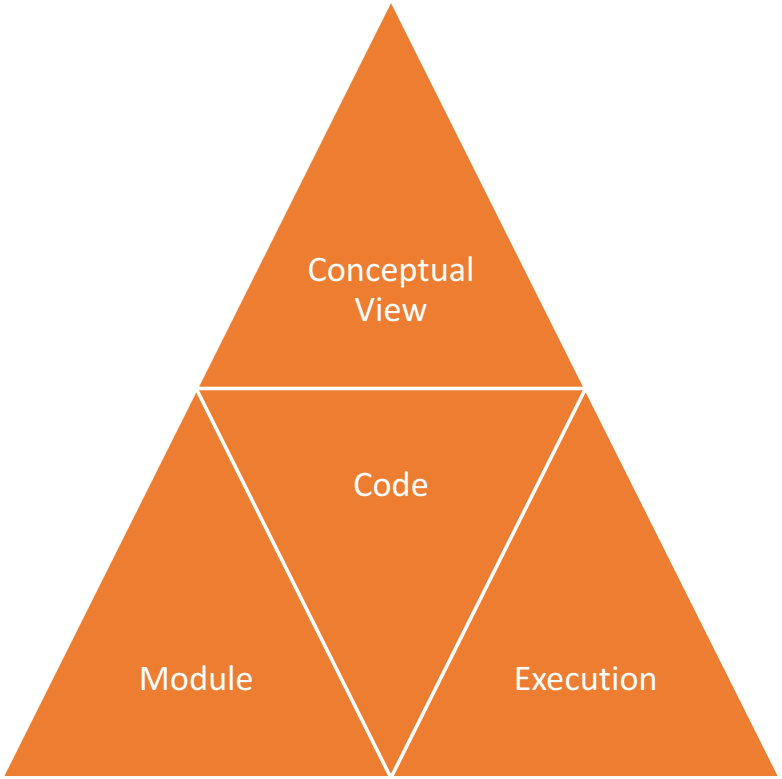


P. C. Clements *et al.*, *Documenting software architectures: views and beyond*, 2nd Editio. Pearson Education, 2010.

Structures and Views: Kruchten 4+1



Structures and Views: Hofmeister et al



Some Intermediate Stages

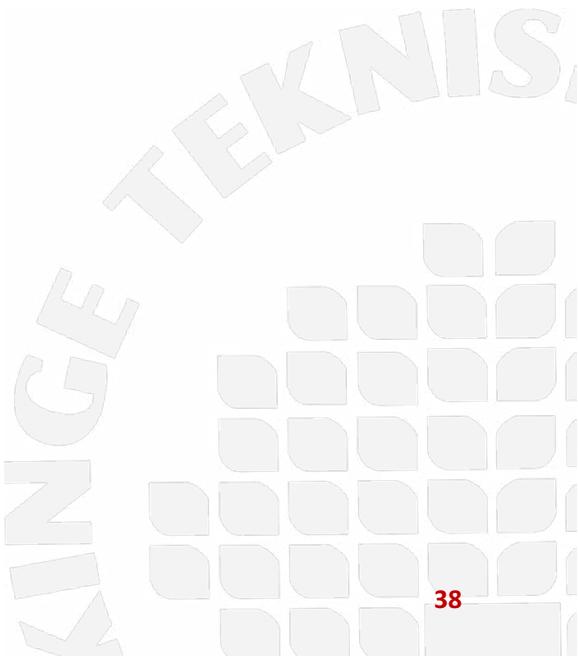
- Architectural Patterns (and Styles)
  - Collection of elements, relations and constraints
  - Can be seen as packaged strategies to solve a common, recurring problem
  - Exhibit known quality attributes, that can help us fixing some quality issues

Some Intermediate Stages

- Architectural Patterns (and Styles)
  - Collection of elements, relations and constraints
  - Can be seen as packaged strategies to solve a common, recurring problem
  - Exhibit known quality attributes, that can help us fixing some quality issues
- Reference Model
  - Decomposition of a problem into parts that cooperatively solve the problem
  - Domain specific
- Reference Architecture
  - Reference model mapped into a system software elements

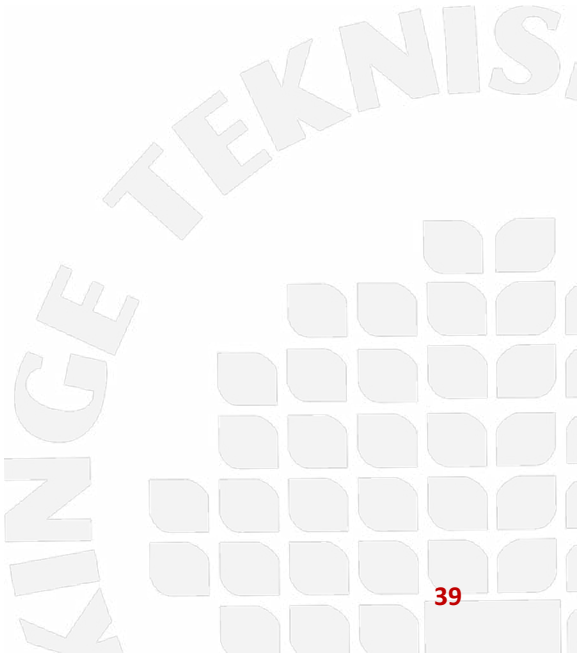
The Napkin Architecture

- The few necessary abstractions needed to describe simple solutions to complex problems
- The Occam's Razor
  - *“Among competing hypotheses, the one with the fewest assumptions should be selected.”*
  - *The simpler the better…*



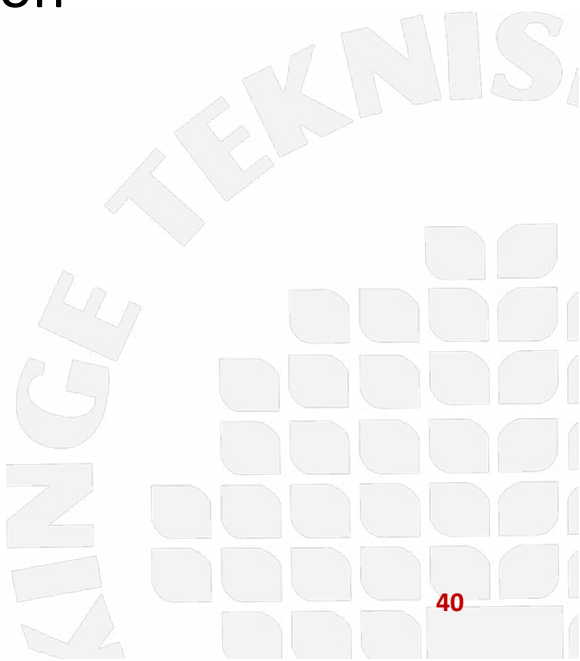
Uses and Audience of a Software Architecture

- Architecture Is a means of *understanding and educating*
  - introducing the system to new team members or stakeholders
- Is a *primary vehicle for communicating* among stakeholders
- Helps communicating to developers what and how to implement [parts of] the system
- Is a tool for planning



Uses and Audience of a Software Architecture

- Is a tool for planning
- Constraints the implementation
- Is the basis for quality evaluation and prediction
- Allows checking the conformance of the implementation
- Allows identifying *reuse*

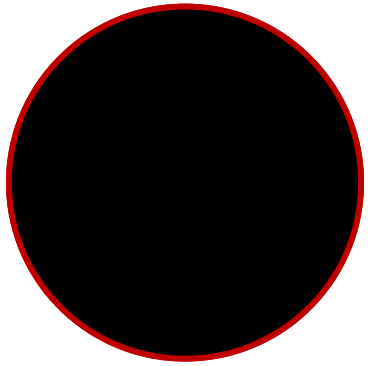


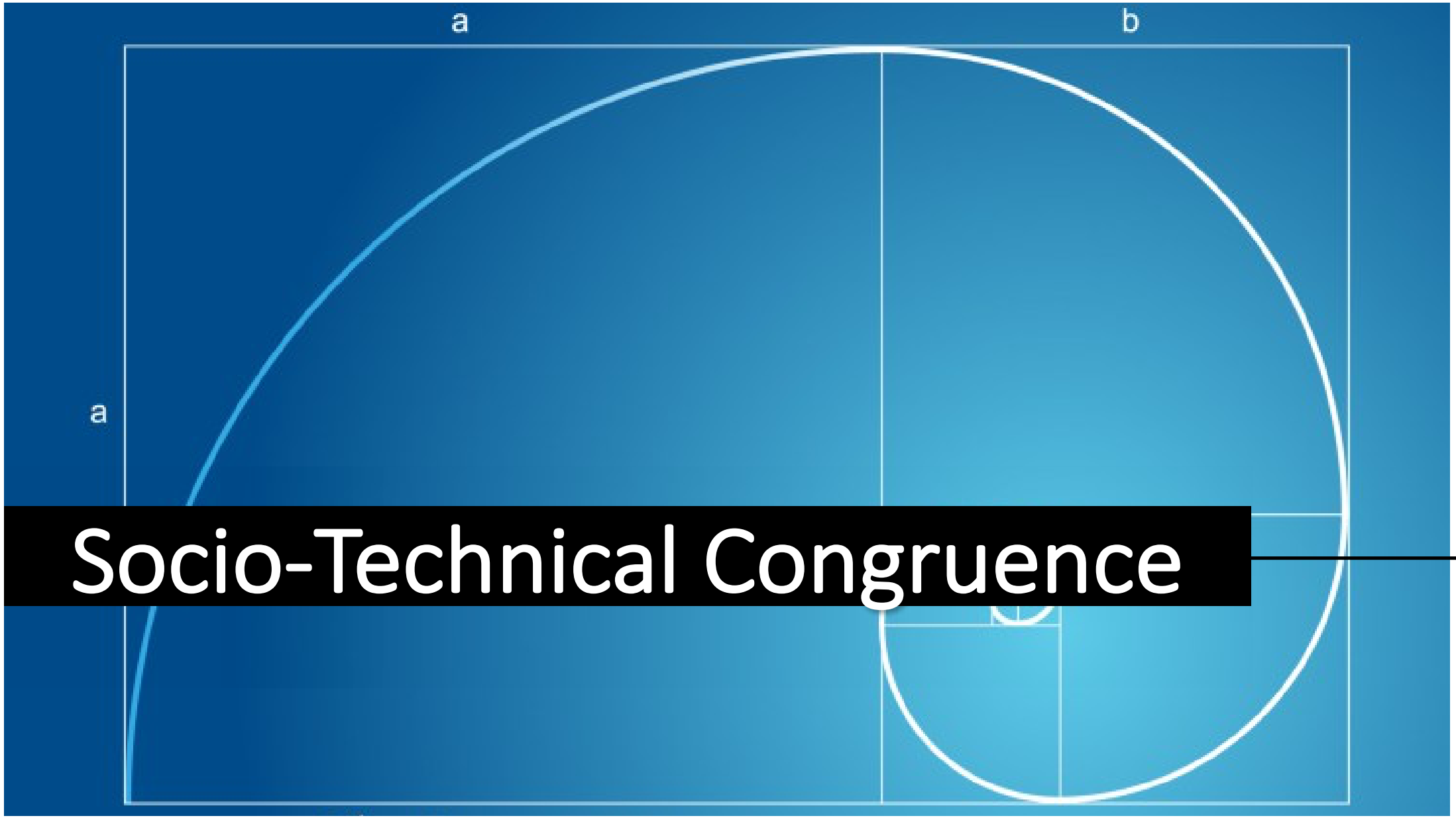
What is a ”Good” Software Architecture

- The architecture is defined based on a prioritized list of current and future quality attributes
- Makes use of a set of well-known architectural patterns, styles and tactics to assure thee quality attributes
- It should be documented using different views
- Is evaluated to assure that meets the specific quality goals for the system
- The interactions between components are handled through a small set of interaction patterns that are consistently used

Reflection

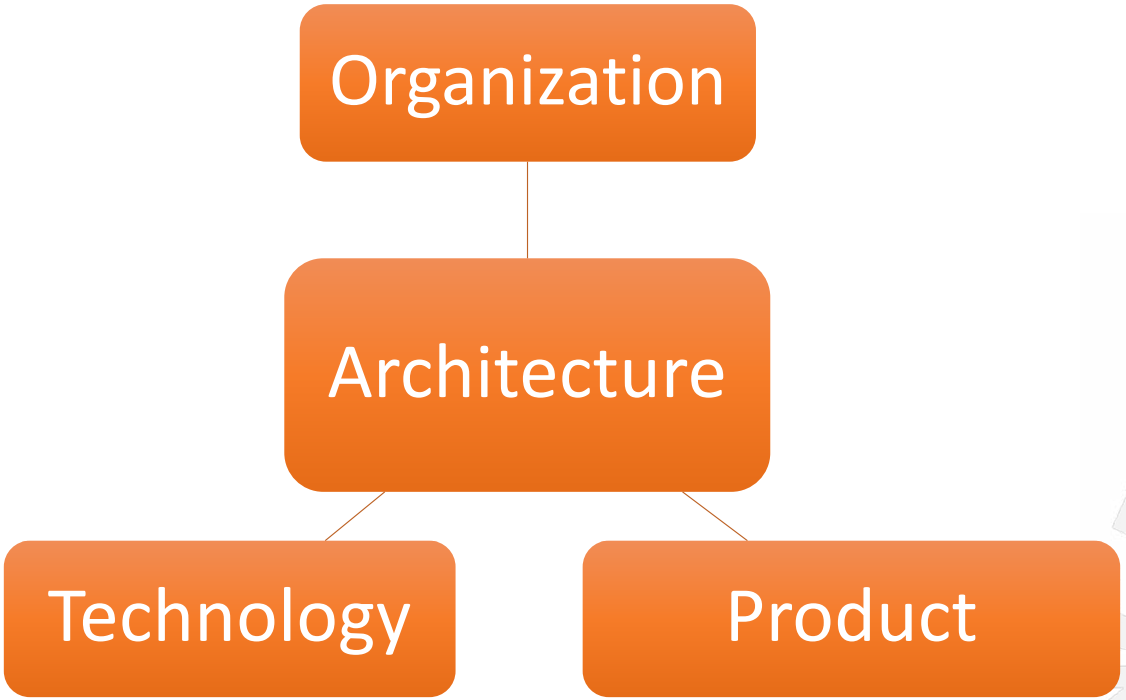
- **Discuss during 2 minutes in groups of 2-3 persons: What are the Factors that Influence the Architecture?**





Socio-Technical Congruence

Architecture Influencing Factors

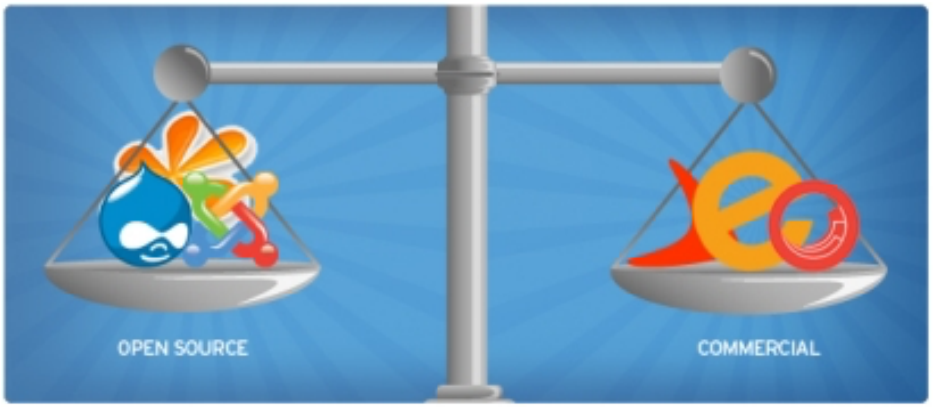


Influencing Factors

- There are several factors that influence the architecture
  - **Product Factors:** Customer functional and quality requirements
  - **Technological Factors:** The decisions made influenced by the hardware, framework, libraries, target platforms or programming languages
  - **Organizational Factors:** The decisions made influenced by the development team, budget or schedule

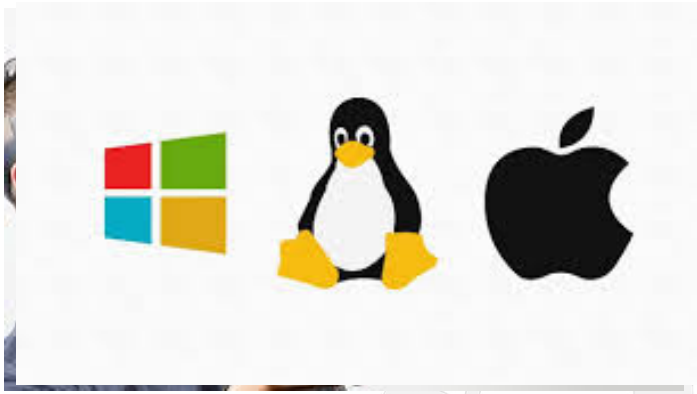
Product Factors

- **P1:** Functional features
- **P2:** User Interface
- **P3:** Product Qualities
- **P5:** Service
  - Service Features
  - Software Installation and Upgrade
  - Maintenance
- **P7:** Product costs
  - Shareware vs commercial costs
  - Licensing policies
  - Licensing costs



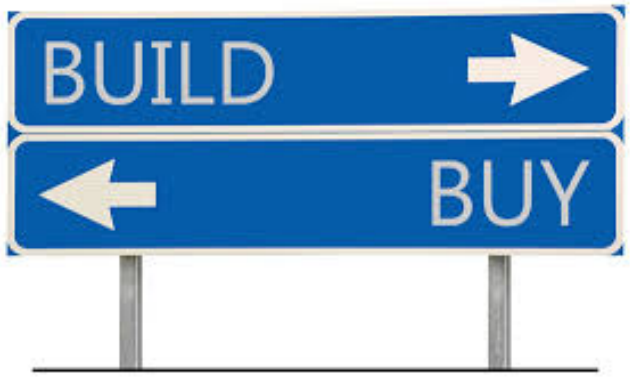
Technological Factors

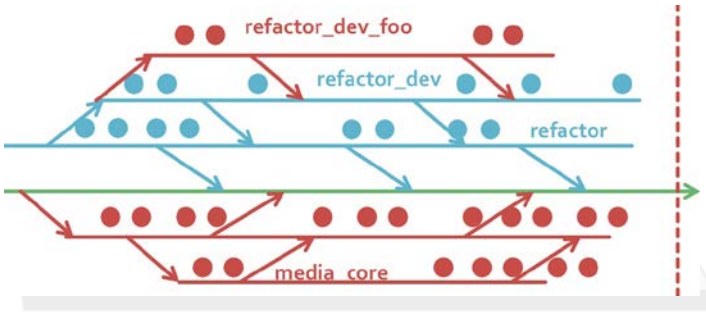
- **T1:** General Purpose Hardware
- **T2:** Domain or Project Specific Hardware
- **T3:** Software Technology
  - Operating System /s
  - User Interface
  - Software Components
  - Implementation Language
- **T4:** Standards
  - Interfaces
  - Communication
  - Coding Conventions



Organizational Factors

- **M1:** Management
  - Build vs buy
  - Schedule vs functionality
  - Environment
  - Business goals
- **M2:** Process and Development Environment
  - Development platform
  - Development process and tools
  - Configuration management process and tools
  - Testing Process and tools



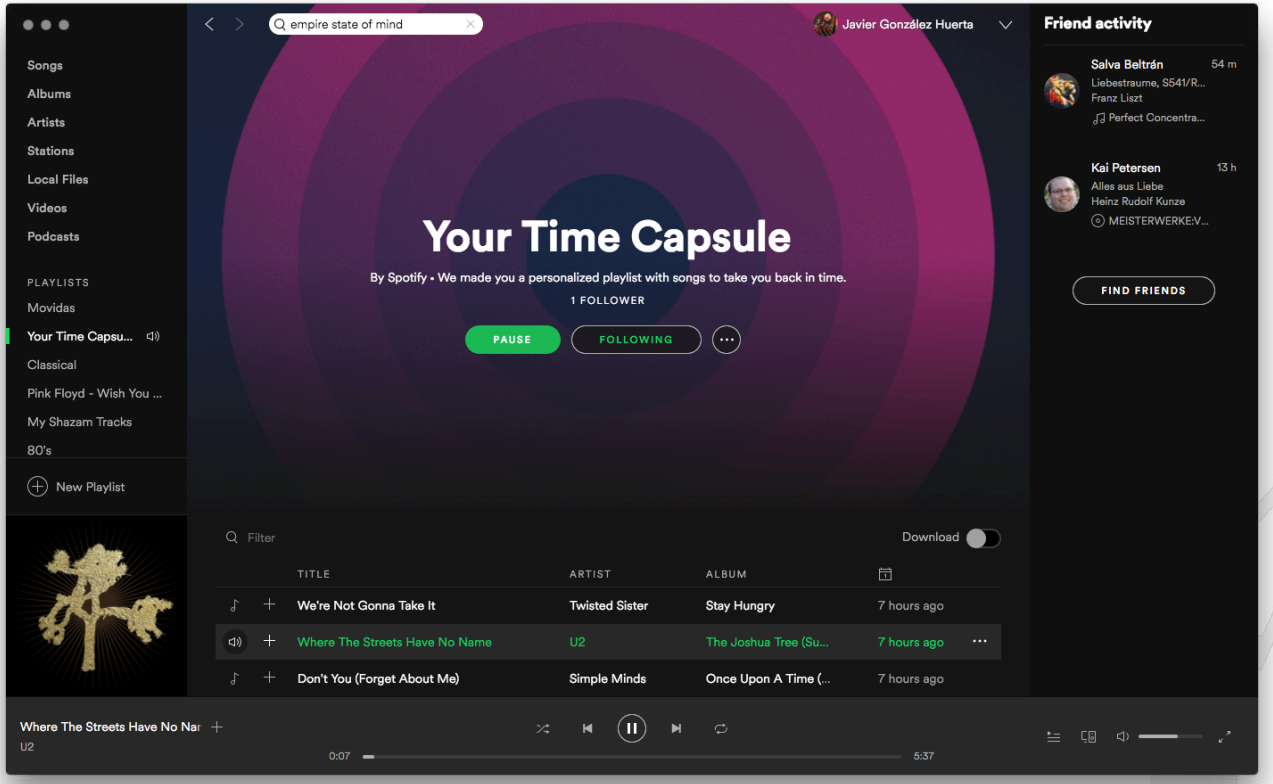


Conway's Law

- **"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."**

[1] M. E. Conway, "How do committees invent," *Datamation*, vol. 14, no. 4, pp. 28–31, 1968.

A small example

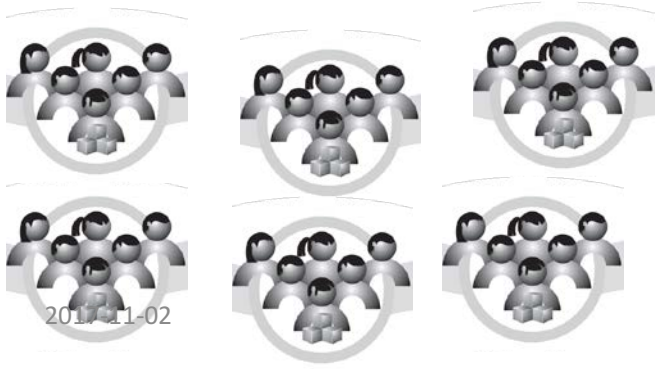


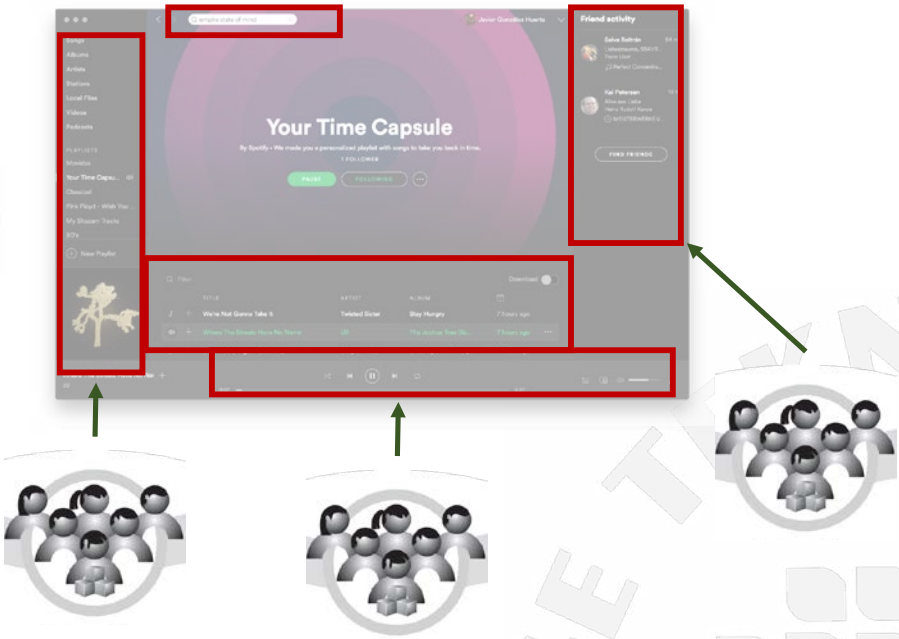
A small example: Spotify

"Monolithic" Architecture

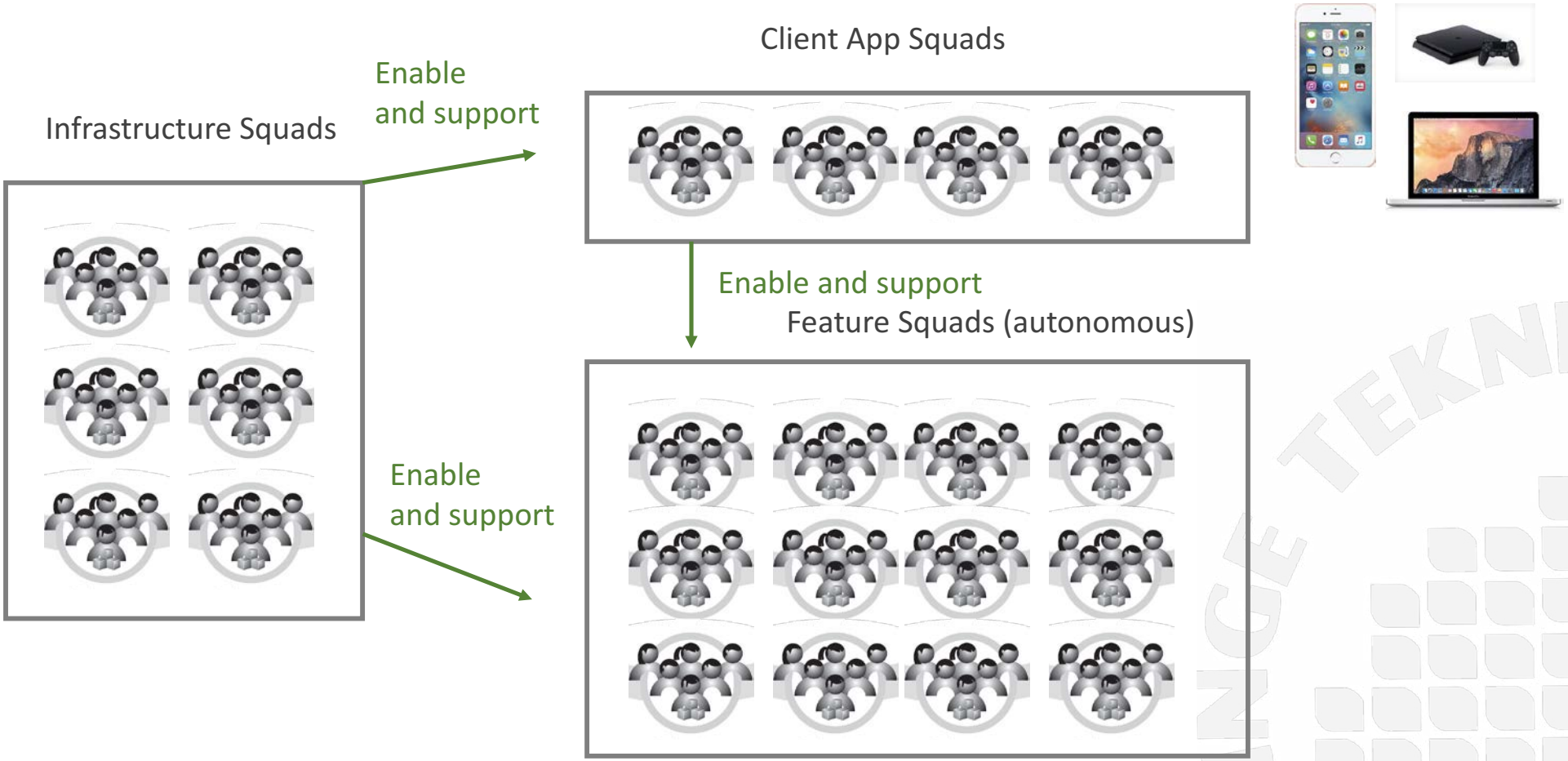
@#$&&!!!!

WTF????





A small example: Spotify



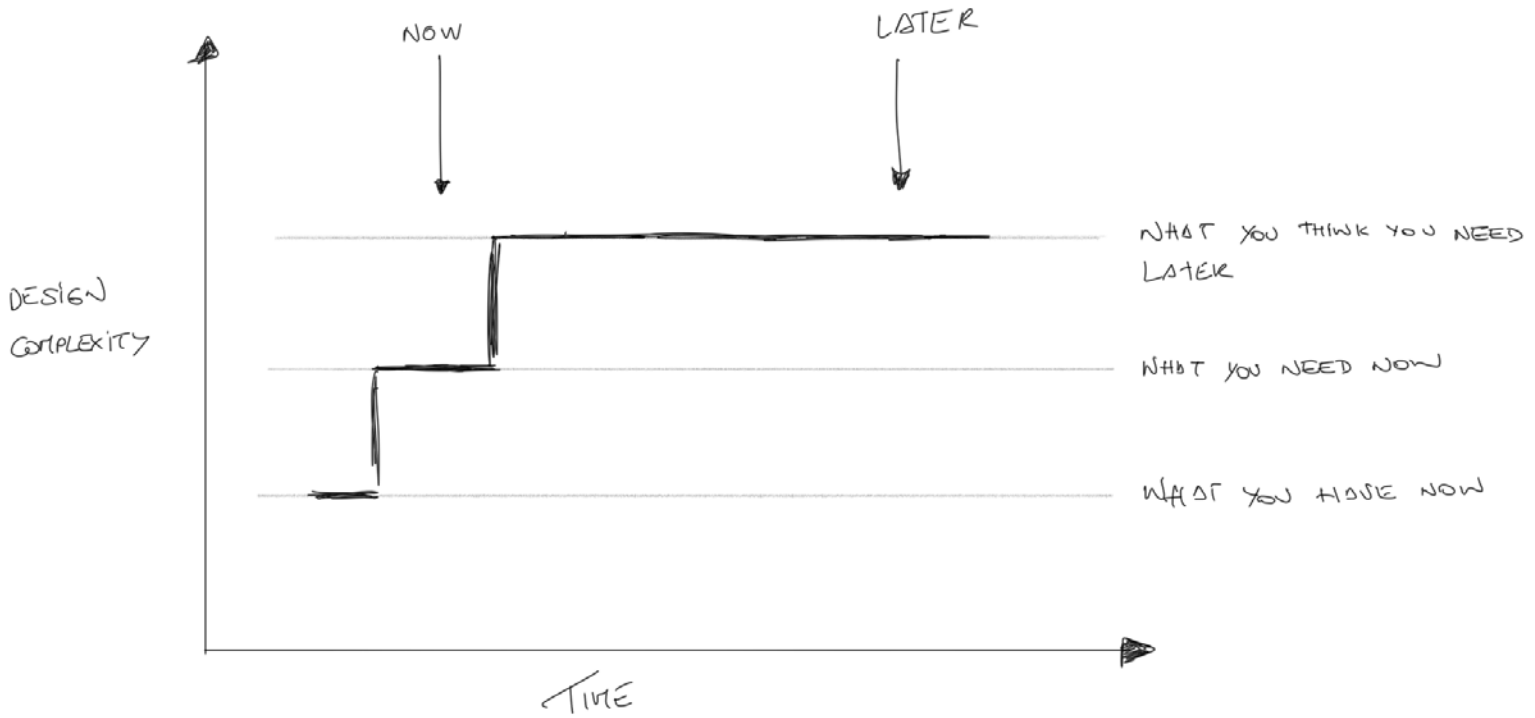
Designing and Architecting for Agility: Technical Debt

Agile Practices: Simple Design

- **Simple design:**
  - The design should be able to run all tests,
  - Allows understanding all what needs to be understood
  - No duplicated code, and fewest possible classes and methods.
  - Enough design to meet the requirements **(and no more design is done)**.
- **Refactoring:** the design of the system is continuously evolving through code refactorings carried out by the developers as soon as the improvements are found

Traditional Approaches: Design for Tomorrow

Developers are really good at trying to anticipate future problems and design for the future

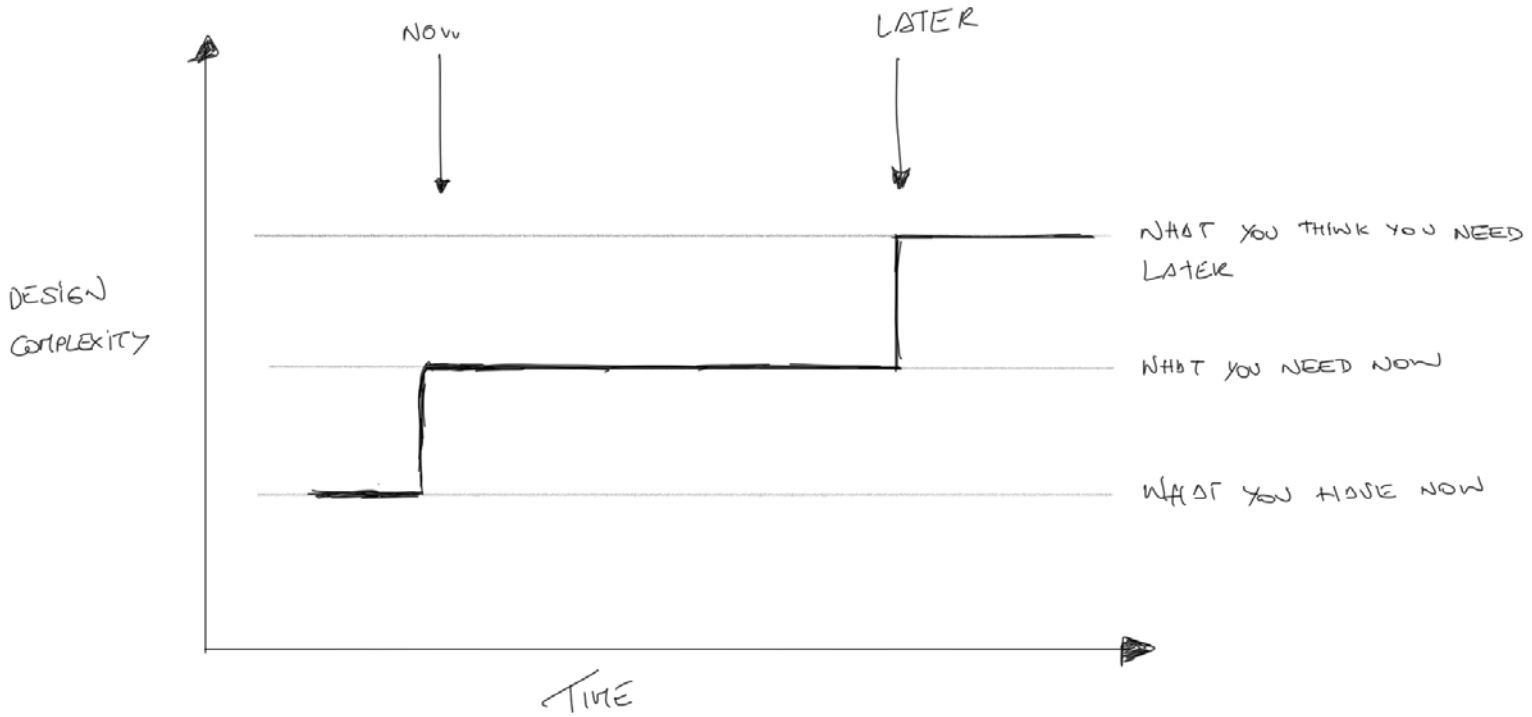


Beck, Kent, and Cynthia Andres. 2005. Extreme Programming Explained: Embrace Change (The XP Series). Addison-Wesley Professional.

Traditional Approaches: Design for Tomorrow

- The problem of this approach is the uncertainty
- What happens if tomorrow never comes?
  - May be the features we though ahead are removed from the product by the customer
  - May be we need to change something else that makes the new design not feasible
  - May be we learn how to do things better between *now* and *later*

XP (and Agile Approaches)



Beck, Kent, and Cynthia Andres. 2005. Extreme Programming Explained: Embrace Change (The XP Series). Addison-Wesley Professional.

Technical Debt

Technical Debt

- Technical Debt (TD) tries explain the long term impact of **sub-optimal decisions** made due to the **need of speed**
  - Hinders future feature development
  - Hinders agility
  - The main solution to TD (refactoring) is usually at the cost of introducing bugs or chain reactions


Wanted:
SOFTWARE ARCHITECT
SUCCESS
IDEA
PRODUCT
RESULT
WORK GROUP
STORAGE
SERVER
WORKSTATION

I have good news…

CNN Money International + Markets Economy Companies

Best Jobs in America

2015

CNNMoney/PayScale's top 100 careers with big growth, great pay and satisfying work.

Recommend 27K 1 of 100

1. Software Architect

NEXT

Median pay: $124,000
Top pay: $169,000
10-year job growth: 23%

PHOTO: JULIE FELPEL

Christopher Felpel, software architect/consultant

In the same way an architect designs a house, software architects lay out a design plan for new programs. That usually means leading a team of developers and engineers, and making sure all the pieces come together to make fully-functioning software.

What's great: New problems come up all the time and new technologies arise, making each day different, and keeping professionals in demand. "I'm pinged at least once or twice a week for new opportunities," said software architect Christopher Felpel. "There's just a lot of work out there, and that doesn't surprise me." *--Jillian Eugenios*

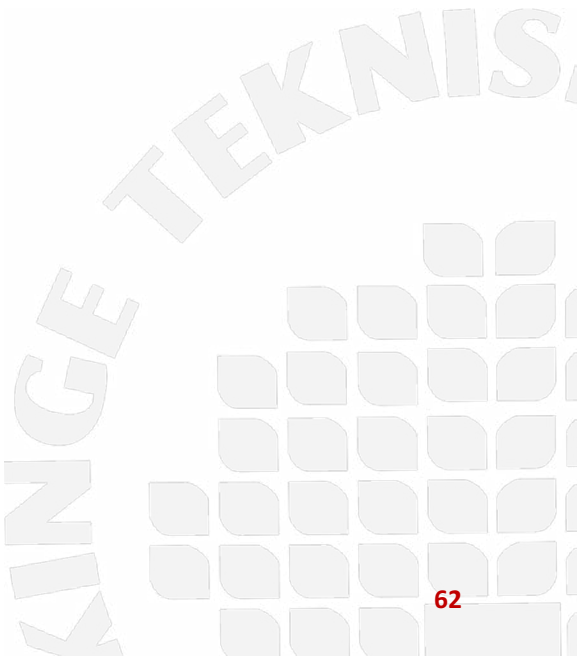
Quality of life ratings:
Personal satisfaction: **A** | Benefit to society: **B** | Telecommuting: **A** | Low stress: **A**

NEXT: Video Game Designer

http://money.cnn.com/gallery/pf/2015/01/27/best-jobs-2015/index.html

The Role of the Architect

- Bridge between business & technology
- Solid knowledge in Software Engineering and Computer Science
- Knowledge in Business Economics



Required Development Skills

Problem analysis	Software design	Programming	Testing	Deployment
• Domain modeling • Interviewing • User studies	• Object oriented design • Functional design • Apply design patterns	• Programming multiple paradigms • Networking • Databases • Execution platforms • Debugging • Unit-testing • Source code management • Code reviewing	• Test analysis • Automatic testing • Code analysis	• Software upgrade (without down-time) • Software installing • Trouble-shooting

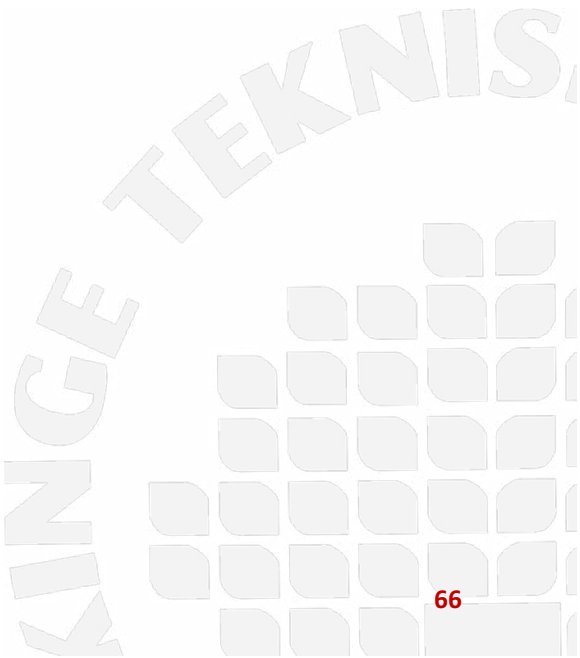
Required Business Skills

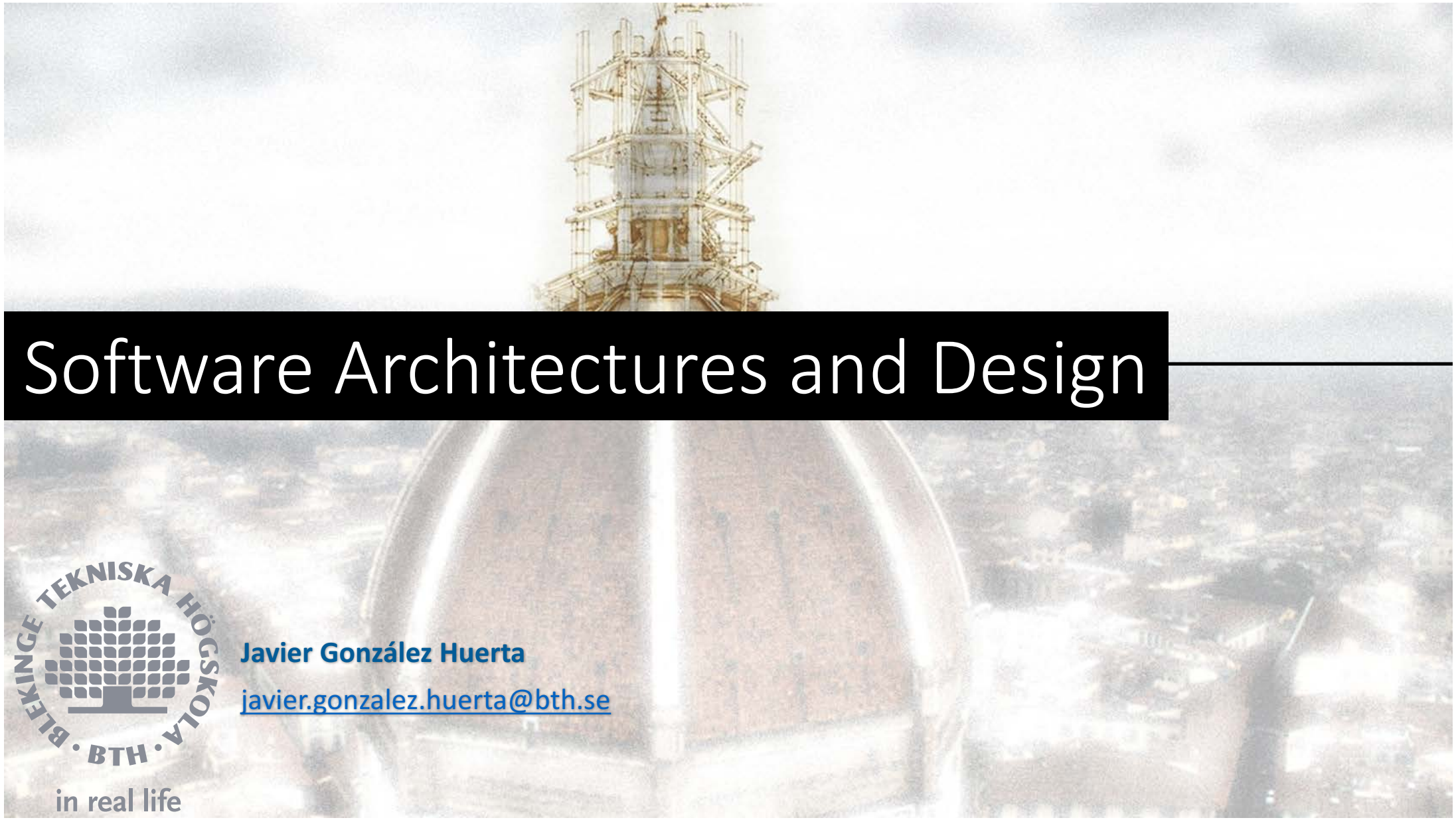
Project Management	Business	Leadership	Communication	Interpersonal	Work
• Domain modeling • Interviewing • User studies	• Economics • Business Administration • Identifying Business Goals • Develop capabilities • Technical / architecture road map	• Guiding apprentices • Mentoring • Act through others	• Present technical vision • Present product roadmap • Listen, interview, consult, negotiate • Promote simplicity	• Provide feedback • Respect • Conflict resolution	• Decision Making • Workload Management

And now?

Next Steps

- In the course we will focus more in general-purpose software architectures
- We will learn how to analyze a system and identify:
  - Architectural bottlenecks
  - Shortcomings and opportunities for improvement





Software Architectures and Design

Javier González Huerta

javier.gonzalez.huerta@bth.se

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

in real life