Architecture

By Glib Kutepov

Glib.kutepov@iese.fraunhofer.de

Outline

1. Why Architecture?
2. What is Architecture?
3. How to create an Architecture?
    - Alignment
    - Modeling and Structuring
        - Architectural Views
        - System decomposition
        - Design for quality

--- Why Architecture? ---

What is the mission of Software Engineering?

- Create a
  - **Large-scale**
  - **Long-living**
- software system, with the:
  - Necessary **functionality**
  - Adequate **quality**
  - Adequate **cost**
- in the
  - Adequate **time**

What are the challenges?

- **Complexity** of the system itself
- **Interconnection** of the systems (e.g. SAP R/3)
- Continuous **change** of the system
- **Distributed** development processes
- **Technology** choice

What are the reasons?

1. Complex **functionality**
2. Importance of **quality**
    - Time to market
    - Maintainability
    - Fault tolerance
    - …

[Source: P. Clements, Software Product Line Architectures, Presentation at RiSE Summer School on Software Reuse]

What is the solution?

- Develop your system **architecture centric**
- In your architecture:
    - Take care of crucial challenges
    - Find appropriate solutions for these challenges

--- What is Architecture? ---

Architecture is about

- **Alignment**
  - Adjustment/Optimization of the relationship
  - between Business and Technology
  - between Requirements and Solutions
- **Modeling** and **Structuring** in the software system
  - Manage complexity (separation of concerns)
  - Blueprint for the development

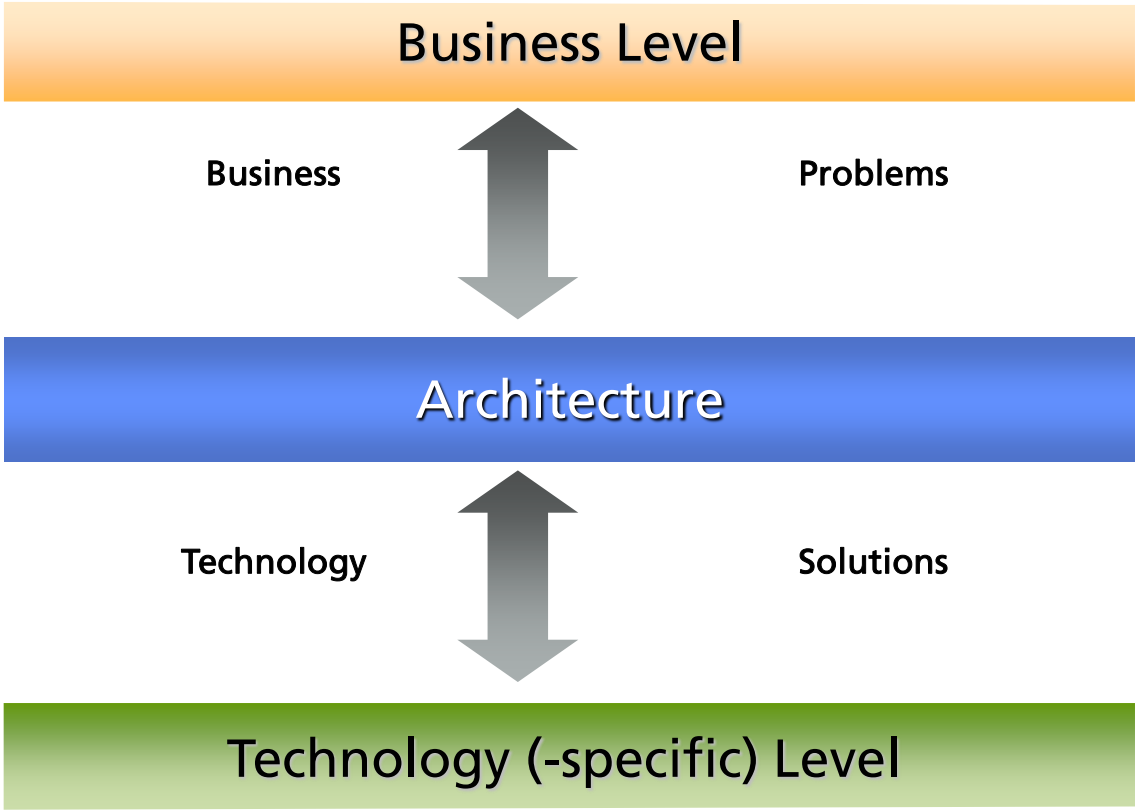
Architecture is

- **Tool** for
  - **Description** of the decisions made
  - **Prescription** of the way things should be implemented
  - **Prediction** of the future properties of the system
- Set of **activities**
  - Which you carry out during the project lifecycle
  - Customer interviews, design, documentation etc.
- Set of **artifacts**
  - Which you produce during the project lifecycle
  - Documents, wikis, POCs etc.

--- How to create an architecture?---

---Take care of Alignment---

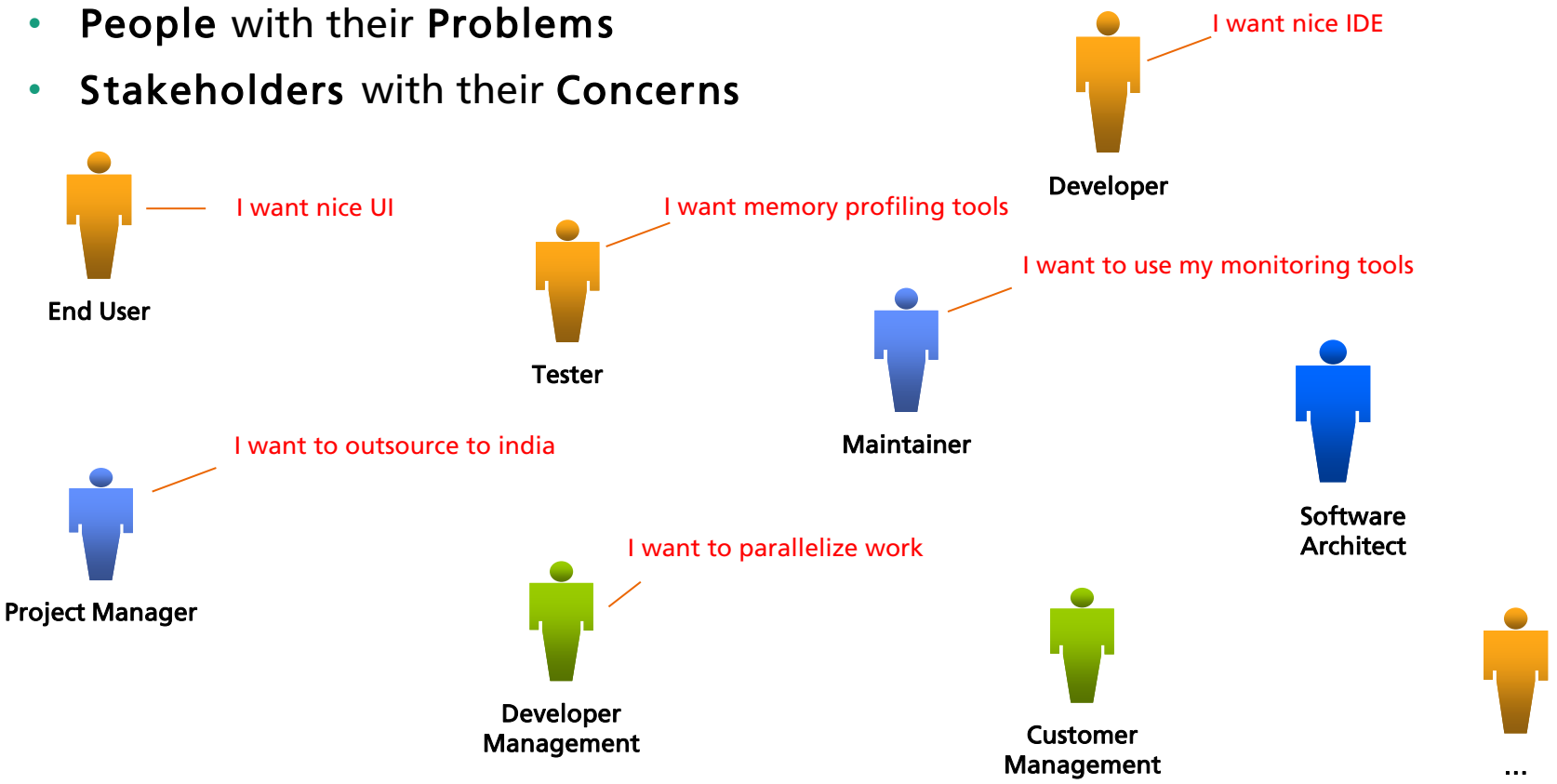
Alignment



How good are the solutions?

Business Level

- **People** with their **Problems**
- **Stakeholders** with their **Concerns**

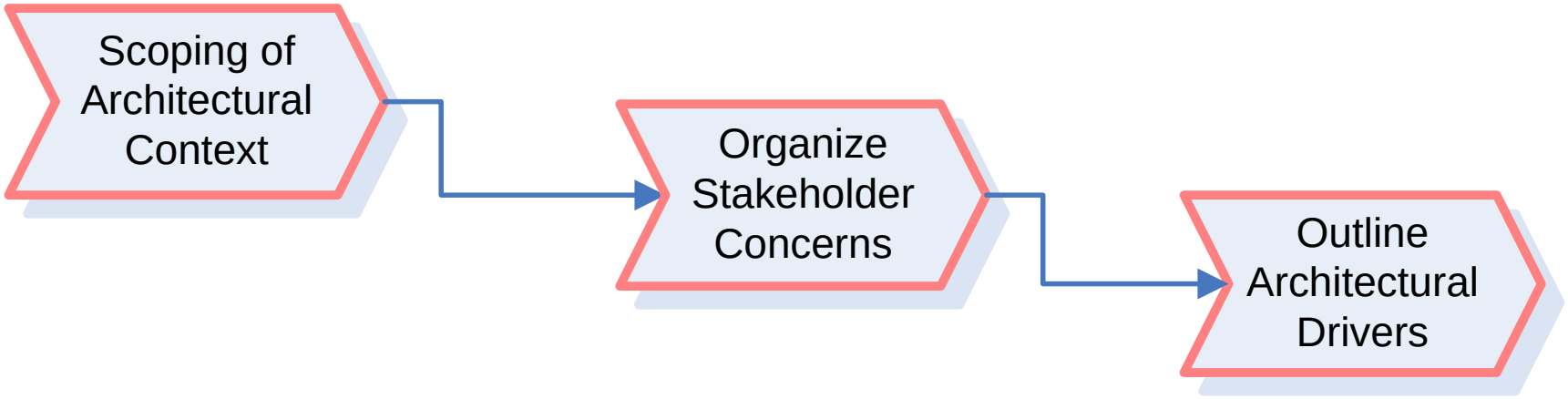


Concerns/Architectural Drivers

- **Concerns** – things which **drive** your **architecture**
- They come from Stakeholders
- **Identify** Stakeholders
- Find out their **concerns**

- Start with **Stakeholder Analysis…**

Analyzing Stakeholders – Overview



--- Step 1: Scope Architectural Context ---

Why Scoping the Architectural Context?

- Know where to stop
- Without a clear scope
    - You might promise too much
        - Hard to reverse later
        - Hard to keep to
        - Unnecessary things
    - You will waste **time**
        - Of your team
        - Of your customer

Scoping of the Architectural Context

1 – Capture common vocabulary

- Identify common terms (Glossary)

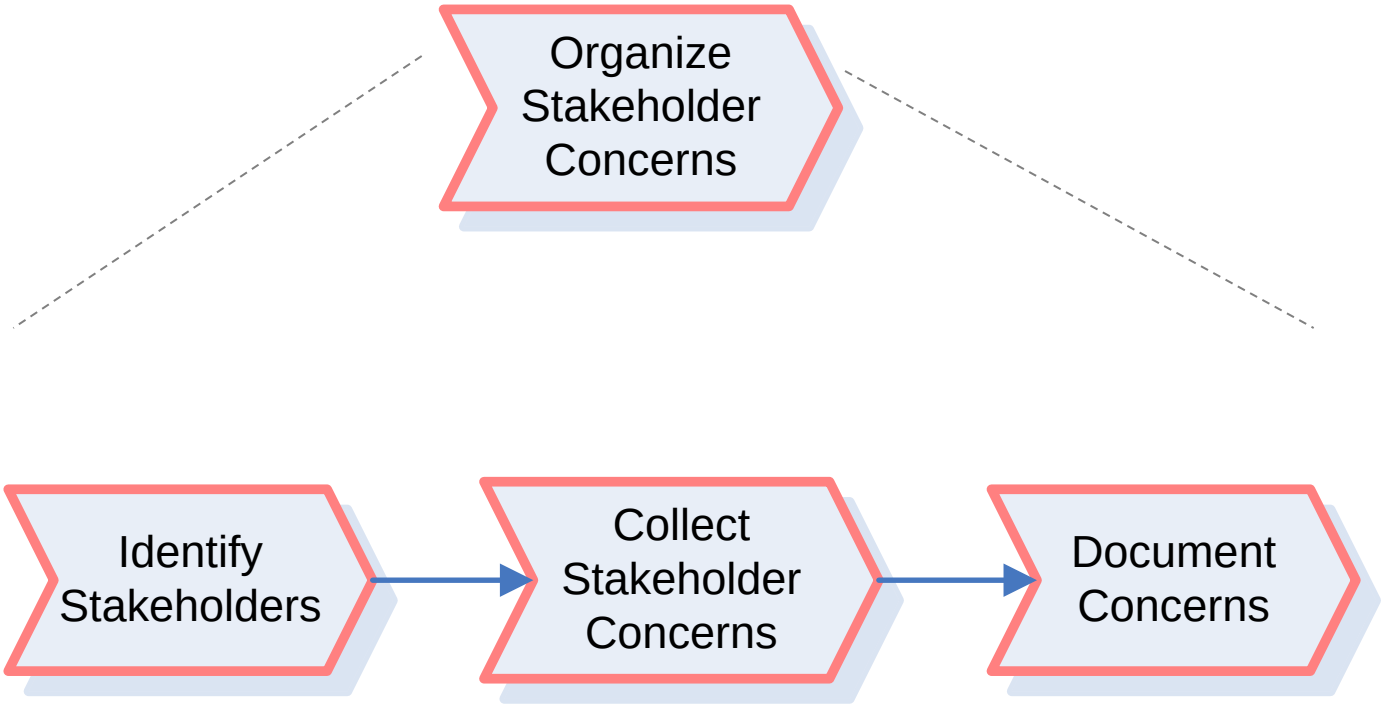
2 – Identify main business goals

3 – Determine architecture scope

- Scope of functional areas to be provided by the architecture
- External systems to interface with
- Identify constraints

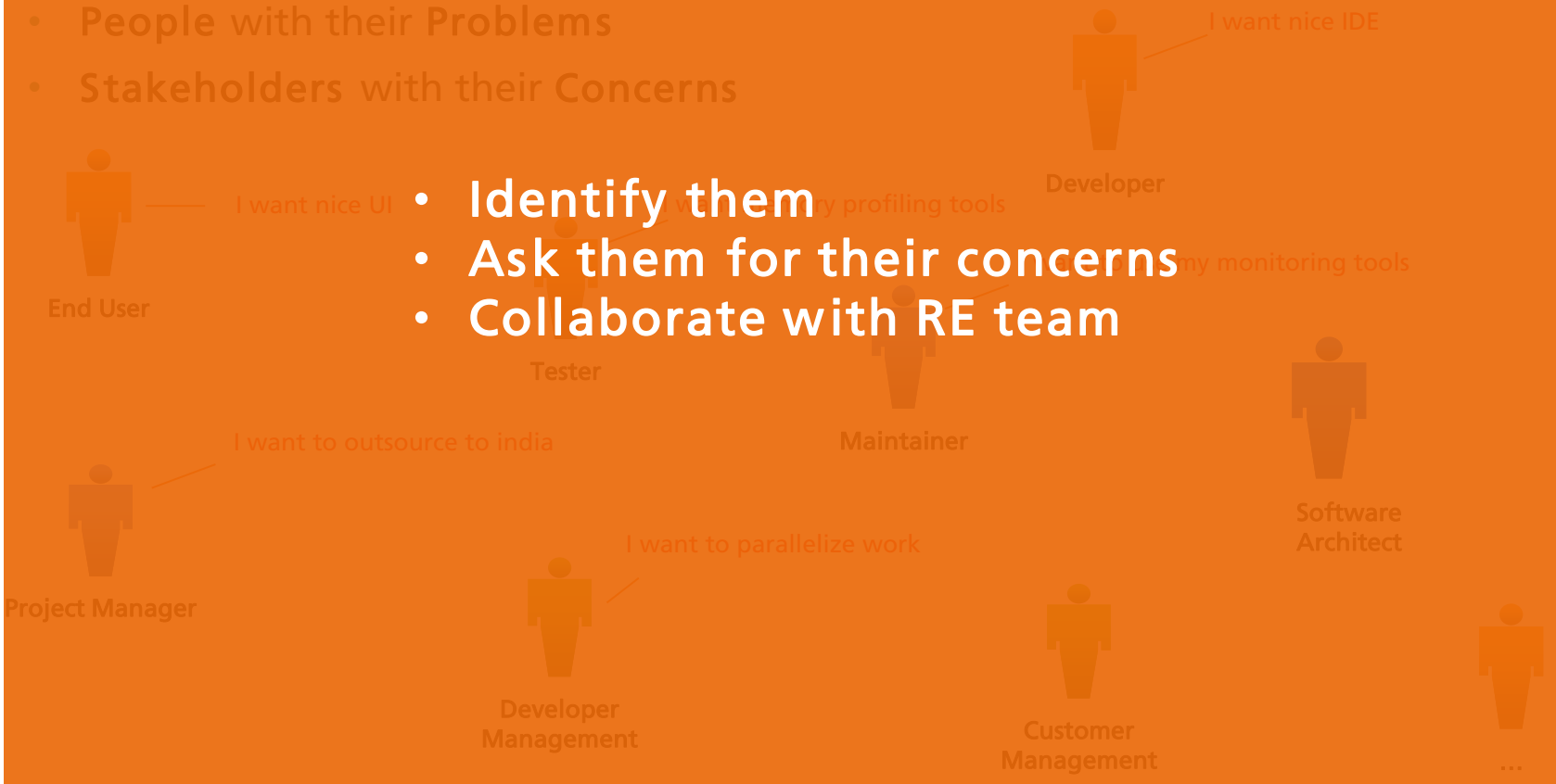
--- Step 2: Organize Stakeholder Concerns ---

Organize Stakeholder Concerns – Overview



Business Level

- People with their Problems
- Stakeholders with their Concerns



- **Identify them**
- **Ask them for their concerns**
- **Collaborate with RE team**

What are Architectural Concerns

- **Business goals**
  - Develop a line of products, tightly integrate with SAP etc.
- **Quality attributes**
  - Banking application without a password
  - Runtime: Performance, Security
  - Development time: Maintainability, Accountability
- **Key functional requirements**
  - What is system supposed to do?
  - Only the ones which influence architecture
- **Constraints**
  - Organizational and technical (e.g. use ISO XML, keep data in DE)
  - Cost and time

How to Document Concerns?

- Use **Architectural Scenarios** for
  - Quality Attributes
  - Functional Requirements
- You may use **text** for
  - Business goals
  - Constraints

- **What are the architectural scenarios?**

"A situation which the system must be capable to cope with in order to satisfy the specific quality attribute or architectural requirement"

Why Architectural Scenarios?

- You need to make things price enough
- There is no standard meaning of what it **means to be "secure"**
- If you don't know what is **"secure"** you can't check if it is **"secure"**
- Scenarios help us to avoid all of these problems
- They are **precise** and **verifiable**

Types of Scenarios

- **Functional** Scenarios
  - Show functional executions of the system
  - Might be Use Case Scenarios
- System **Quality** Scenarios/Quality Attribute Scenarios
  - Focusing on the non-functional aspects of a system
  - Are a means for formalization

Documenting Architectural Scenarios

Scenario	*Name of scenario*
Quality	*Related quality attribute*
Environment	*Context applying to this scenario*
Stimulus	*The event or condition arising from this scenario*
Response	*The expected reaction of the system to the scenario event*
Response Measure	*The measurable effects showing if the scenario is fulfilled by the architecture*

Example

Scenario	*Extension #1 new Device at Runtime*
Quality	*Flexibility (Extensibility)*
Environment	*The system has been delivered with 3 devices, The system is in operation (normal)*
Stimulus	*A new device is to be integrated*
Response	*The system recognizes the new device and installs it in order to be used by the applications [up to 10 devices]*
Response Measure	*(Down-time of the system or its services +) Installation time of the new device is below 1 min*

Usages of Scenarios

- Input to the architecture definition process
- Evaluating the architecture
- Communication with stakeholders
- Finding missing requirements

--- Step 3: Outline Architectural Drivers ---

Outline Architectural Drivers/Concerns

- Prioritize!
  - Wish ^= need
  - Importance
  - Risks
  - Make tradeoffs
  - Say no

Recall your state

- You have identified, precisely defined and document architectural drivers:
  - Business goals
  - Functional requirements (via Scenarios)
  - Quality requirements (via Scenarios)
  - Concerns
- Now you need to **find** and **express** your solutions
- Let’s start with the expresion…

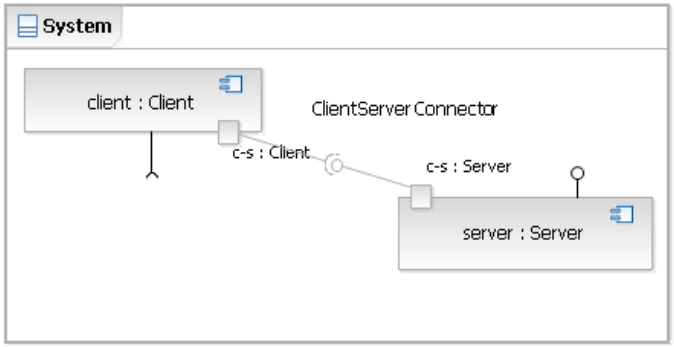
---Take care of Modeling and Structuring---

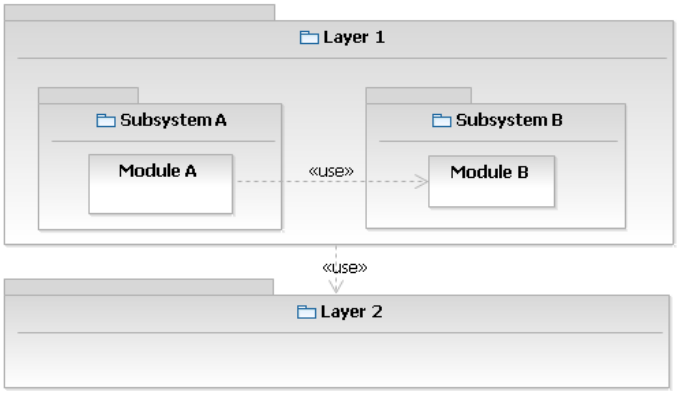
"Software *architecture* is the structure or structures of the system, which comprise software elements, the externally visible properties of these elements, and the relationships among them."

[Bass et al., SA in practice]

Modeling and Structuring Because

- You need to **model** your system in order to manage complexity
  - **Abstraction**
- Your **model** is the set of **structures**
- You express your design decisions in **structures**
  - What elements are there?
  - What are they for and what do they expose?
  - How do they work together?





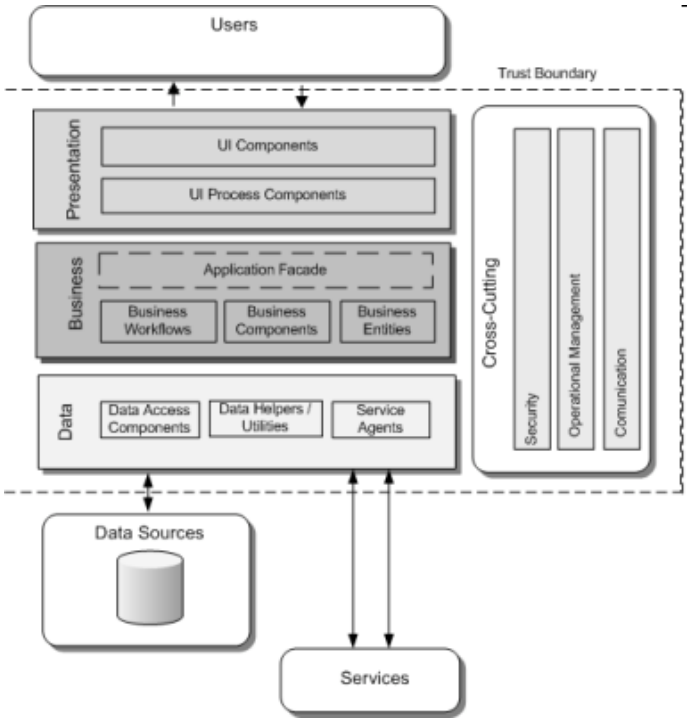
How many structures?

- No single structure expresses the architecture of the system
  - Too complex
  - Doesn't resolve the complexity problem
- **Separation of concerns**
- Use structures to express particular aspects of the system:
  - System @ runtime
    - Elements: Components, processes, computational nodes, …
    - Relationships: Calls, is executed by, …
  - System @ development time
    - Elements: Module, class, development team, …
    - Relationships: Is decomposed into, is developed by, …

Structure 1

- Commonly used structure
  - Layered architecture

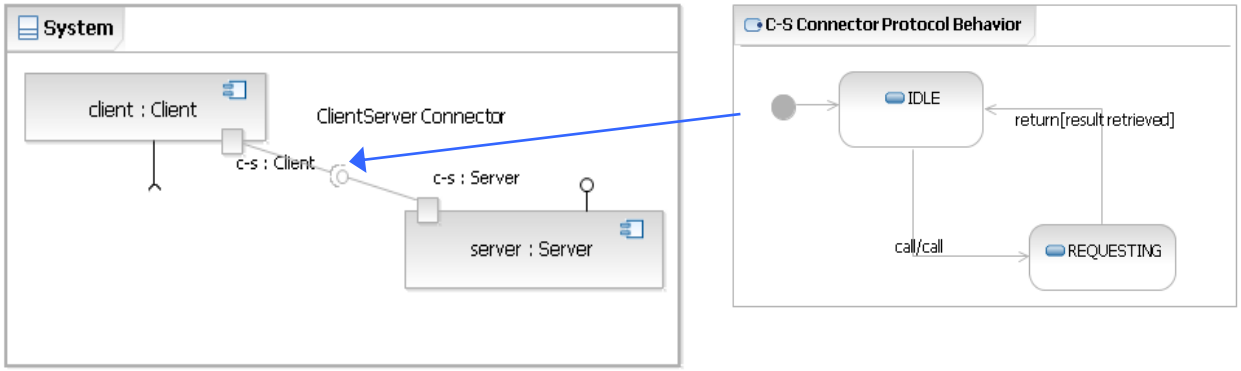
- That's not enough
  - Meaning of boxes and lines?
  - Other structures needed



[Source: Microsoft]

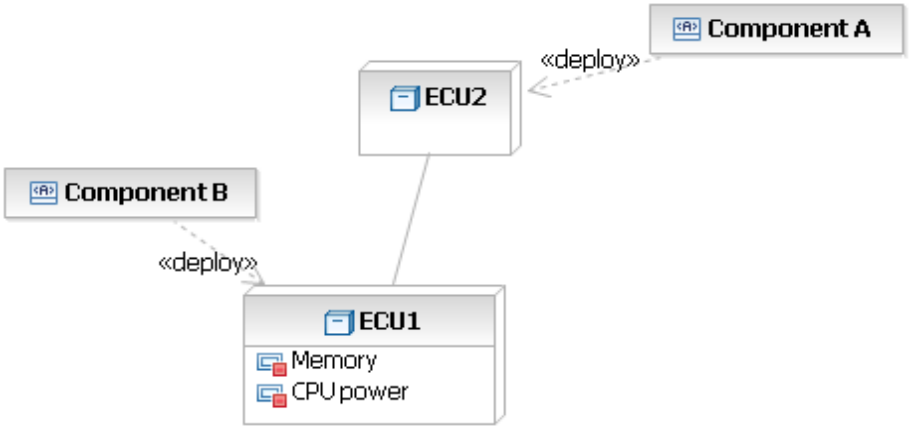
Structure 2

- **Behavior**
  - How is functionality realized by components
  - How do components communicate?



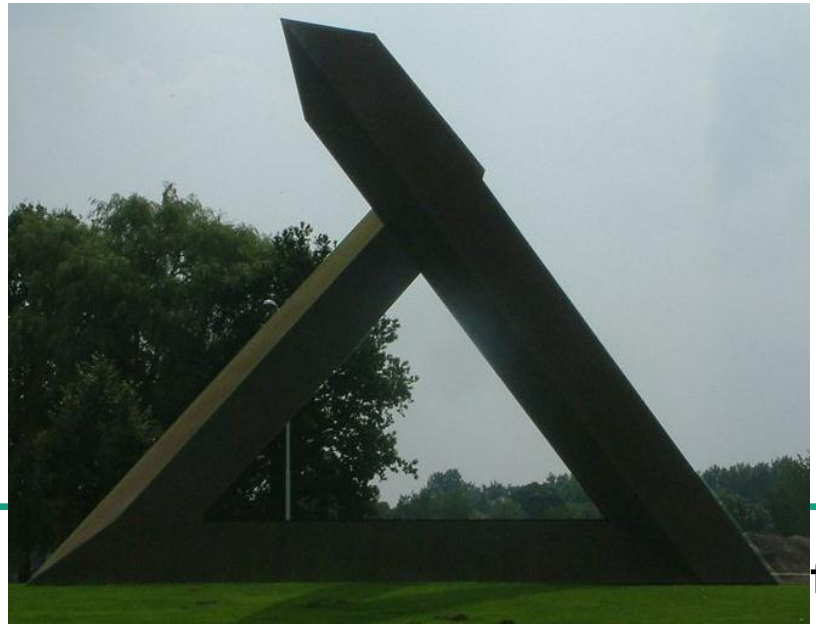
Structure 3

- Allocation
  - How are software elements allocated to hardware elements?
  - How are development tasks allocated to development teams?

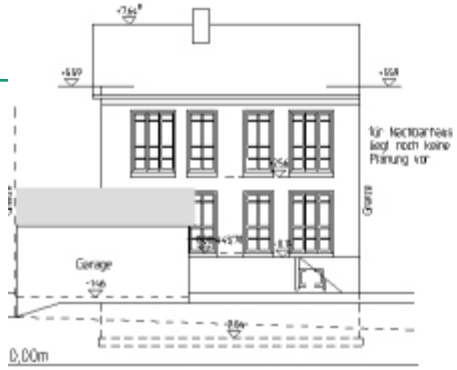


---Represent Structures with Views---

Sculpture “SWING” by Arie Berkulin

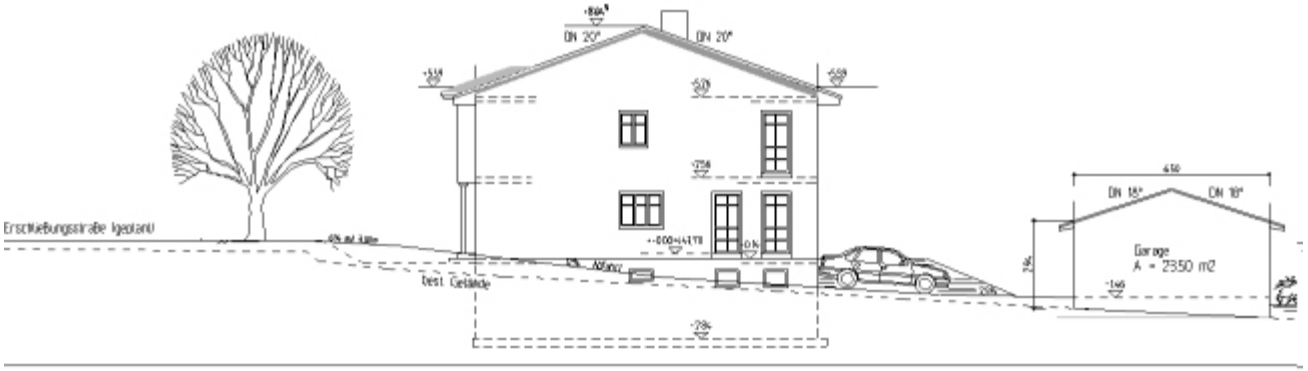


View 1



ANSICHT AUS OSTEN (GARAGE)

ANSICHT AUS WESTEN

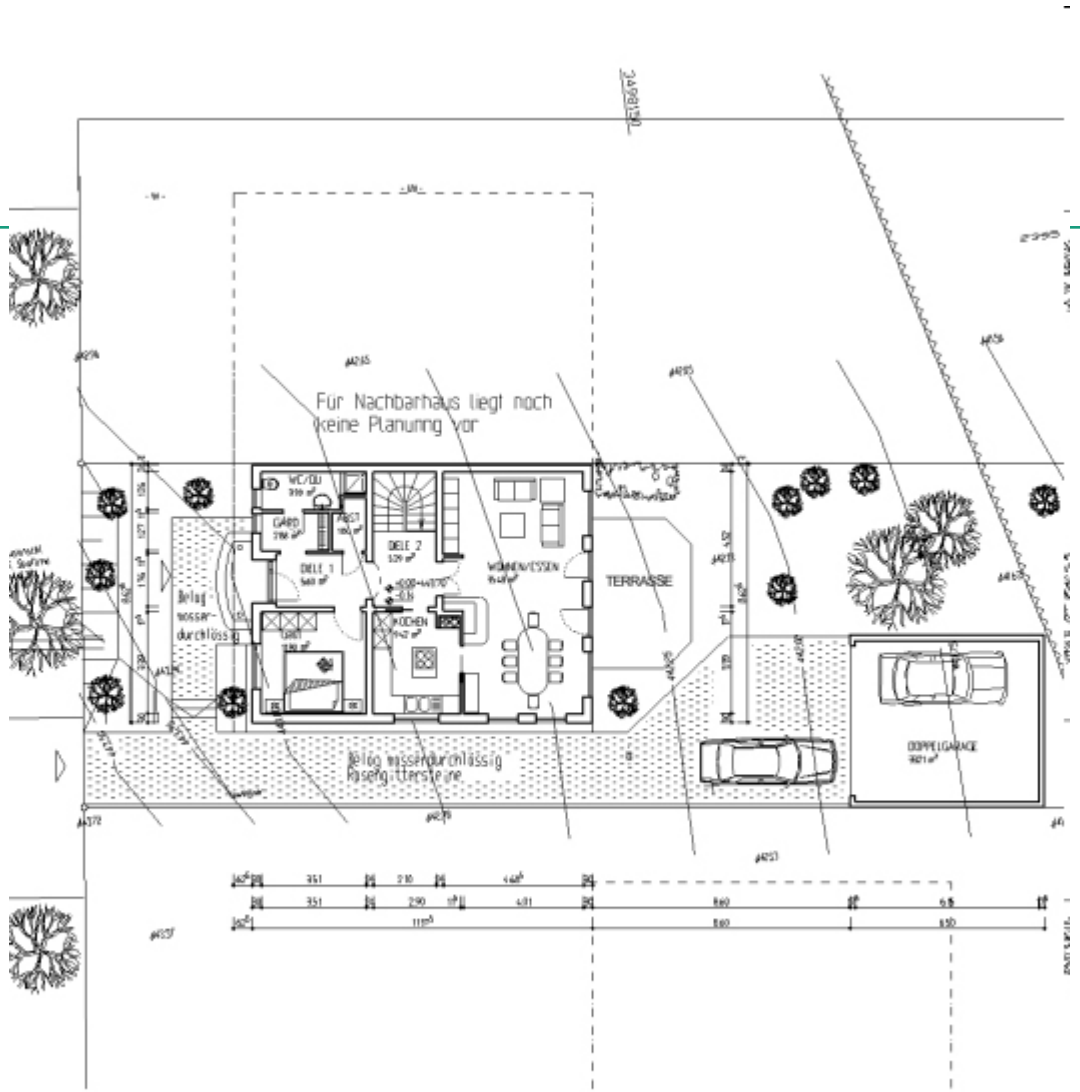


ANSICHT AUS SÜDEN

http://www.planungswerkstatt-bau.de

Fraunhofer IESE

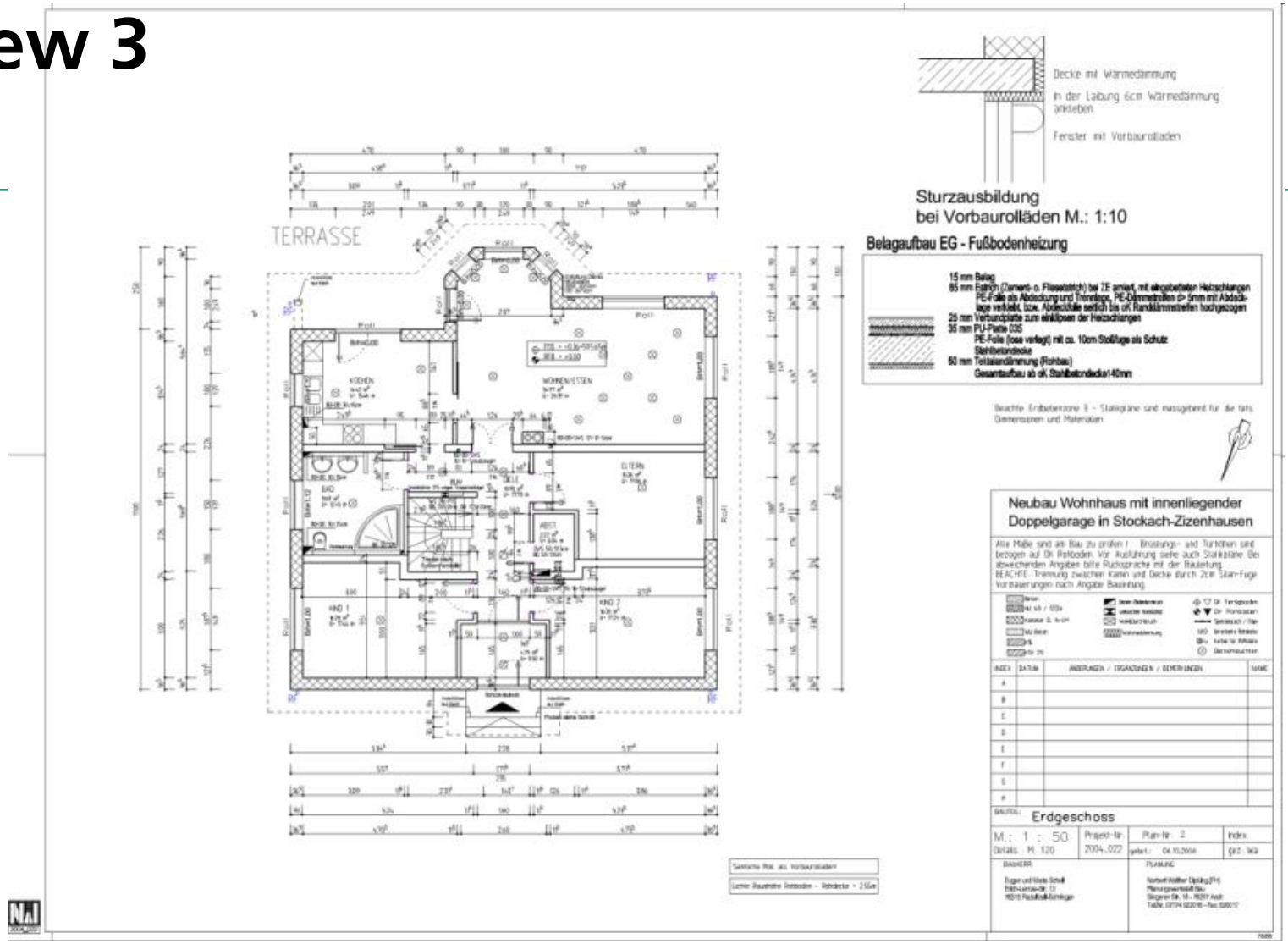
View 2



http://www.planungswerkstatt-bau.de

Fraunhofer IESE

View 3



http://www.planungswerkstatt-bau.de

Fraunhofer IESE

Benefits of Views

- Separation of Concerns
  - Separate Analyses
- Complexity reduction
  - Focus on a particular concern only
- Communication with Stakeholders
  - Adjust the language
  - Especially developers … focus on implementation relevant portions

Which Views?

- **No general** set of views
- Architect selects **useful** views
- **Depends** on
    - The **system**
    - The **stakeholders**
    - Their **concerns**
    - How they will **use** the documentation

"When developing a view, be clear in your mind what sorts of concerns the view is addressing, what types of architectural elements it presents, and who the viewpoint is aimed at. Make sure that your stakeholders understand these as well"

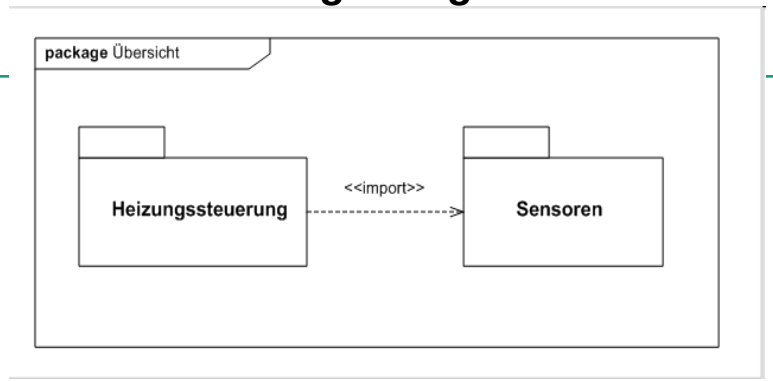
[Rozanski, Woods, 2006]

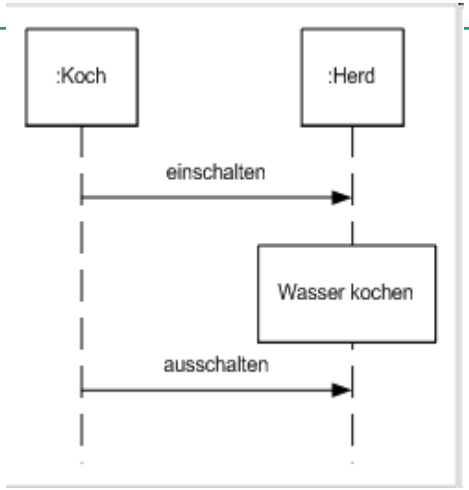
Standard Views

- System or Context View
- Component & Connector View
    - Structural Part
    - Behavioral Part
- Data View
- Module and Implementation View
- Deployment View
- Team-Assignment View

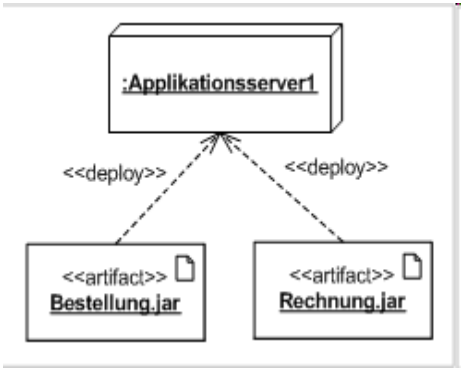
Represent Views with UML

Package Diagrams

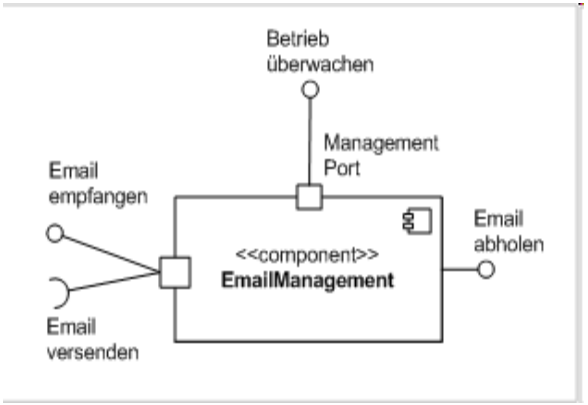




Sequence Diagrams



Deployment Diagrams



Component Diagrams

Fraunhofer IESE

UML and Views

- System or Context View: **Deployment Diagram**
- Component & Connector View
  - Structural Part: **Component Diagram**
  - Behavioral Part: **Sequence Diagram**
- Data View: **Class Diagram /Component Diagram**
- Module and Implementation View: **Package Diagram / Class Diagram**
- Deployment View: **Deployment Diagram**

Recall your state

- You have identified, precisely defined and document architectural drivers:
  - Business goals
  - Functional requirements (via Scenarios)
  - Quality requirements (via Scenarios)
  - Concerns
- You know how you express your solutions
- Now you need to know how you **find** the solutions

--- Decompose your System based on Functionality ---

Decomposition

- … is context-sensitive
- … in practice, systems are typically decomposed driven by **functionality**
- … not much guidance possible

Functional (De-)Composition

- Goal of decomposition: **Reduce coupling** of elements
  - E.g. Use Adapters for communication
- Goal of composition: **Increase semantic cohesion** btw. Elements
  - Elements that change together are packaged together
  - Elements that are used together are packaged together
  - E.G. Flight Manager, SAP Adapter etc.

Decomposition Steps

1. Identify **elements**
    - Identify responsibilities
    - Identify unique, self-contained roles/elements
    - Identify types of composed/aggregated elements

2. Identify **dependencies**
    - Identify data/information exchange requirements
    - Identify interfaces
    - Identify dependency types (The dependency graph of elements must have no cycles)

--- Add Quality Aspects ---

Addressing Quality

- You need a solutions for your architectural scenarios, don't you?
- You can apply your own solutions or…
- …use proven solutions - **patterns**
  - **Solution** to the **Problem** in the **Context**

- Patterns can be classified according to the decomposition dimensions
  - **Development-time** Patterns (e.g. MVC)
  - **Runtime** patterns (e.g. Job Scheduling, Heart-beat)
- **Know** them, **use** them!

--- Use Tools ---

Tool Support for Architecture Modeling

- First use paper
- Then use **Enterprise Architect** – UML Modeling tool
- Use word for documenting things

Recall your state

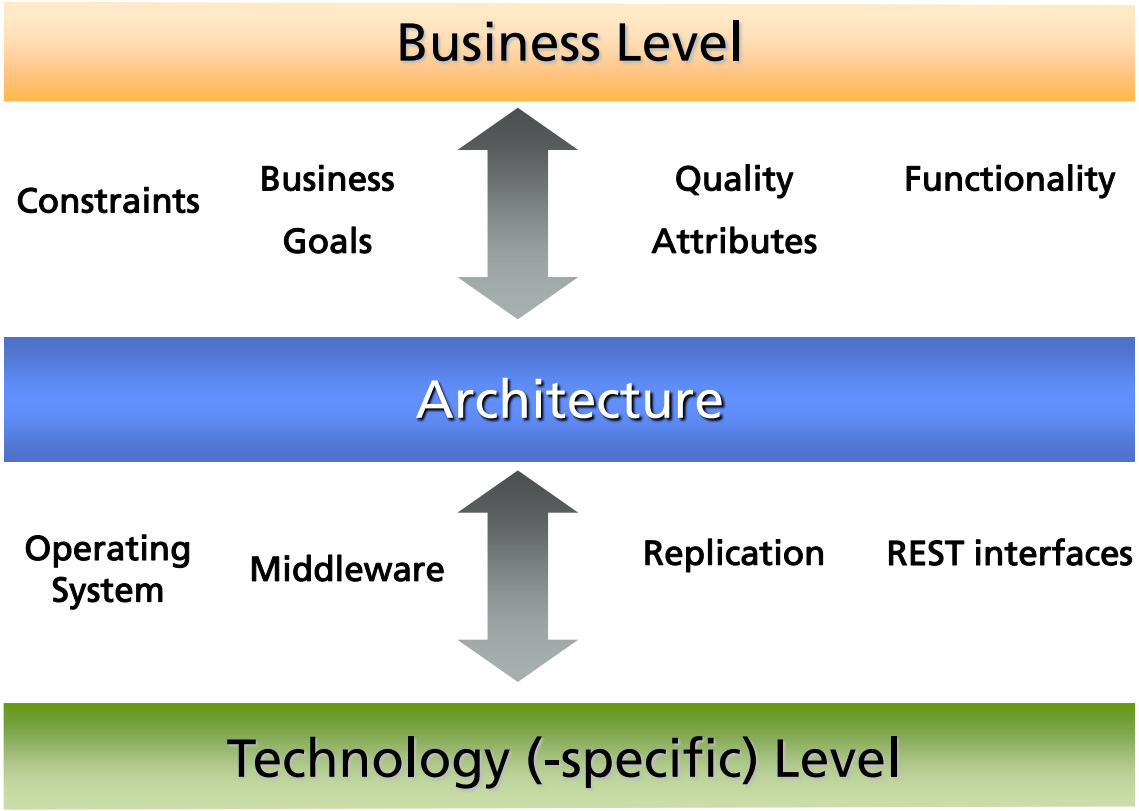
- You have identified, precisely defined and document architectural drivers:
  - Business goals
  - Functional requirements (via Scenarios)
  - Quality requirements (via Scenarios)
  - Concerns
- You know how do you express your solutions
- You have found the solutions
- You have expressed and documented your solutions

--- Are you done?---

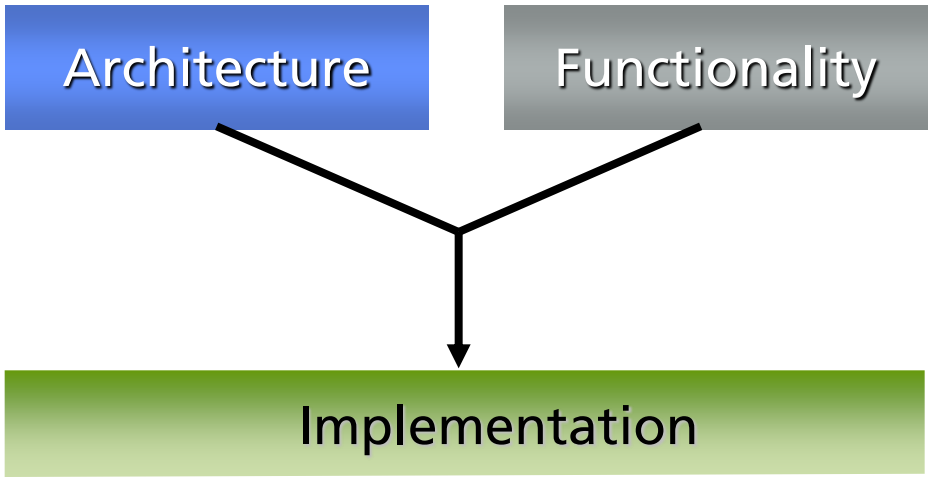
Hard question…

- You are done when
  - Key requirements are clear and addressed
  - You have confidence that they can be achieved
  - The structure(s) of the system are clear
  - You can start Implementing:
    - assign work units to developers
    - control the parallel development and integration

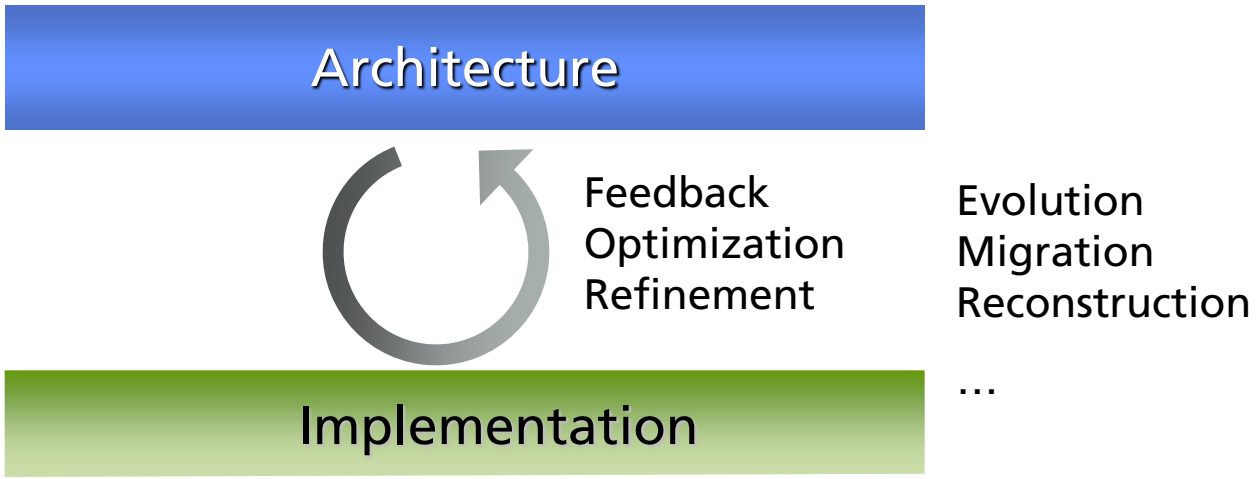
You have done Alignment, Structuring and Modeling



You can start your implementation



In the real world the story only starts…



So, be brave!

Thank you