Fraunhofer
IESE

Architecture Tutorial – Capstone 2019

Rodrigo Falcão

28.09.2018
IESE

Outline

- What is Architecture?
- How to create an Architecture?
  - Stakeholders and project settings
  - Architecture significant requirements
  - Design and Modeling
  - Documentation
  - Prediction and Control
- Organizational Aspects

What is Architecture?

Software Architecture Definitions

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

[Software Architecture in Practice, L.Bass, P.Clements, R.Kazman]

- Software architecture is the fundamental **concepts or properties of a system** in its **environment** embodied in its **elements**, **relationship**s, and in the **principles** of its **design and evolution**.

[Systems and software engineering — Architecture description, ISO Standard 42010]

- Software architecture is the **set of design decisions** which, if made incorrectly, may cause your project to be cancelled.

[E. Woods]

- Software architecture is the set of **principal design decisions** made about the **system**.

[Software Architecture: Foundations, Theory, and Practice , E.Dashofy, N.Medvidovic, R. Taylor.]

Architecting vs. Architecture

Activities

Design
Modeling
Communication
Negotiation

Artefacts

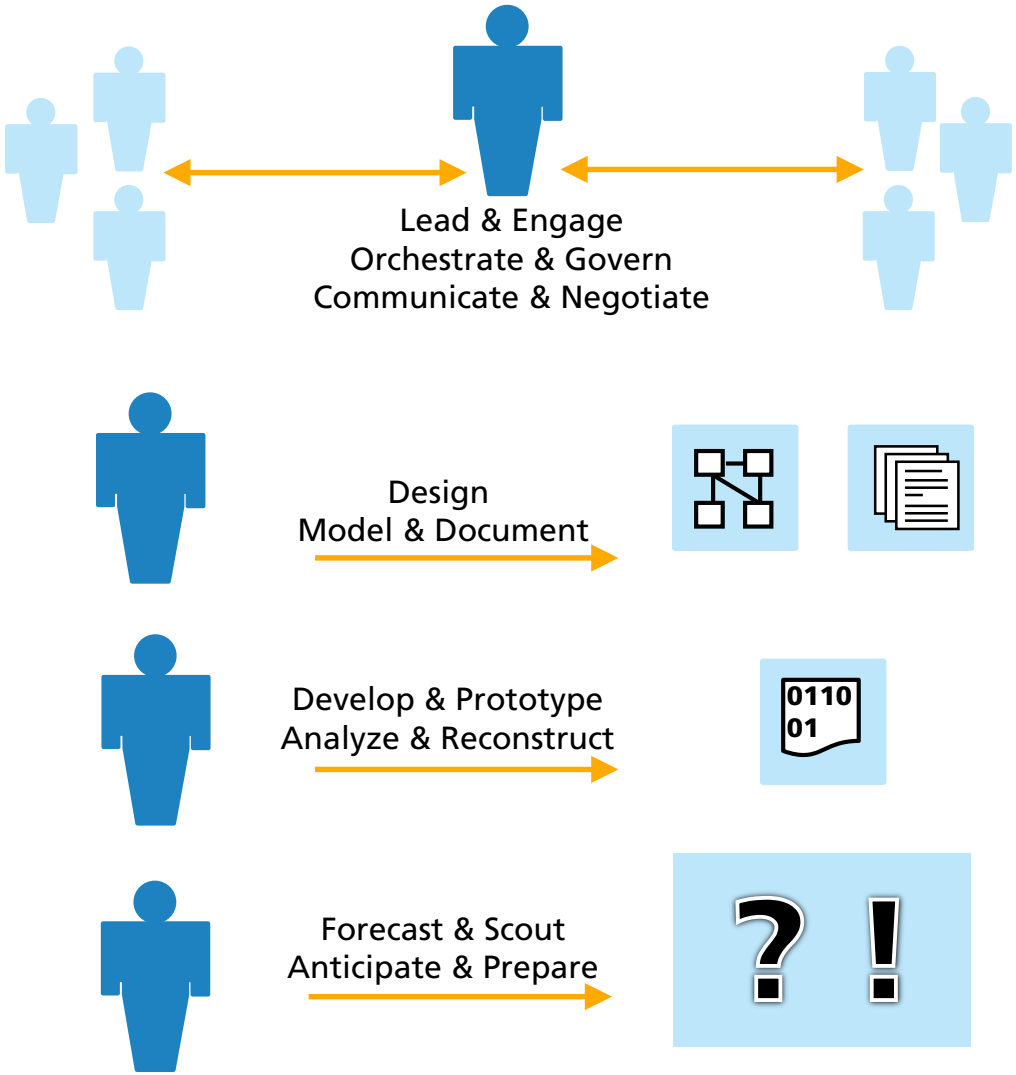
Design Decisions
Blueprints & Models
Documentation
Implemented Decisions

Source:
http://www.karl-gotsch.de
http://blogs.artinfo.com/objectlessons/files/2012/09/0723-architect_sm.jpeg

Architectures: The Artifact

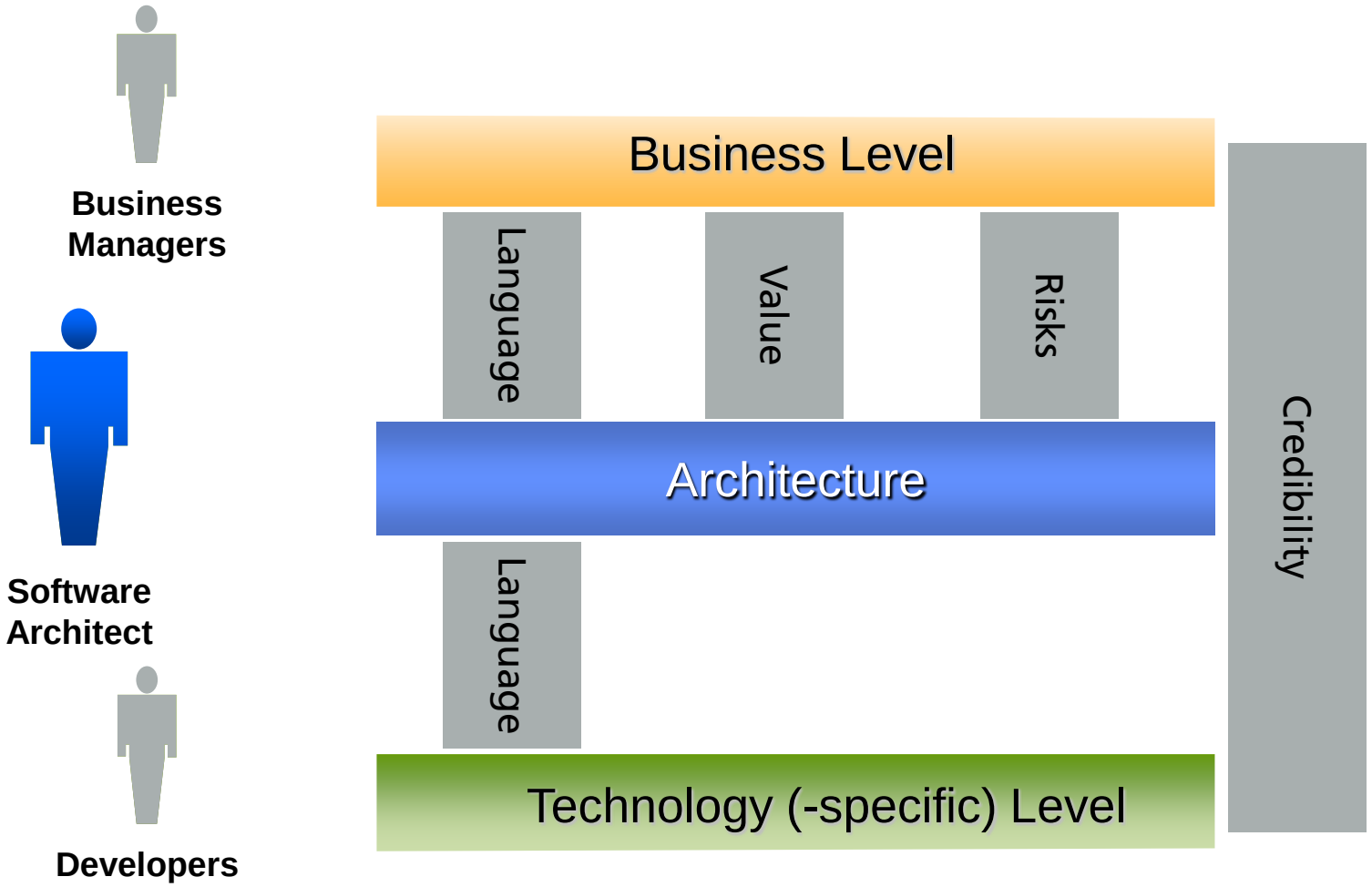
- … **provide guidance**
  - Plan for constructing a system
  - Technical leadership and coordination
  - Standards and consistency
- … **enable communication**
  - Clear technical vision and roadmap
  - Explicit documentation for communication
- … **balance technical risks**
  - Identification and mitigation
  - Definition of solution concepts
  - Anticipation (preparation) for changes
- … **manage the inherent complexity** of software
  - Products to be built
  - Increasing interconnection of systems
  - Integration with legacy systems
  - Collaboration of organizational units

Architecting: The Activity



Fraunhofer IESE

Architect as a Mediator and Communicator



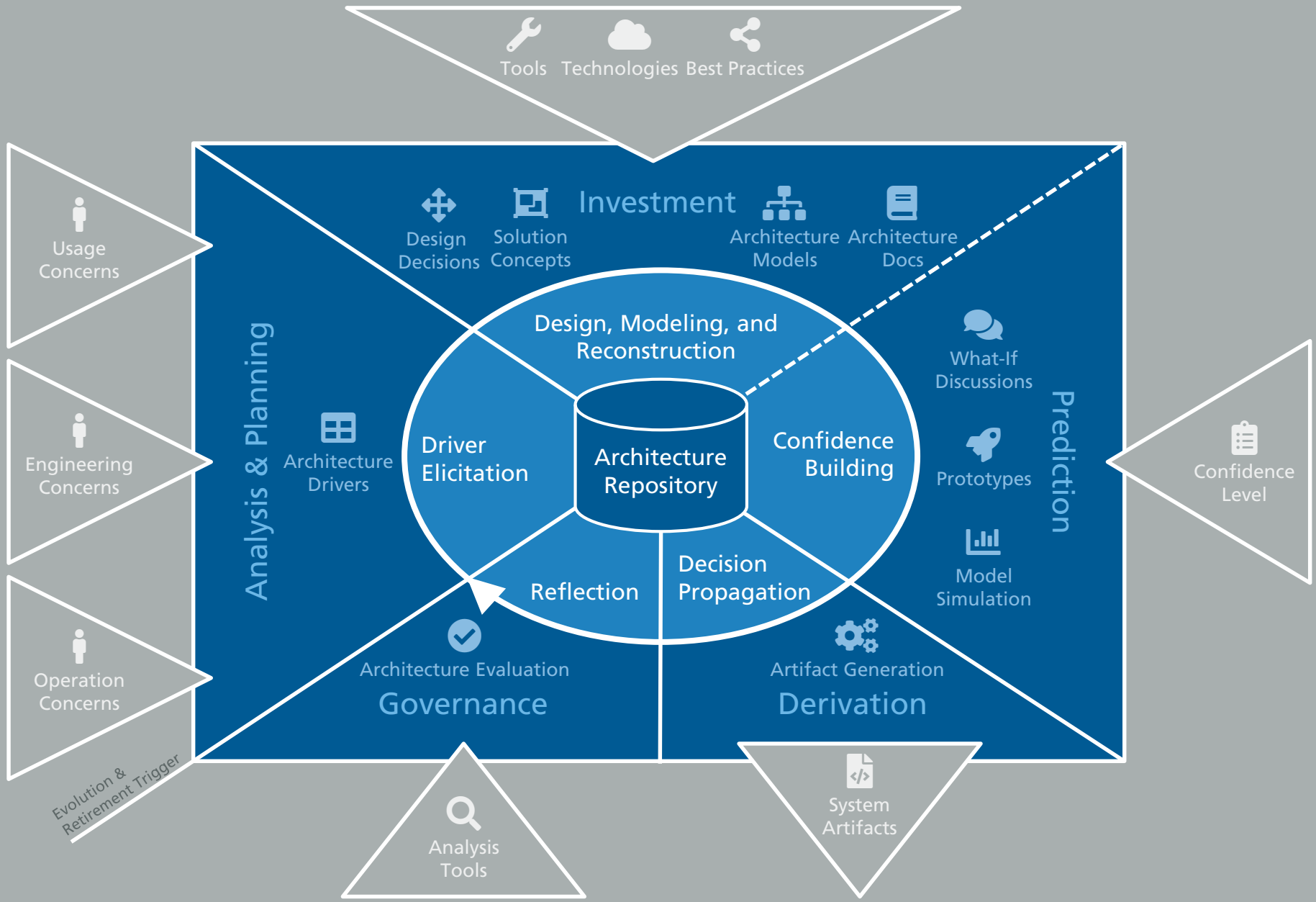
What are the challenges?

- Complexity of the system itself
- Interconnection of the systems
- Continuous change of the system
- Distributed development processes
- Technology choice
- Integration with already existing systems
- Conflicting quality requirements
  - Time to market
  - Maintainability
  - Fault tolerance
  - …

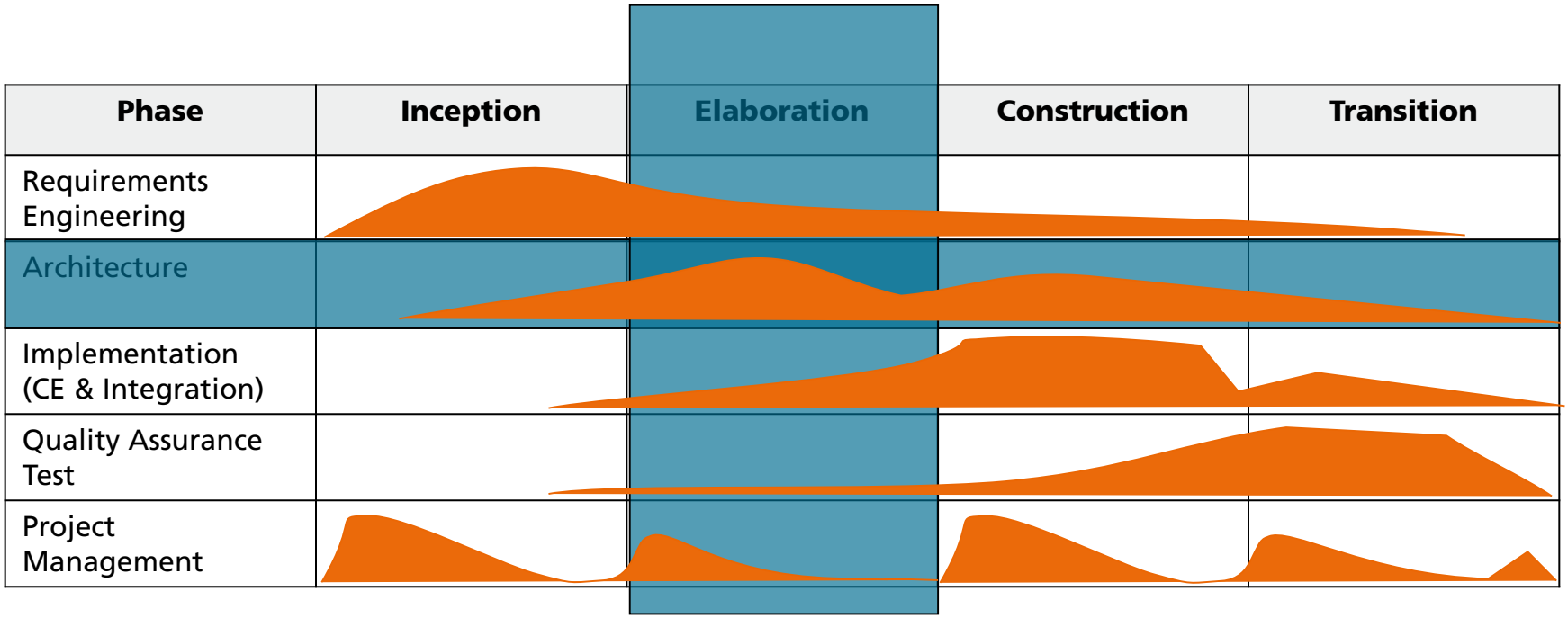
What is the solution?

- Develop your system architecture centric
- In your architecture:
  - Identify crucial challenges
  - Find appropriate solutions for these challenges

The Big Picture of Architecting



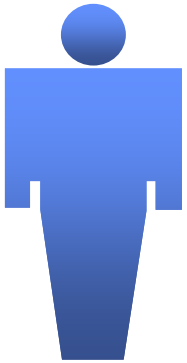
When to Spend Architecting Effort…
… in the Rational Unified Process (RUP)



Stakeholders and Project Settings

Typical Stakeholders

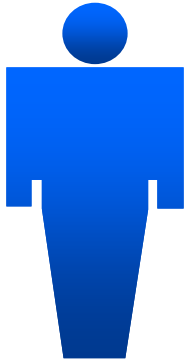
Customer management



Project manager

End user

...

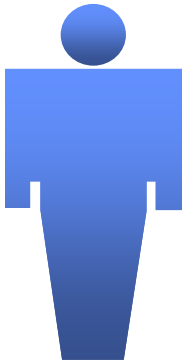


Software architect

Developer

Tester

Development manager



Maintainer

The Role of Stakeholders and their Involvement

- **Stakeholders** have **concerns**
  - Concerns form the product…
  - … and drive the architecture
- The architect has to
  - **Identify** and know the stakeholders!
  - **Involve** the stakeholders early and continuously!
  - Know their **concerns**!
    - Real needs, wishes
  - **Manage their expectations**!
    - Prioritize: not every wish can be fulfilled
    - Make tradeoffs

Task: Identify the key stakeholders of the project

Architecture Significant Requirements

Architectural Drivers

- **Business goals**
  - Customer organization
  - Developing organization
- **Quality attributes**
  - System in use
  - System under development
- **Key functional requirements**
  - Unique properties
  - Make system viable
- **Constraints**
  - Organizational and technical
  - Cost and time

- cause complexity
- might be competing
- might be interpreted differently
- need to be managed

→ are captured in **architecture scenarios**

Architecture Driver Template

Categorization	
Driver Name	*Concise short name*
Driver ID	*Unique identifier*
Status	*[Open, Elicited, Under Design, Designed, Under Realization, Realized, Done]*
Priority	*[High - Medium – Low]*

Responsibilities	
Supporter	*Stakeholders supporting the driver*
Sponsor	*Stakeholders paying for the driver*
Author	*Responsible for filling this template*
Inspector	*Stakeholders reviewing this driver*

Description		Quantification
Environment	*Context and/or initial situation applying to this driver*	▪ *Measurable effects applying to the environment*
Stimulus	*The event, trigger or condition arising from this driver*	▪ *Measurable effects applying to the stimulus*
Response	*The expected reaction of the system to the driver event (black box view putting no constraints on the design)*	▪ *Measurable effects applying to the response* ▪ *Measurable indicators that the driver has been achieved by the architecture*

Architecture Driver Example

Categorization	
Driver Name	Application startup time
Driver ID	AD.01.PERFORMANCE
Status	Realized
Priority	High

Responsibilities	
Supporter	Carla Customer
Sponsor	Mike Manager
Author	Arnold Architect
Inspector	Alfred Architect

Description		Quantification
Environment	The application is installed on the system and has been started before at least once. The application is currently closed and the system is running on normal load.	▪ Previous starts >= 1
Stimulus	A user starts the application from the Windows start menu.	
Response	The application starts and is ready for inputting search data in less than 1 second. The application is ready for fast answers to search queries after 5 seconds.	▪ Initial startup time < 1s ▪ Full startup time < 5s

Architecture Driver Example
Most Important

Categorization	
Driver Name	Application startup time
Driver ID	AD.01.PERFORMANCE
Status	Realized
Priority	High

Responsibilities	
Supporter	
Sponsor	
Author	
Inspector	

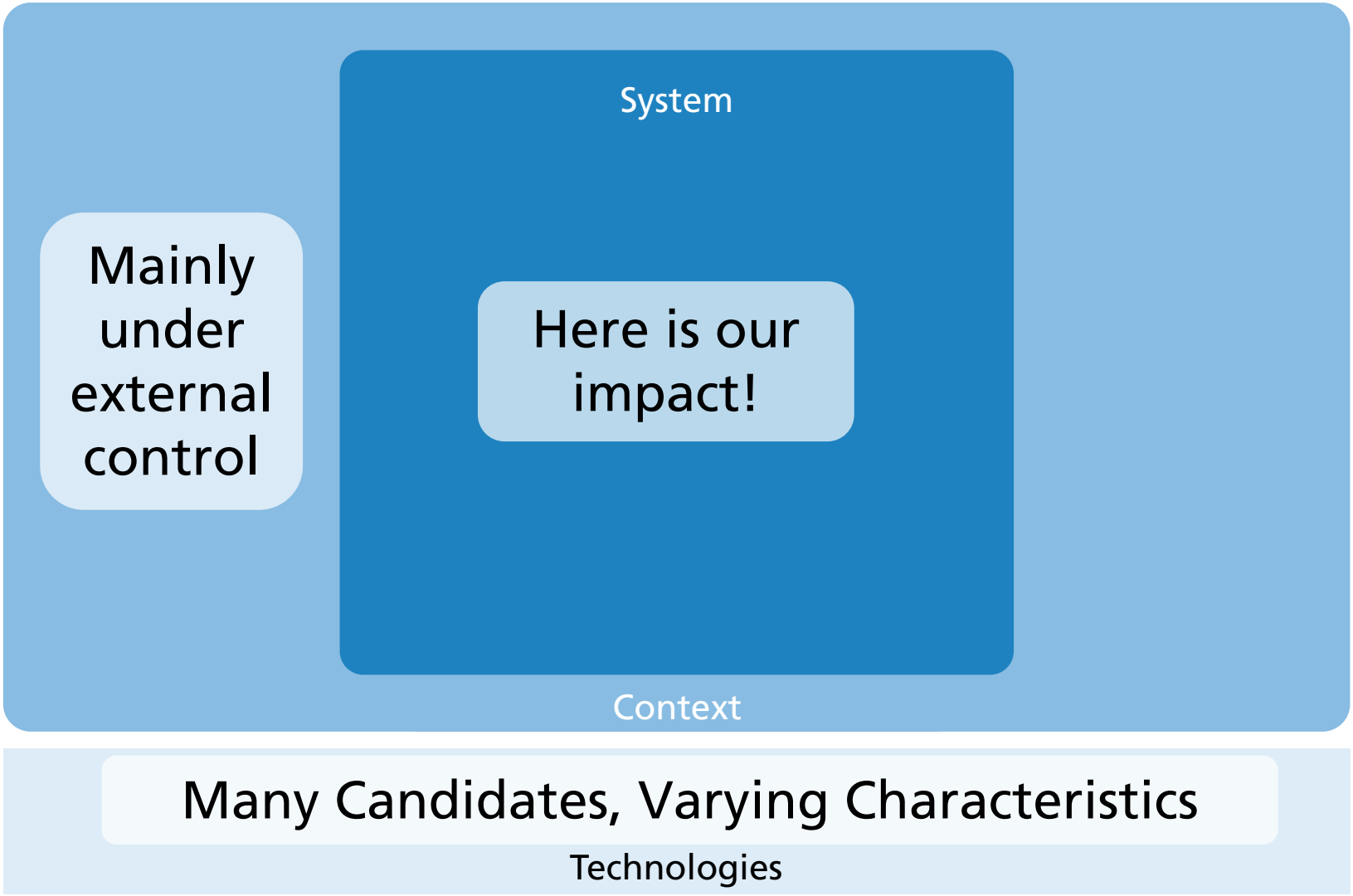
Description		Quantification
Environment	The application is installed on the system and has been started before at least once. The application is currently closed and the system is running on normal load.	▪ Previous starts >= 1
Stimulus	A user starts the application from the Windows start menu.	
Response	The application starts and is ready for inputting search data in less than 1 second. The application is ready for fast answers to search queries after 5 seconds.	▪ Initial startup time < 1s ▪ Full startup time < 5s

Notations for Architecture Drivers

Business Goals	Constraints	Quality Attributes	Key Functional Requirements
• **Natural Language** • **Links to Other Documents**	• **Natural Language** • Links to Other Documents	• **Drivers** • **Scenarios** • Links to Other Documents • (Use Cases)	• **Use Cases** • **User Stories / Epics** • Driver • Scenario • Natural Language • Links to Other Documents

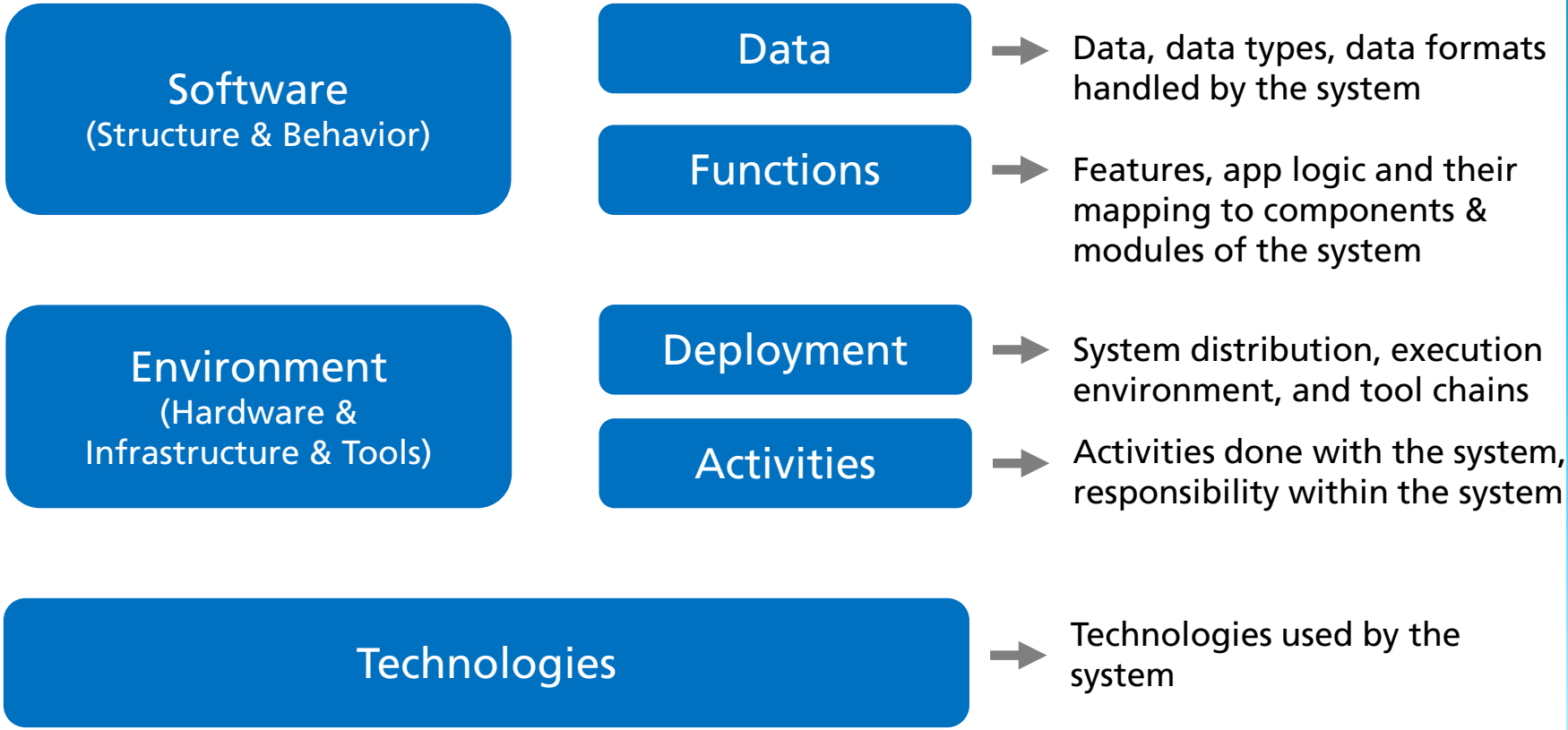
Design

Architectural Scope

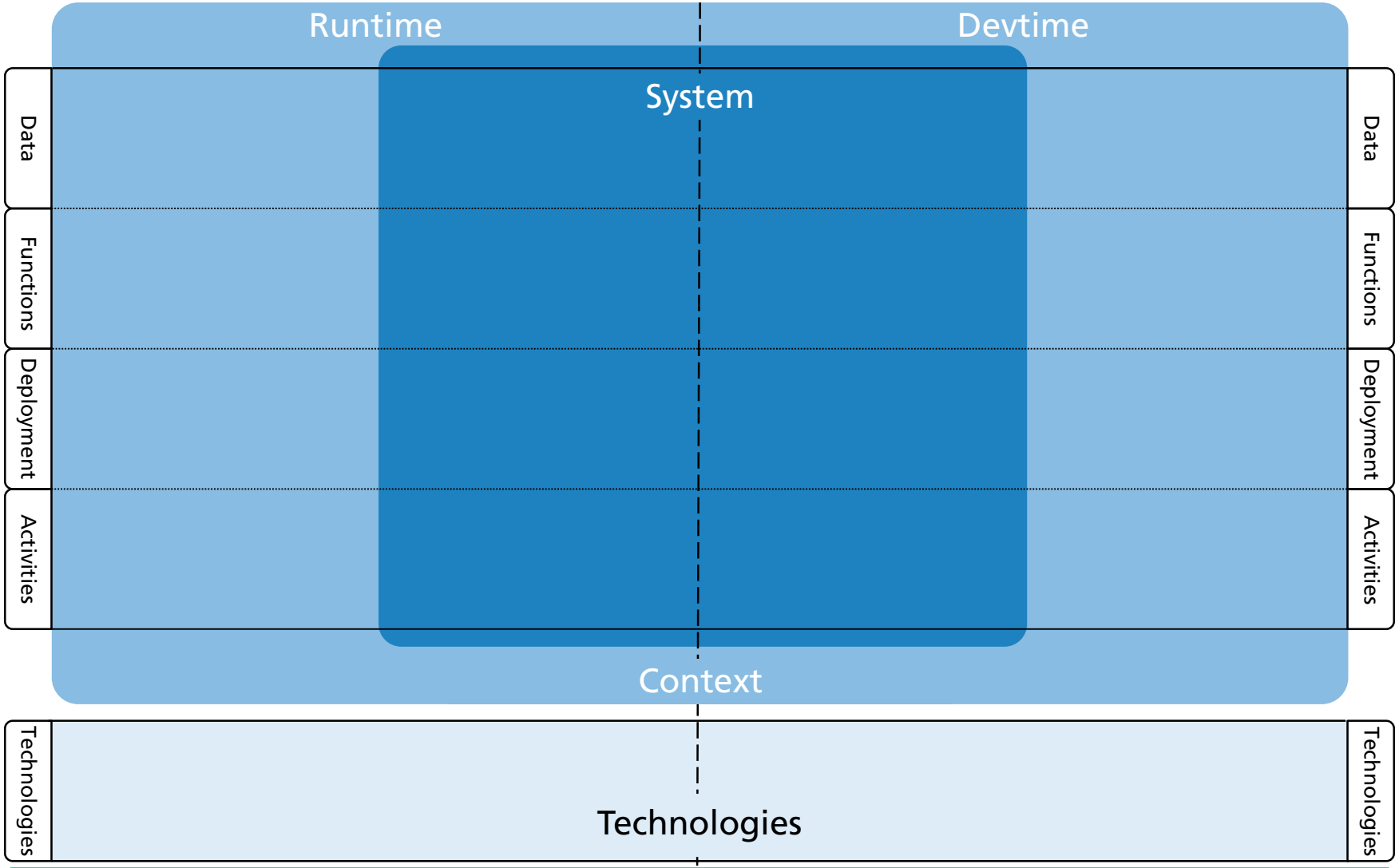


Dimensions

- Decompose by addressing **different dimensions**

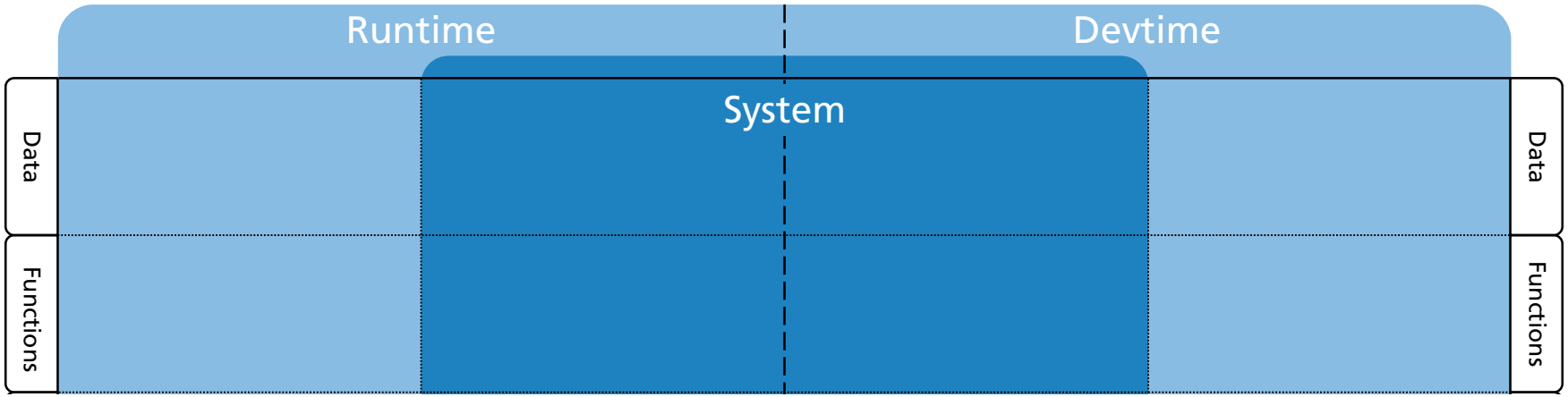


Data | Functions | Deployment | Activities | Technologies



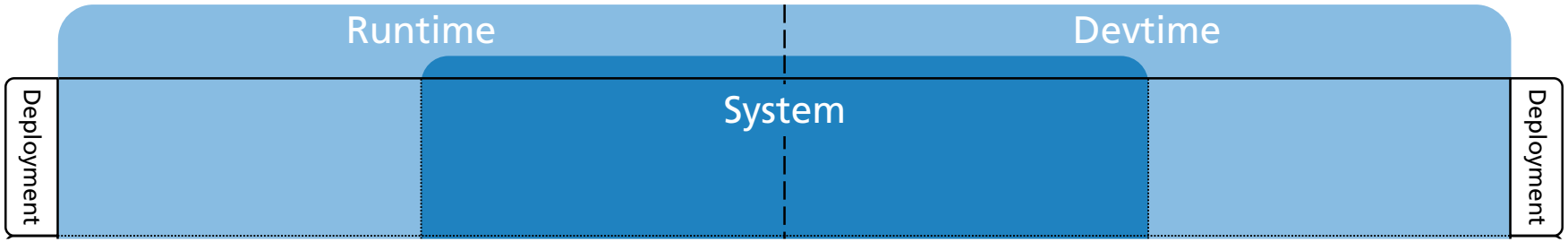
Fraunhofer IESE

Topics to think about – Data & Functions



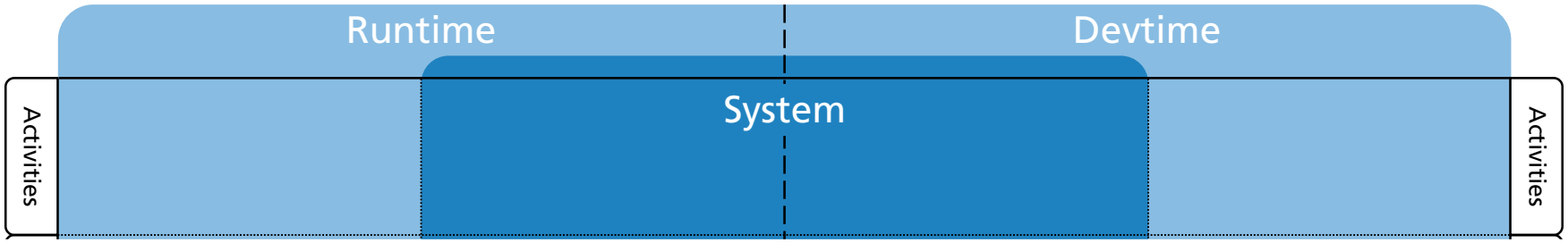
Runtime	Devtime
• What does the system do and what do the systems in the context do? • What are functions and data at the interface to external systems? • How do I decompose the functionality in executable components? • How should the components communicate and what interfaces should they use/provide? • What data is exchanged how many times? • Where is the data created, transported, processed and stored within the system?	• How are data structures defined? • What data formats are needed? • How is the software partitioned in the development environment? • What are units at development time that should be compiled and tested separately? • How can modules be divided so redundant code can be prevented? • How can modules be decoupled?

Topics to think about – Deployment



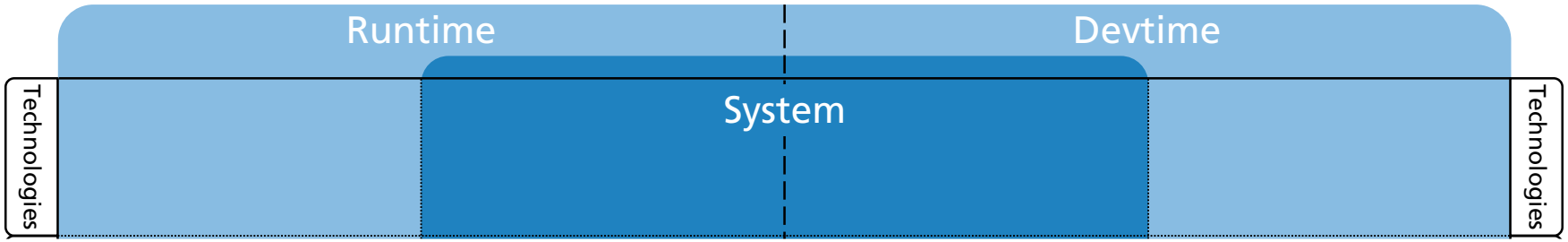
Runtime	Devtime
• What processes do exist during runtime? • Where are processes/tasks allocated and executed? • How do processes communicate with one another? • What execution environments do I need? (e.g. Application Server) • How to partition the SW into parallel tasks? • Who operates the software in what location?	• How do runtime components (allocated to processes/tasks) map to development units? • How are the compilation units mapped to deployable artifacts? • How are deployment units created? • How does the tool chain look like for building and deploying the software to the final execution environment? • How are modules mapped to teams?

Topics to think about – Activities



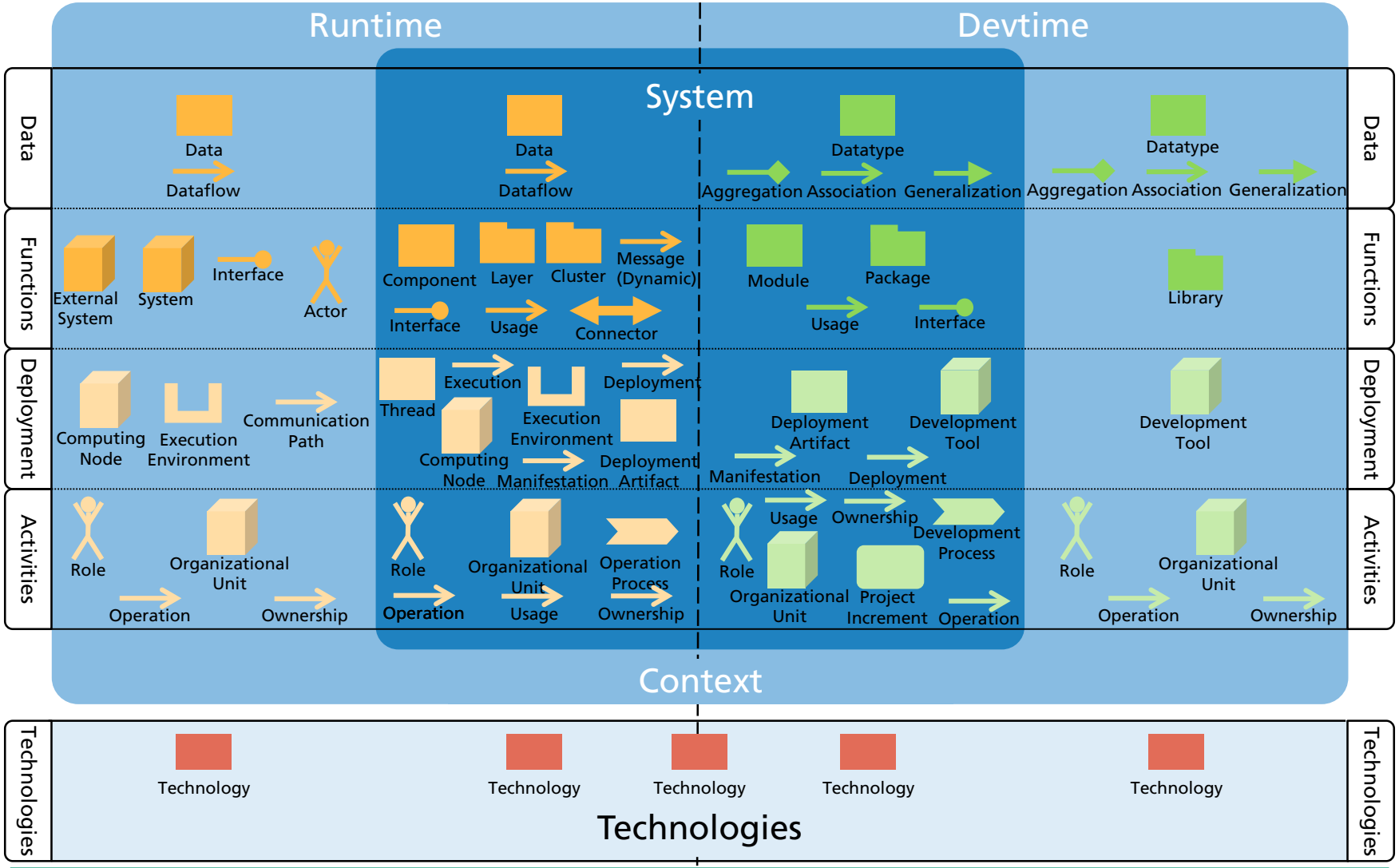
Runtime	Devtime
• How does operation of the system look like?	• Who is involved in the development and delivery of the software system? • Who is responsible for what system portions? • How is the development, quality assurance and delivery of the software organized in terms of processes and organizational units? • How to assign the modules to development iterations?

Topics to think about – Technologies



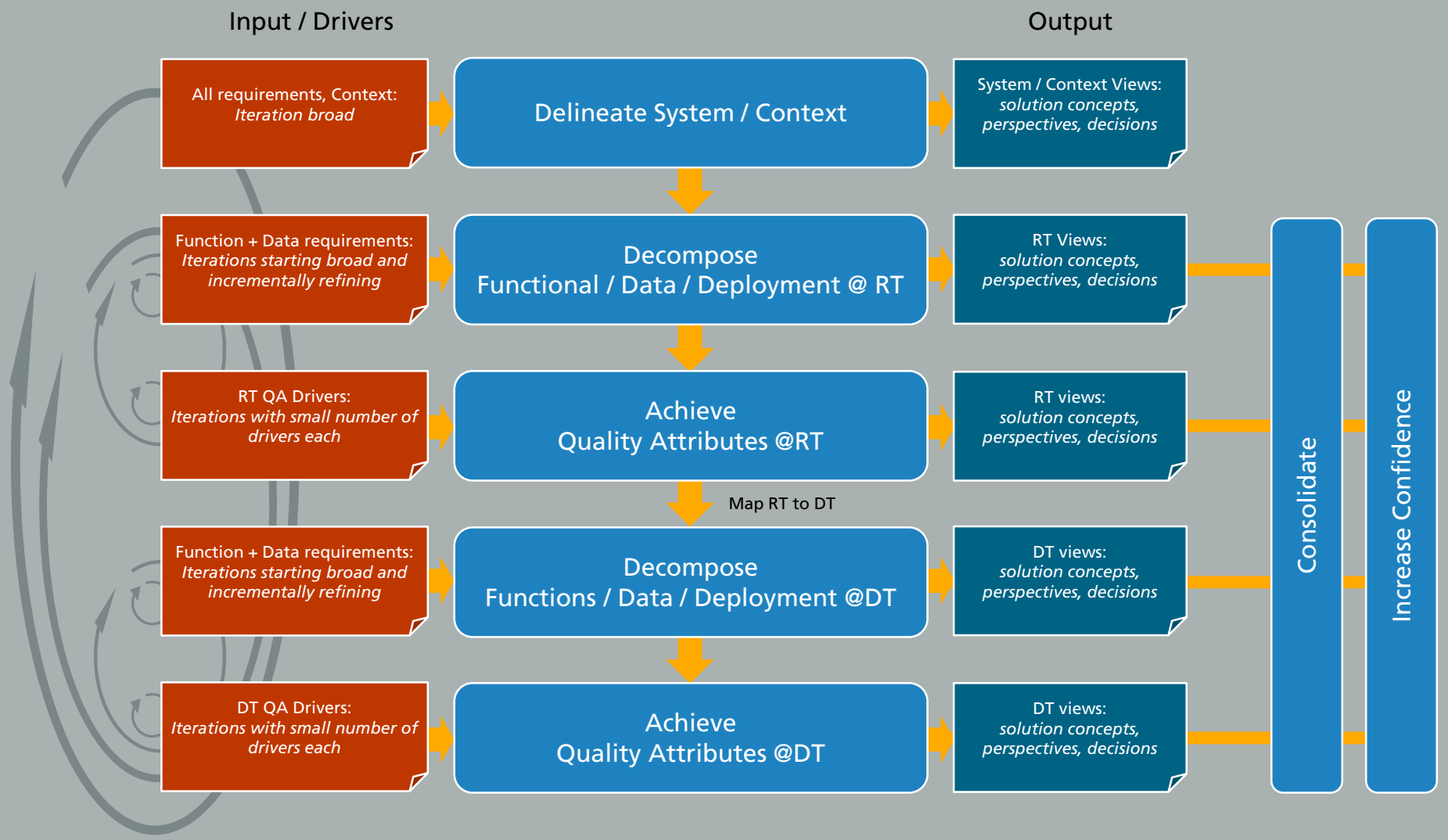
Runtime	Devtime
• What technologies are used at runtime? • What technologies are used for communication between the system and external systems? • What are properties of candidate technologies (bandwidth, latency, throughput, processing power, energy consumption, etc.)	• What tools are being used for development and delivery? • How are the tools connected/integrated?

Notation: Architecture Decomposition Framework



Fraunhofer IESE

Architecture Design Process



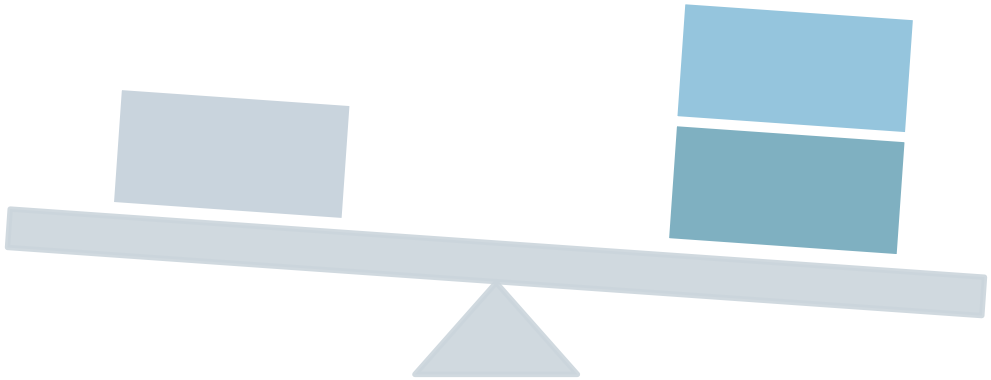
Architecture: Essential Principles

- **Abstraction**
  - Extraction of the “essentials”
- **Modularization & Localization of concerns**
  - Hierarchical decomposition (Divide & conquer)
  - Create modular units with clearly defined interfaces and dependencies
  - Create modular solutions that can be changed in one place
- **Separation of Concerns**
  - Reduce aspects to the relevant information (e.g. view-based)
- **Encapsulation & Information hiding**
  - Avoid global variables
  - Restrict access/visibility to internals
- **Uniformity**
  - Interfaces should be implemented consistently with the same mechanisms (e.g. interface classes, …)
  - Usage of communication protocols

Architecture Design Decisions

- **Design Decisions Balance** competing concerns
- **Some Design Decisions** are made **early** in the lifecycle
  - Typically have **far-reaching** effects
  - Are **hard to change** (in later phases or future projects)

→ The impact of architecture design decisions has to be known!



Decision Rationale Template

Decision Name	*Concise short name*
Design Decision ID	*Unique identifier*
Explanation	*Explanation of the decision rationale*

Pros & Opportunities	**Cons & Risks**
▪ *Points in favor* ▪ *Anticipations of future*	▪ *Points against* ▪ *Unknown or open aspects*

Assumptions & Quantifications	**Trade-Offs**
▪ *Assumption made about the driver solution (or parts of it)* ▪ *Measurable effects applying to the driver solution (or parts of it)*	▪ *Trade-offs to other design decisions, quality attributes, solutions concepts, architecture drivers* ▪ *Potentially impacted if this solution changes*

Manifestation Links	*Links to models, diagrams, additional documentation*

Decision Rationale Example

Decision Name	Decoupled loading of search data
Design Decision ID	DD.01
Explanation	Loading the search data is done in a separate thread. The application's UI can be started and used for typing in search queries before the search data is actually loaded.

Pros & Opportunities	**Cons & Risks**
▪ Data loading time does not add on startup time	▪ Loading in separate thread requires synchronization and makes implementation more difficult

Assumptions & Quantifications	**Trade-Offs**
▪ Data can be loaded in 5s	▪ Maintainability, understandability

Manifestation Links	

Driver Solution Template

Driver Name	*Concise short name*
Driver ID	*Unique identifier*

Steps	1. *Logical flow to explain driver solution (white box view explaining the design)* 2. *The glue between design decisions (accepted and discarded)* 3. *Putting all related design decisions in a combined and larger context*	
Related Design Decisions	*ACCEPTED* ▪ *Link to design decision (detailed description) to enable traceability*	*DISCARDED* ▪ *Link to design decision (detailed description) to enable traceability*

Pros & Opportunities

- *Points in favor*
- *Anticipations of future*

Cons & Risks

- *Points against*
- *Unknown or open aspects*

Assumptions & Quantifications

- *Assumption made about the driver solution (or parts of it)*
- *Measurable effects applying to the driver solution (or parts of it)*

Trade-Offs

- *Trade-offs to other design decisions, quality attributes, solutions concepts, architecture drivers*
- *Potentially impacted if this solution changes*

Manifestation Links	*Links to models, diagrams, additional documentation*

Driver Solution Example

Driver Name	Application startup time
Driver ID	AD.01.PERFORMANCE.

Steps	1. Application always stores preprocessed index-structures on updates of searchable items 2. On startup, loading of search data is moved to a separate thread 3. The UI is started and ready for user input while loading of search data is ongoing 4. After loading the search data, searches can be done without the user noticing that search was not available before
Related Design Decisions	▪ DD.01 Decoupled loading of search data ▪ DD.12 Preprocessed index-structures of search data

Pros & Opportunities

- Very fast startup time, application directly usable by user

Cons & Risks

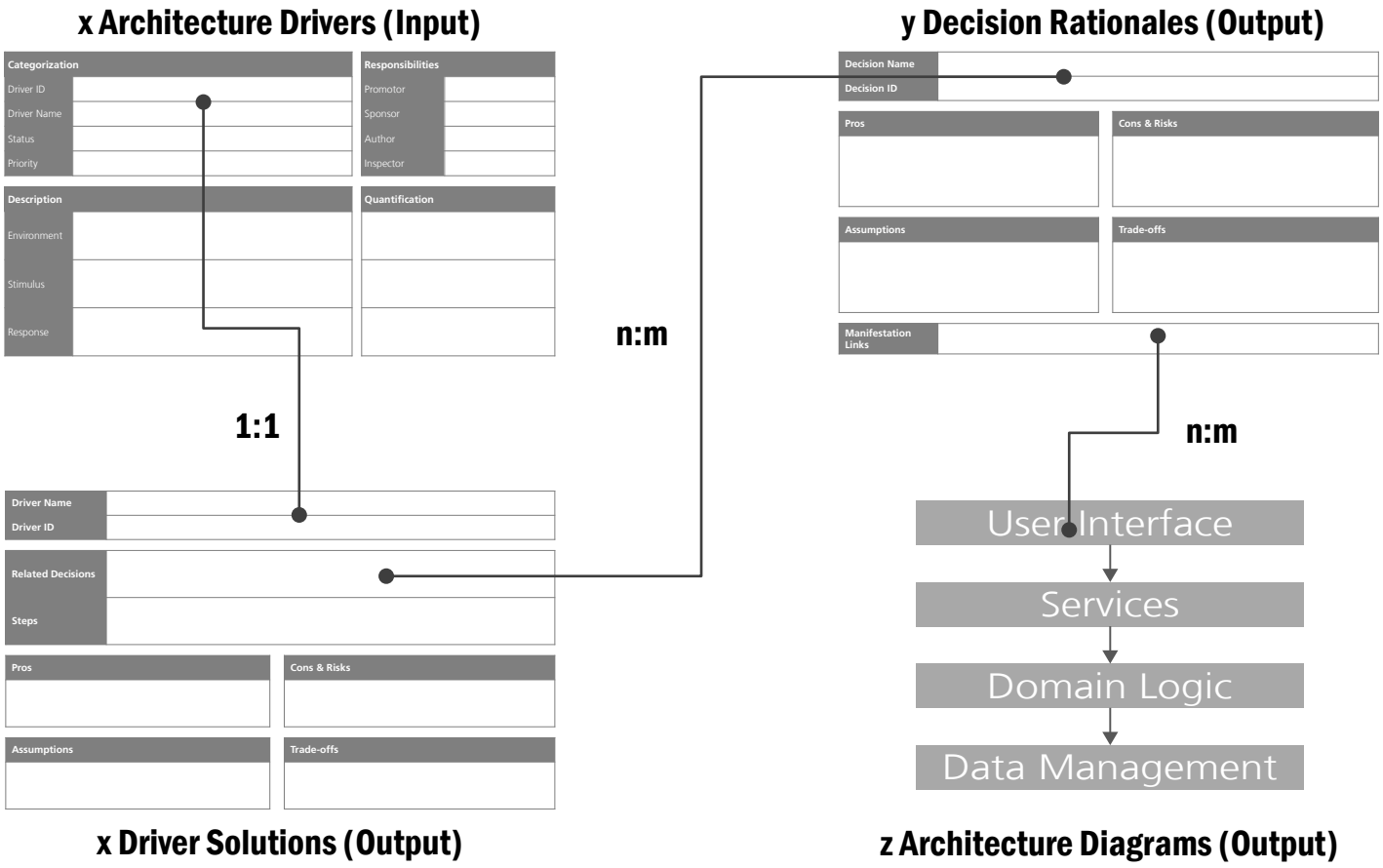
- More effort in realization
- Loading in separate thread requires synchronization and makes implementation more difficult

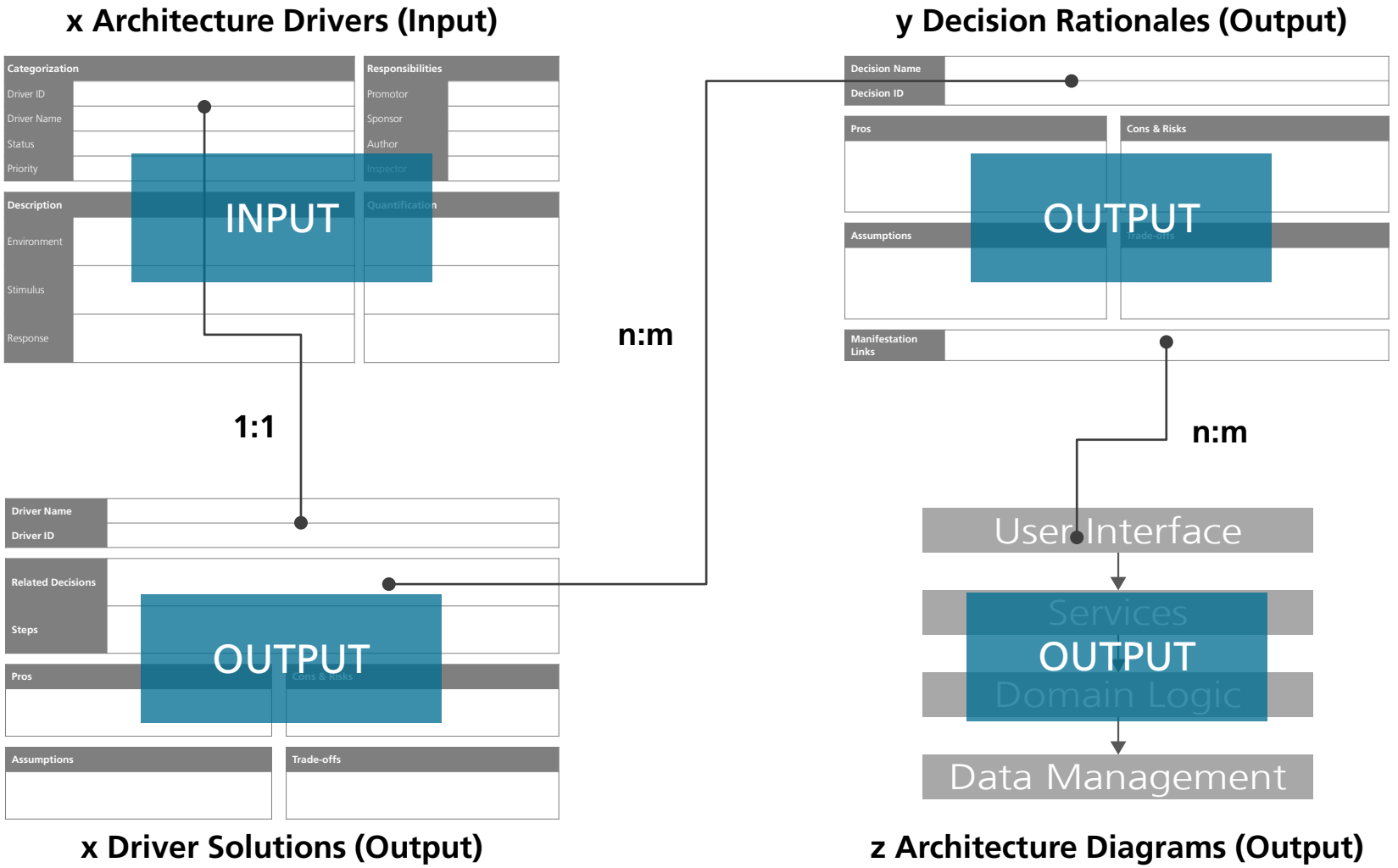
Assumptions & Quantifications

- Data can be loaded in 5s
- User rarely sends a search in less than 4s after start is completed

Trade-Offs

- Maintainability, understandability





Minimum Set of Views

- Context Delineation: System Context Diagram
- Deployment View: Deployment Diagram
- Function View:
  - Structural Part: Component Diagram
  - Behavioral Part: Sequence Diagram
  - Package Diagram/ Class Diagram
- Data View:
  - Data Flow Diagram
  - Class Diagram/ Database Model
- Activity View: Allocating Responsiblity

- Ensure Quality Drivers
  - Use patterns
  - Consolidate and align all the models

Documentation

Who uses the Architecture Documentation?

Developer

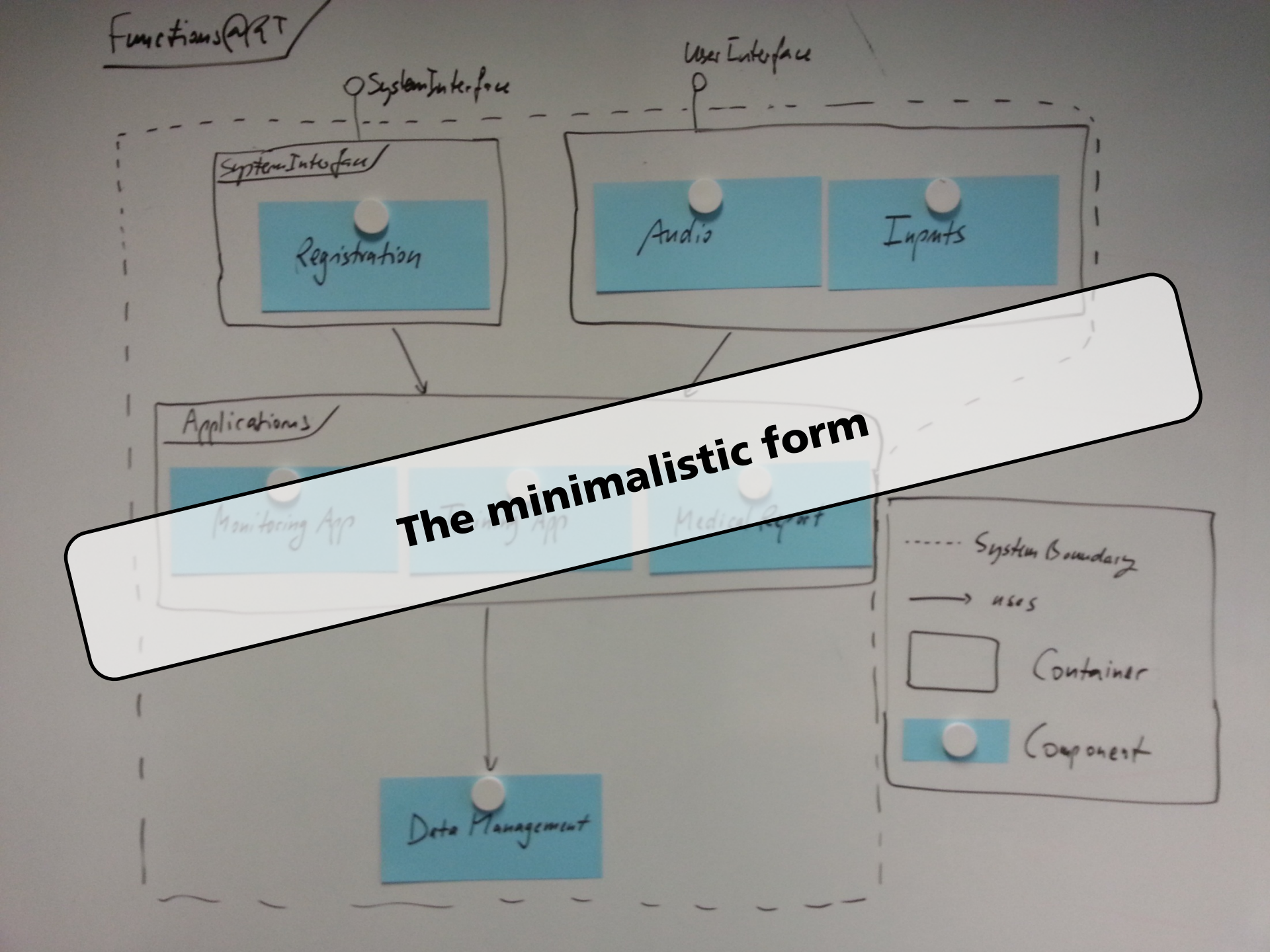
Architect

Projectleader

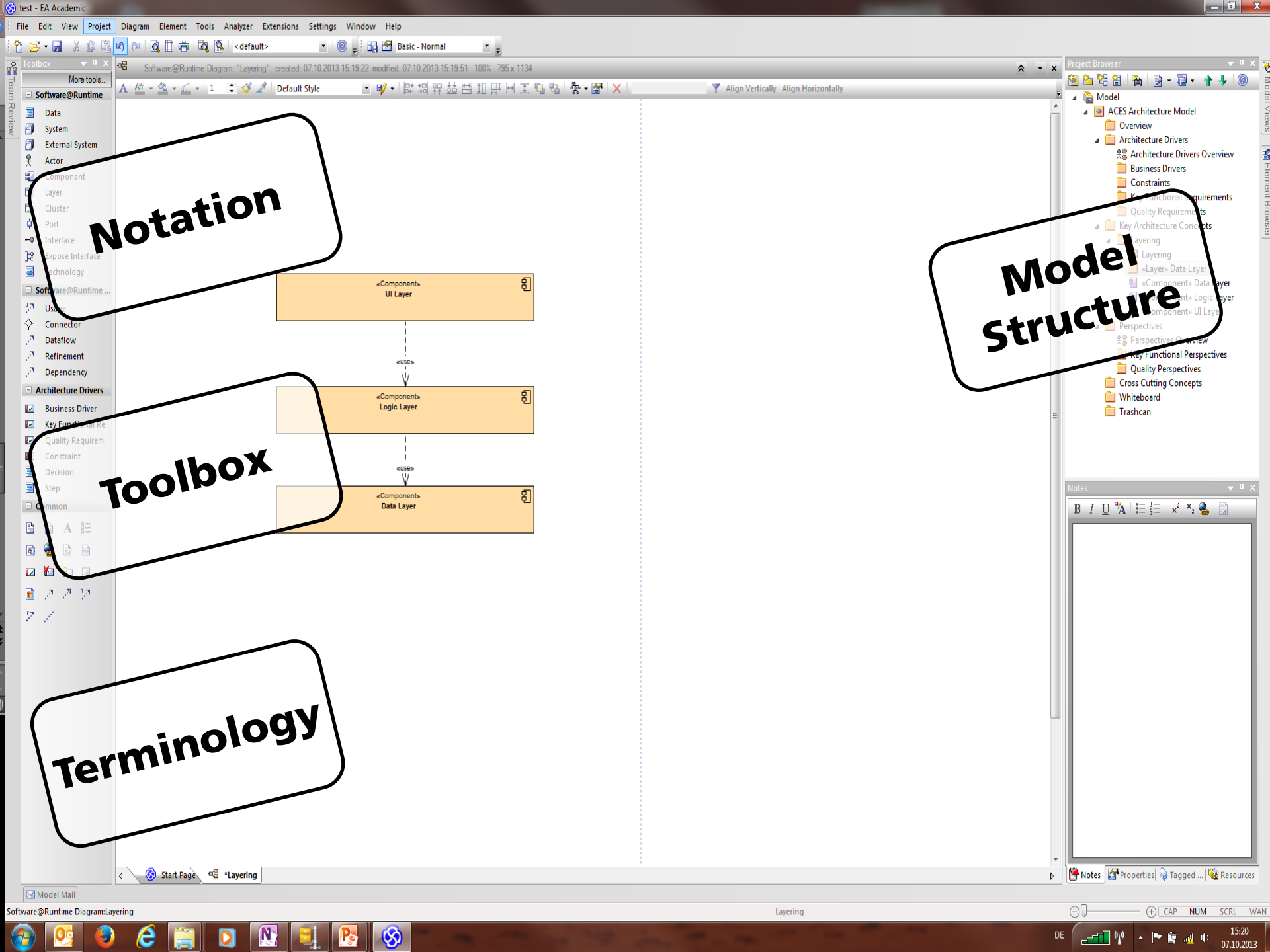
Manager

Quality Engineer

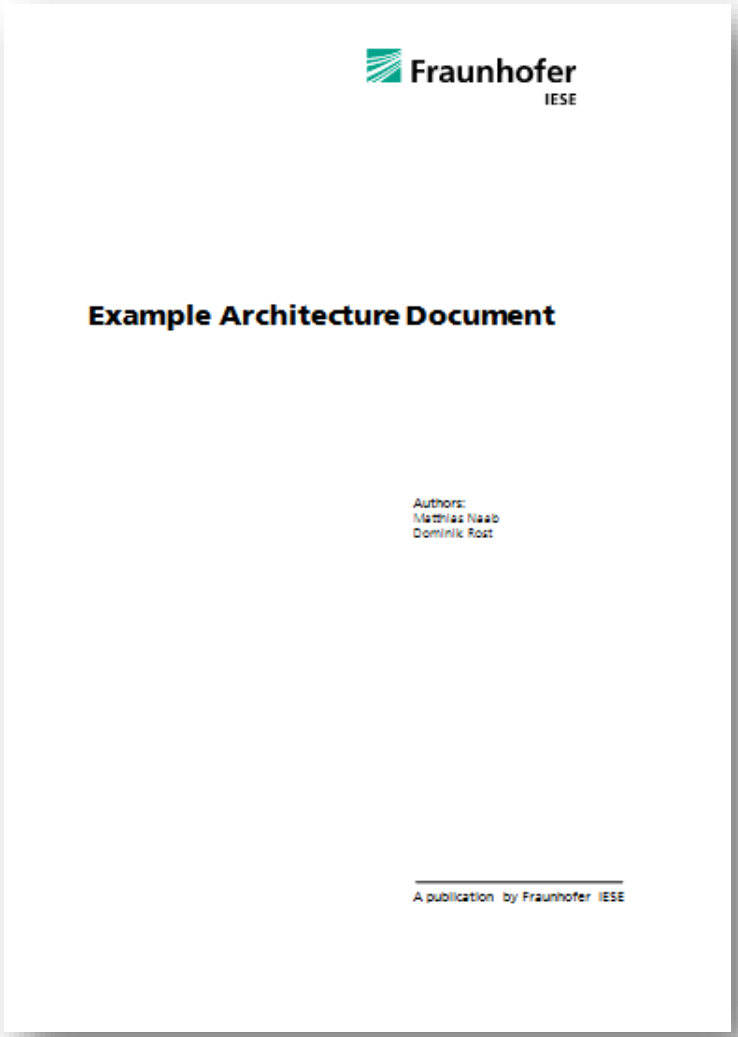
Marketing



The minimalistic form


Notation
Toolbox
Terminology
Model Structure
«Component»
UI Layer
«use»
«Component»
Logic Layer
«use»
«Component»
Data Layer

Software Architecture Document – Example



Fraunhofer IESE

Example Architecture Document

Authors:
Matthias Naab
Dominik Rost

A publication by Fraunhofer IESE

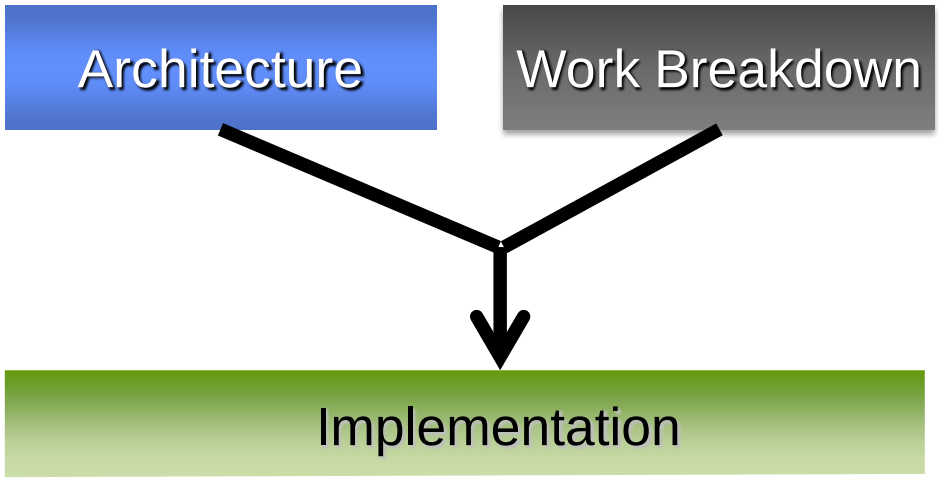
Fraunhofer IESE

Tool Support for Architecture Modeling

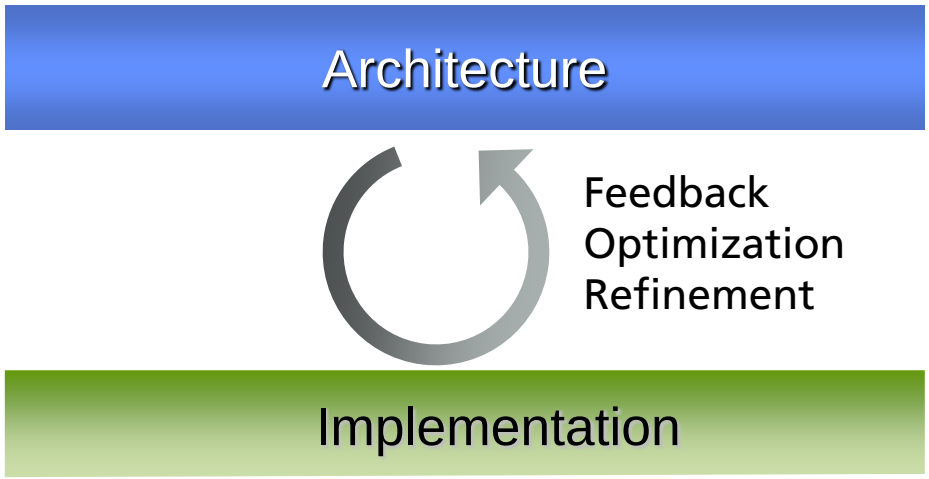
- First use paper
- Then use some Modeling tool (Enterprise Architect for example)
- Use word for documenting things

Prediction and Control

You can start your implementation



In the real world the story only starts…



Evolution
Migration
Reconstruction

…

So, be brave!

Organizational Aspects

Expected Outcome

- Architecture Model
- Architecture Document
  - Architecture Drivers
  - Design Decisions
  - Scenario Solutions
- Technology Assessment
- Lessons Learned

Organizational Aspects

- Architects from every team
  - Discuss inside the team and other stakeholders to design the system
  - Discuss with other teams' architects to align the architectures
  - Manifest the architecture
- Other members of the team
  - Help architects to take decisions
  - Provide feedback on technological matters

Contact for Your Architecture Task:

Rodrigo Falcão
Phone: +49 (0)631 / 6800-2159
Mail: rodrigo.falcao@iese.fraunhofer.de

ACES

architecture@iese.fraunhofer.de
architecture.iese.fraunhofer.de