



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, relationships**, 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



Architectures: The Artifact

■ ... **provide guidance**

- Plan for constructing a system
- Technical leadership and coordination
- Standards and consistency

■ ... **balance technical risks**

- Identification and mitigation
- Definition of solution concepts
- Anticipation (preparation) for changes

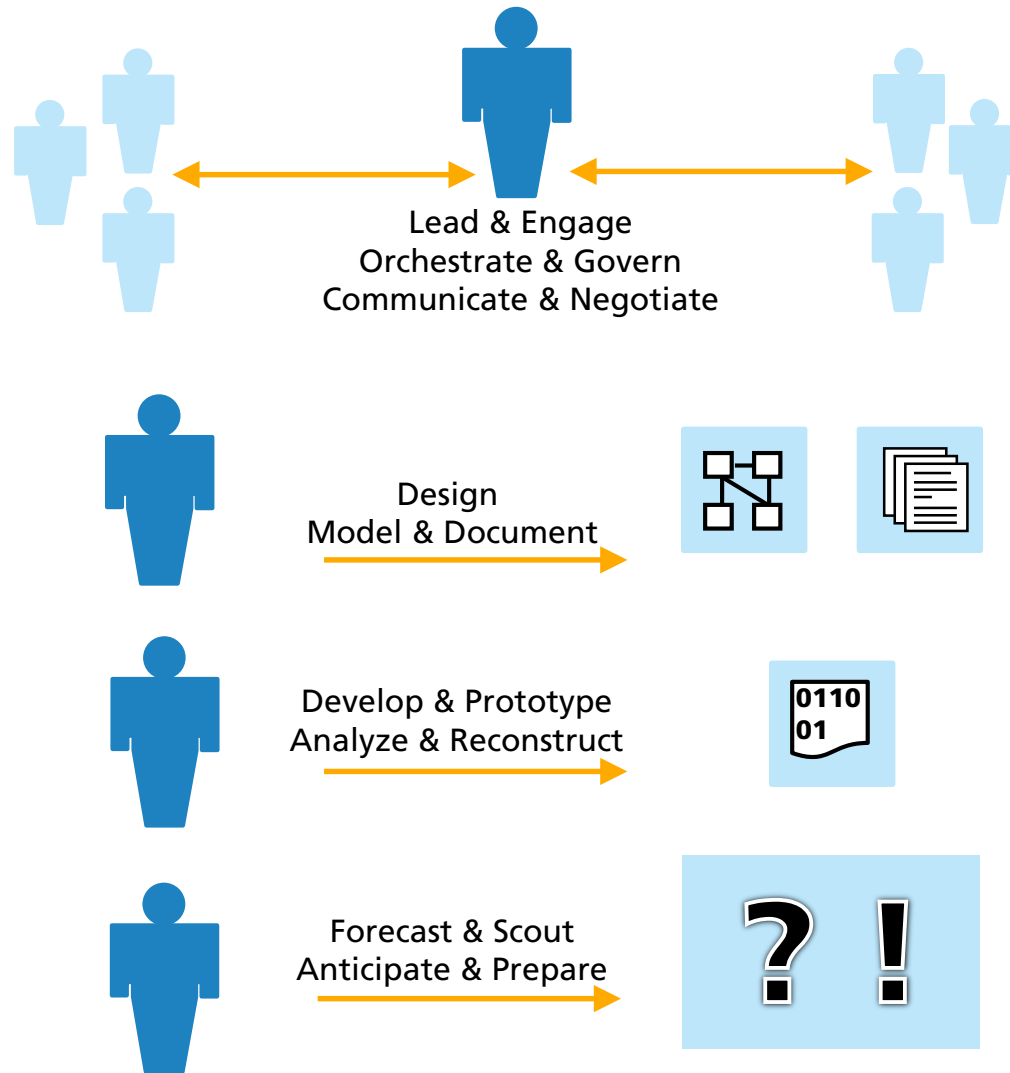
■ ... **enable communication**

- Clear technical vision and roadmap
- Explicit documentation for communication

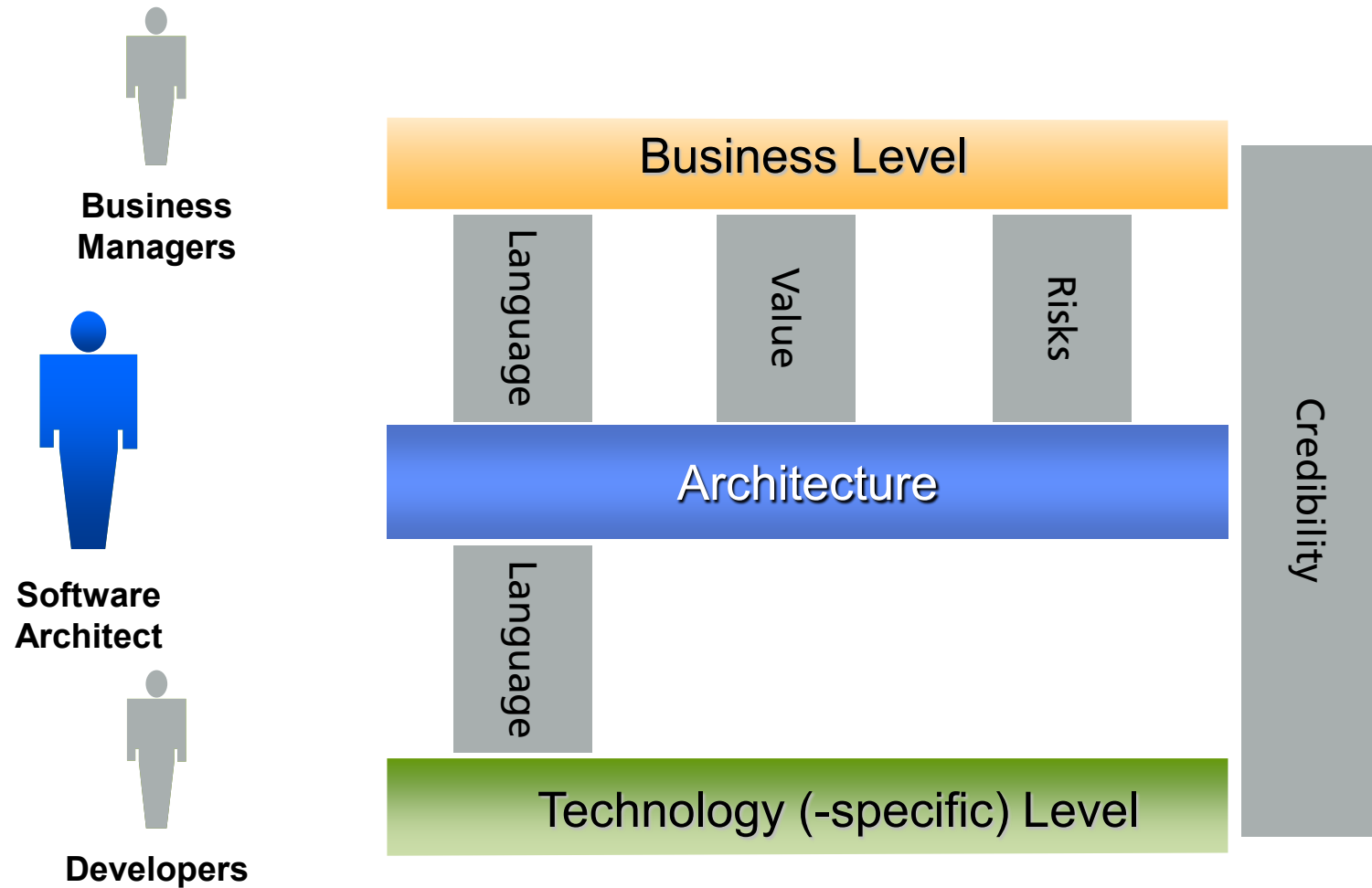
■ ... **manage the inherent complexity** of software

- Products to be built
- Increasing interconnection of systems
- Integration with legacy systems
- Collaboration of organizational units

Architecting: The Activity



Architect as a Mediator and Communicator



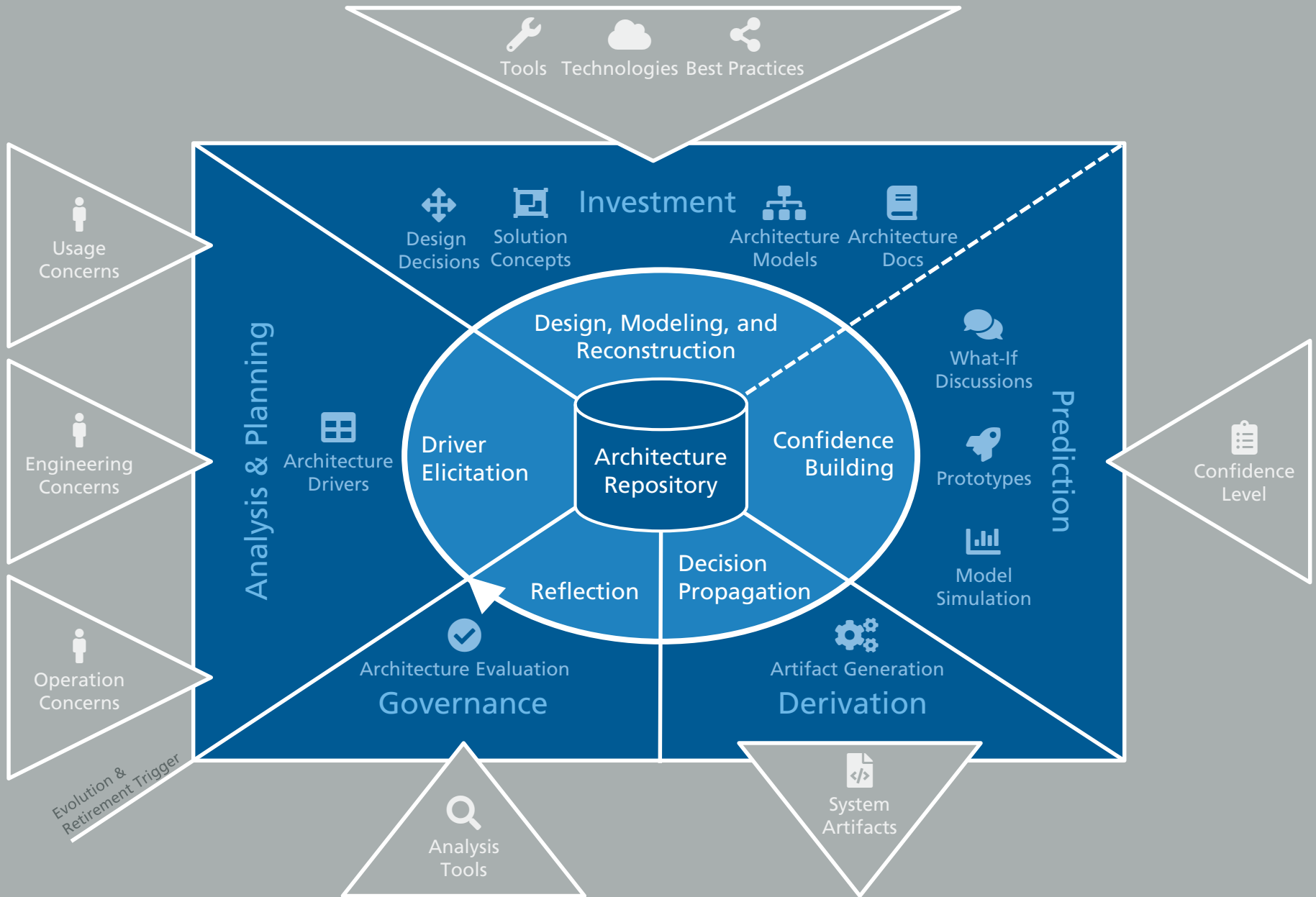
What are the challenges?

- 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
 - ...

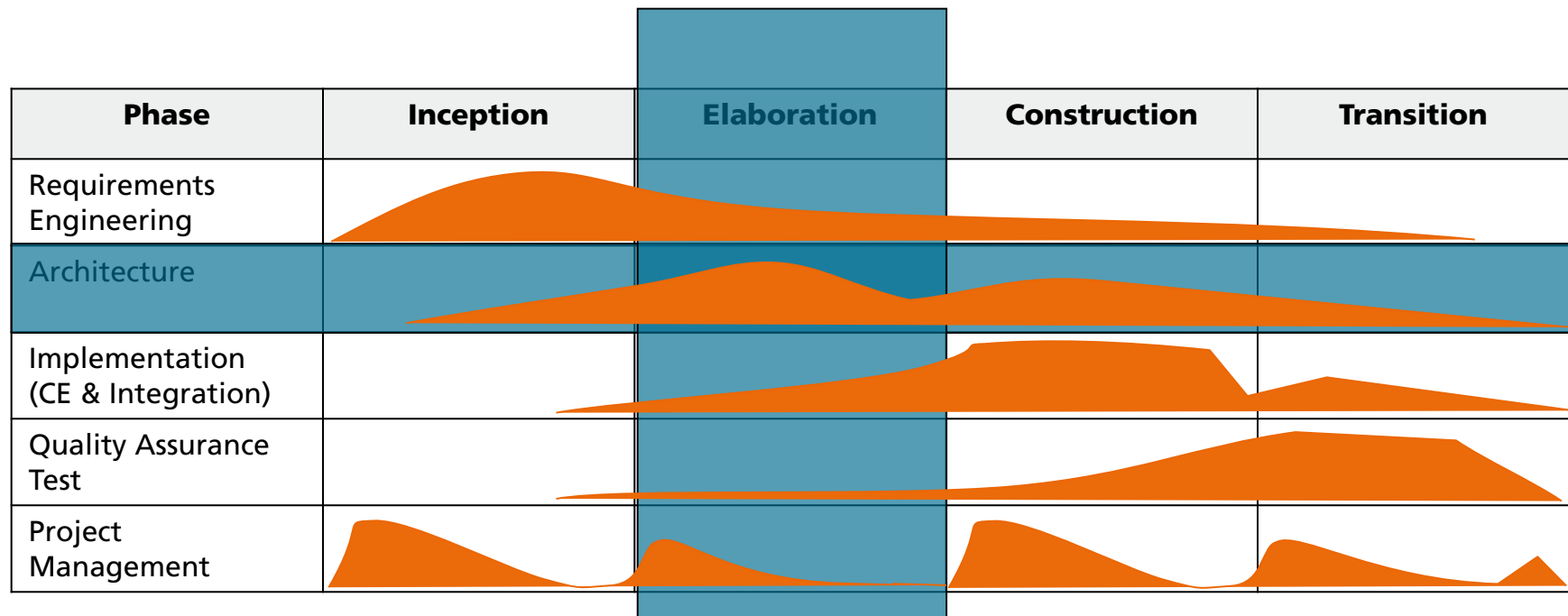
What is the solution?

- 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**



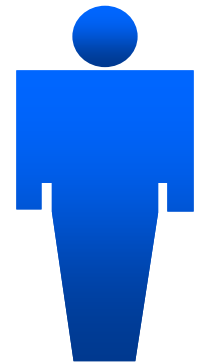
**Project
manager**



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

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		Responsibilities	
Driver Name	<i>Concise short name</i>	Supporter	<i>Stakeholders supporting the driver</i>
Driver ID	<i>Unique identifier</i>	Sponsor	<i>Stakeholders paying for the driver</i>
Status	<i>[Open, Elicited, Under Design, Designed, Under Realization, Realized, Done]</i>	Author	<i>Responsible for filling this template</i>
Priority	<i>[High - Medium - Low]</i>	Inspector	<i>Stakeholders reviewing this driver</i>

Description		Quantification
Environment	<i>Context and/or initial situation applying to this driver</i>	<ul style="list-style-type: none"> ▪ <i>Measurable effects applying to the environment</i>
Stimulus	<i>The event, trigger or condition arising from this driver</i>	<ul style="list-style-type: none"> ▪ <i>Measurable effects applying to the stimulus</i>
Response	<i>The expected reaction of the system to the driver event (black box view putting no constraints on the design)</i>	<ul style="list-style-type: none"> ▪ <i>Measurable effects applying to the response</i> ▪ <i>Measurable indicators that the driver has been achieved by the architecture</i>

Architecture Driver Example

Categorization		Responsibilities	
Driver Name	Application startup time	Supporter	Carla Customer
Driver ID	AD.01.PERFORMANCE	Sponsor	Mike Manager
Status	Realized	Author	Arnold Architect
Priority	High	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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> Initial startup time $< 1s$ Full startup time $< 5s$

Architecture Driver Example

Most Important

Categorization		Responsibilities	
Driver Name	Application startup time	Supporter	
Driver ID	AD.01.PERFORMANCE	Sponsor	
Status	Realized	Author	
Priority	High	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.	<ul style="list-style-type: none"> 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.	<ul style="list-style-type: none"> Initial startup time $< 1s$ Full startup time $< 5s$ 	

Notations for Architecture Drivers

Business Goals

- **Natural Language**
- **Links to Other Documents**

Constraints

- **Natural Language**
- Links to Other Documents

Quality Attributes

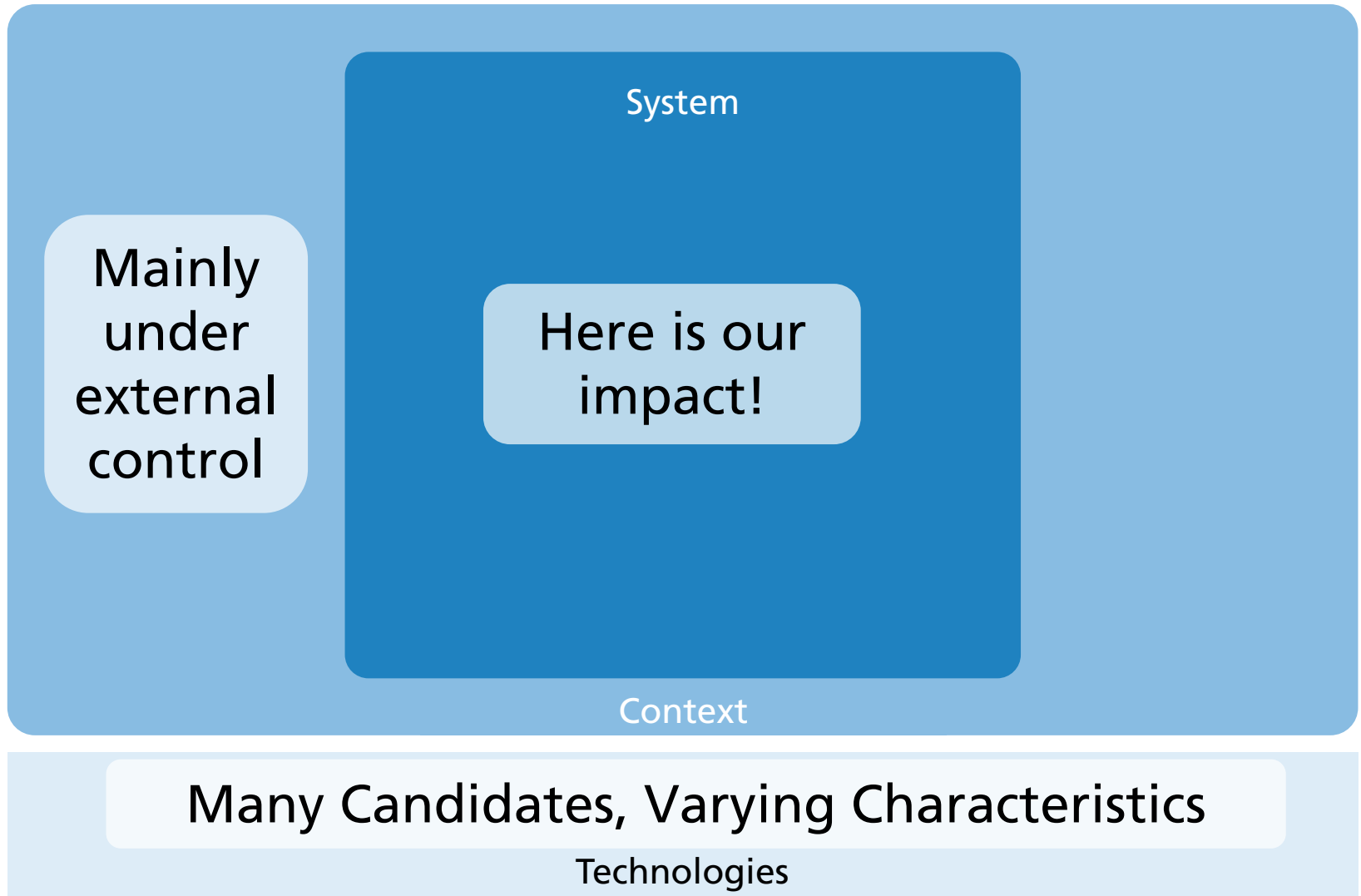
- **Drivers**
- **Scenarios**
- Links to Other Documents
- (Use Cases)

Key Functional Requirements

- **Use Cases**
- **User Stories / Epics**
- Driver
- Scenario
- Natural Language
- Links to Other Documents

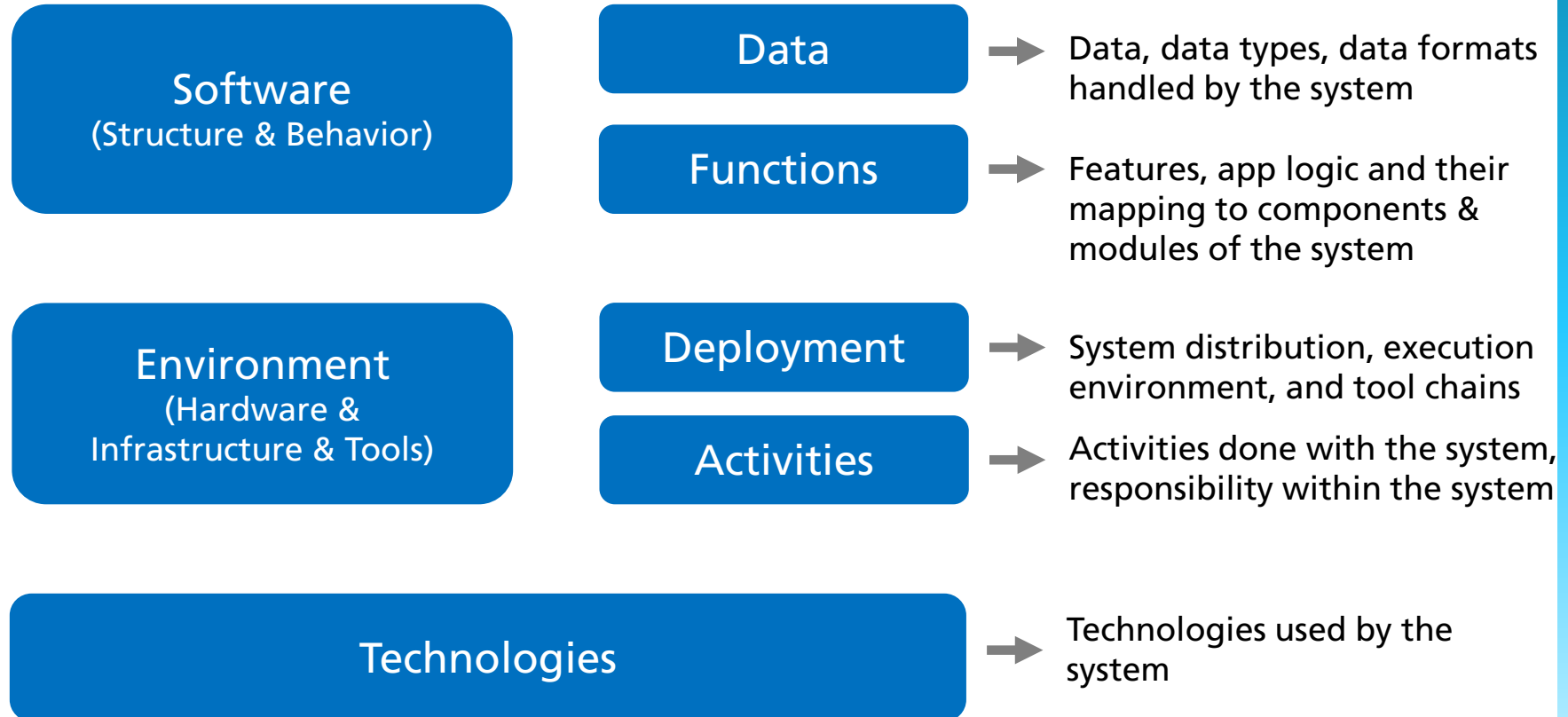
Design

Architectural Scope

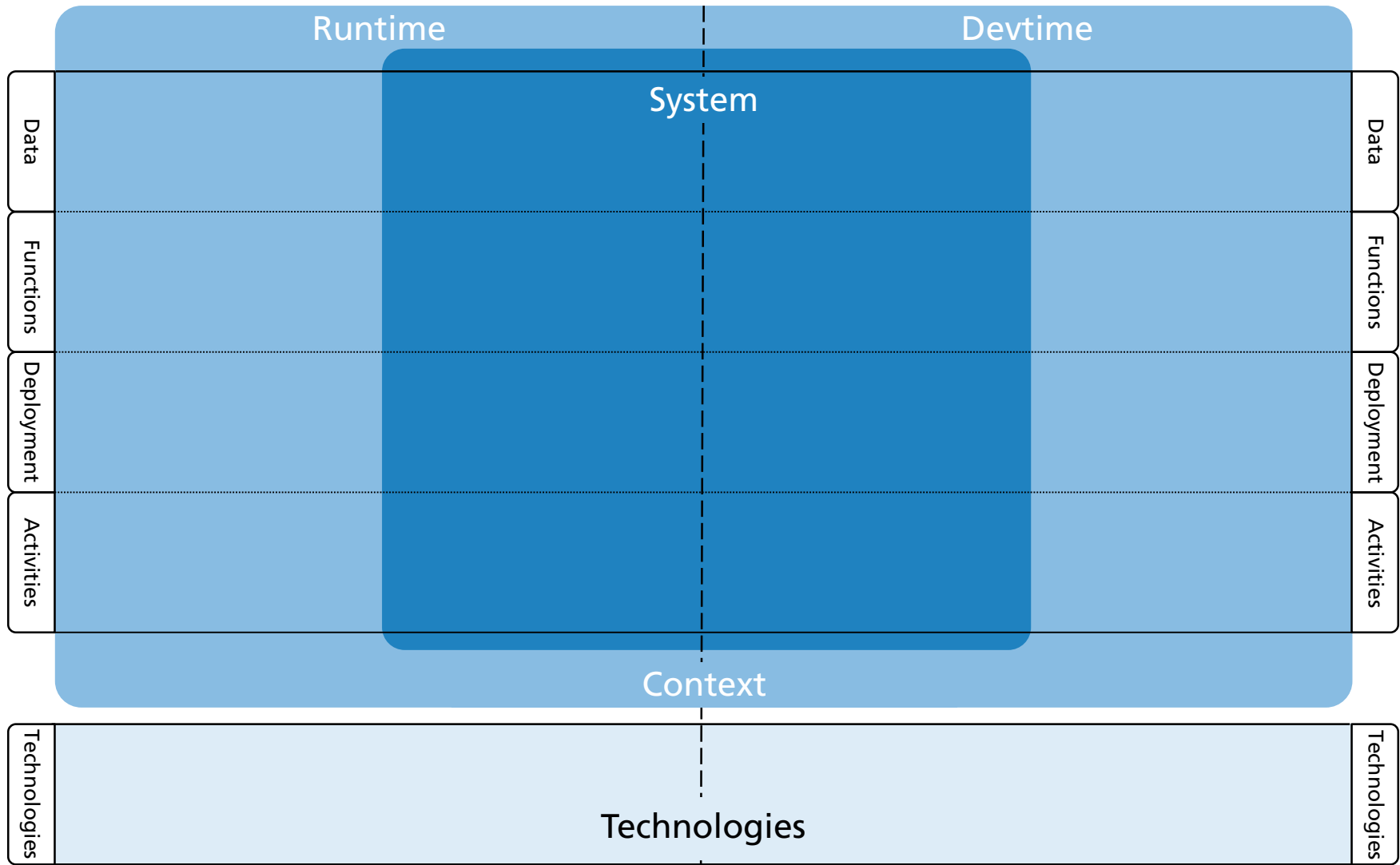


Dimensions

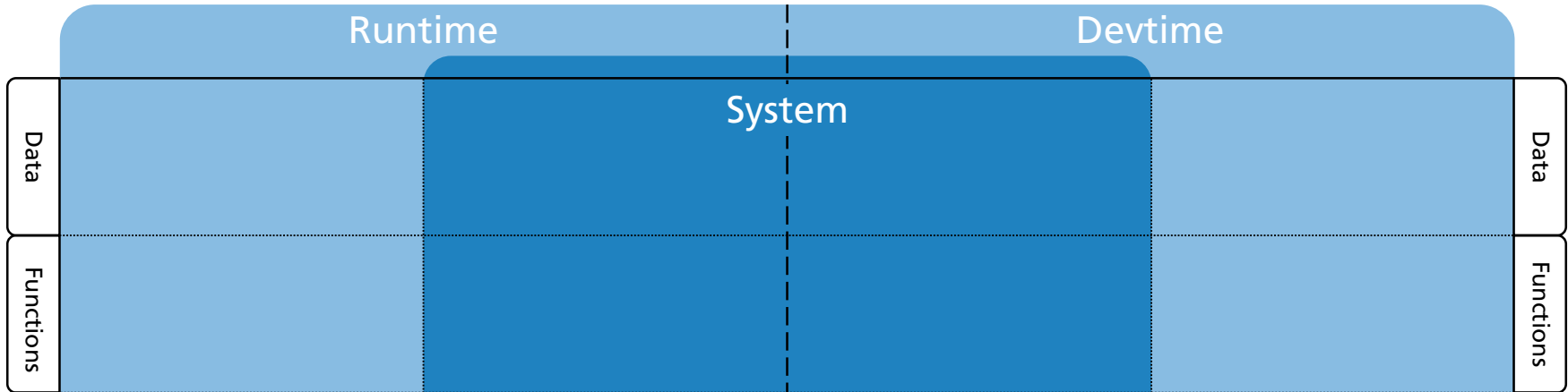
■ Decompose by addressing **different dimensions**



Data | Functions | Deployment | Activities | Technologies



Topics to think about – Data & Functions



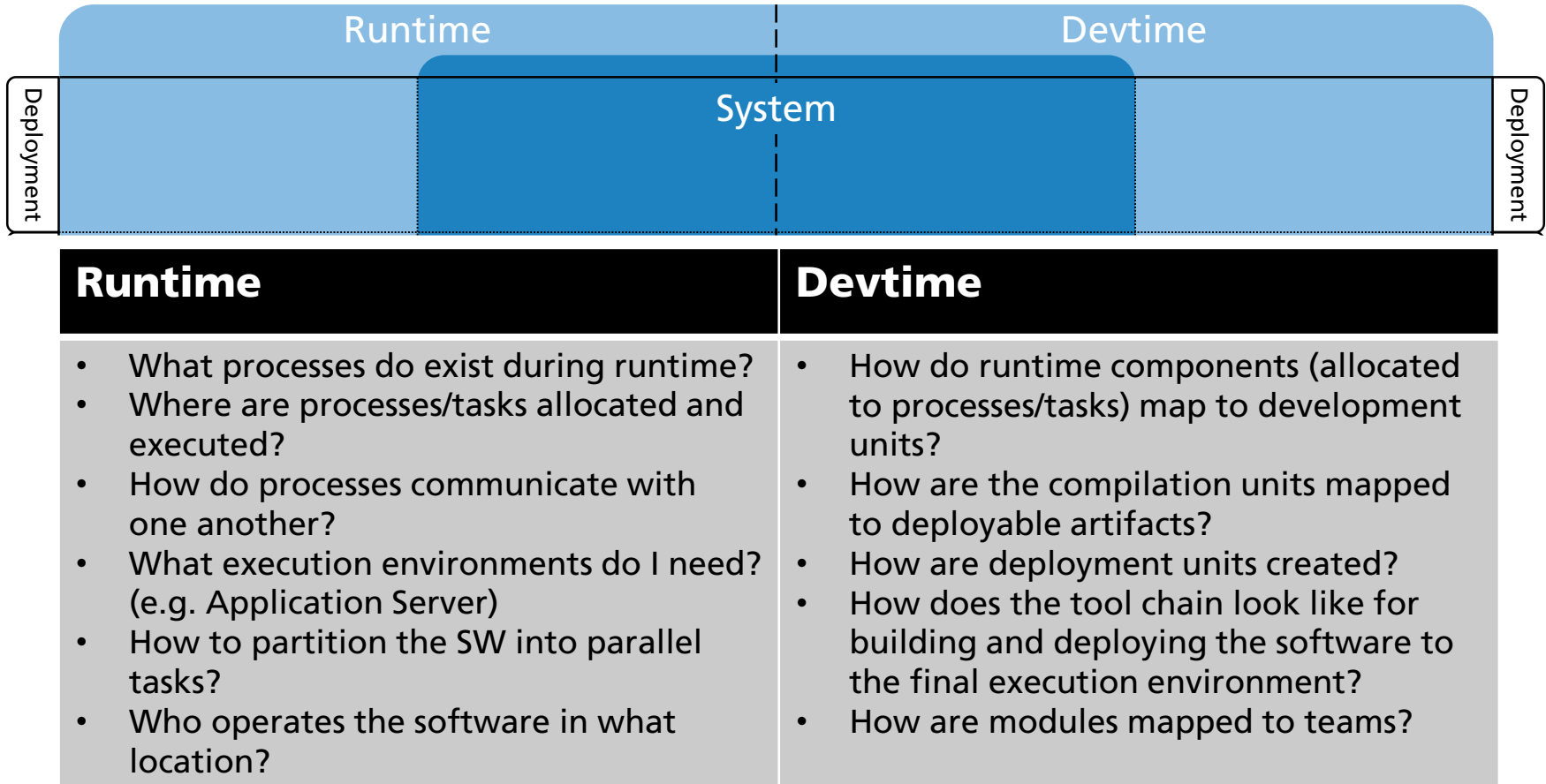
Runtime

- 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?

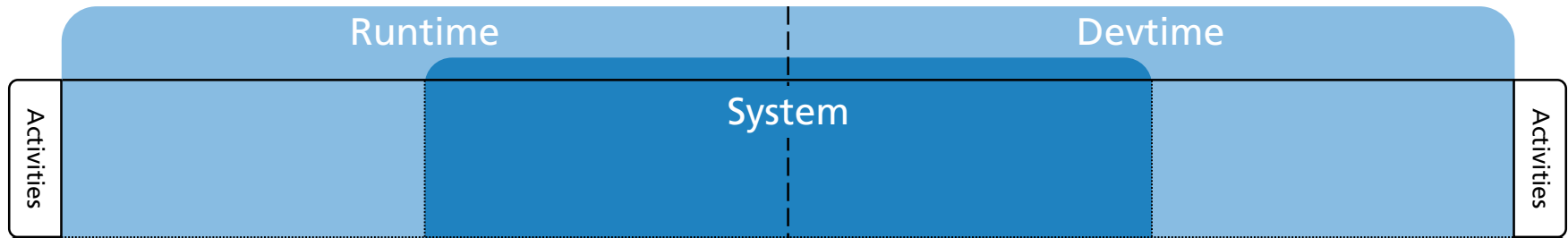
Devtime

- 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

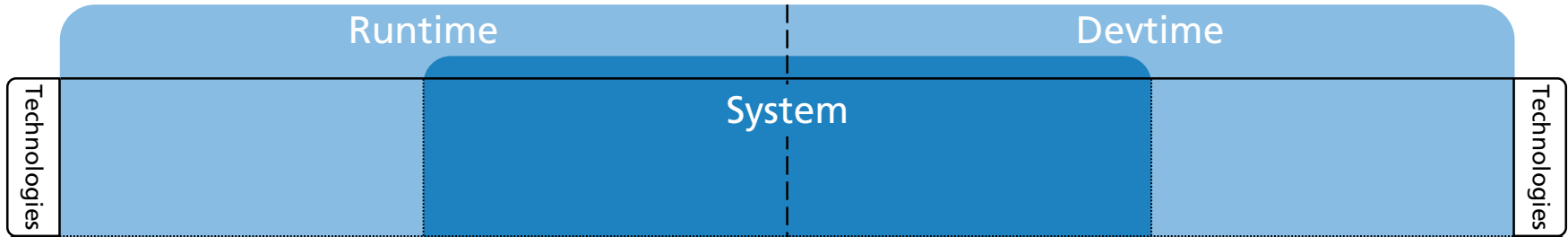


Topics to think about – Activities



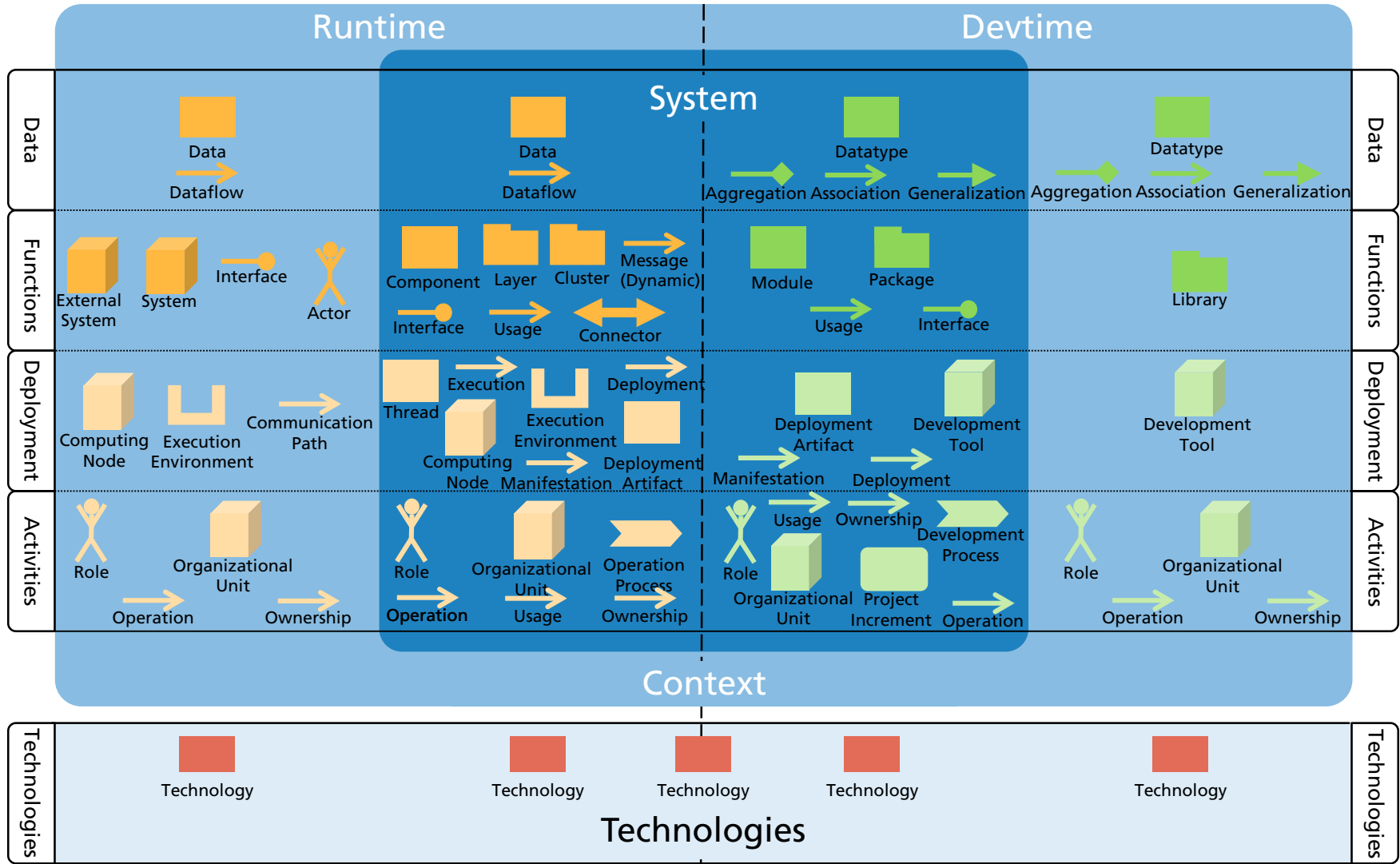
Runtime	Devtime
<ul style="list-style-type: none">• How does operation of the system look like?	<ul style="list-style-type: none">• Who is involved in the development and delivery of the software system?• Who is responsible for what system portions?• How is the development, quality assurance and delivery of the software organized in terms of processes and organizational units?• How to assign the modules to development iterations?

Topics to think about – Technologies

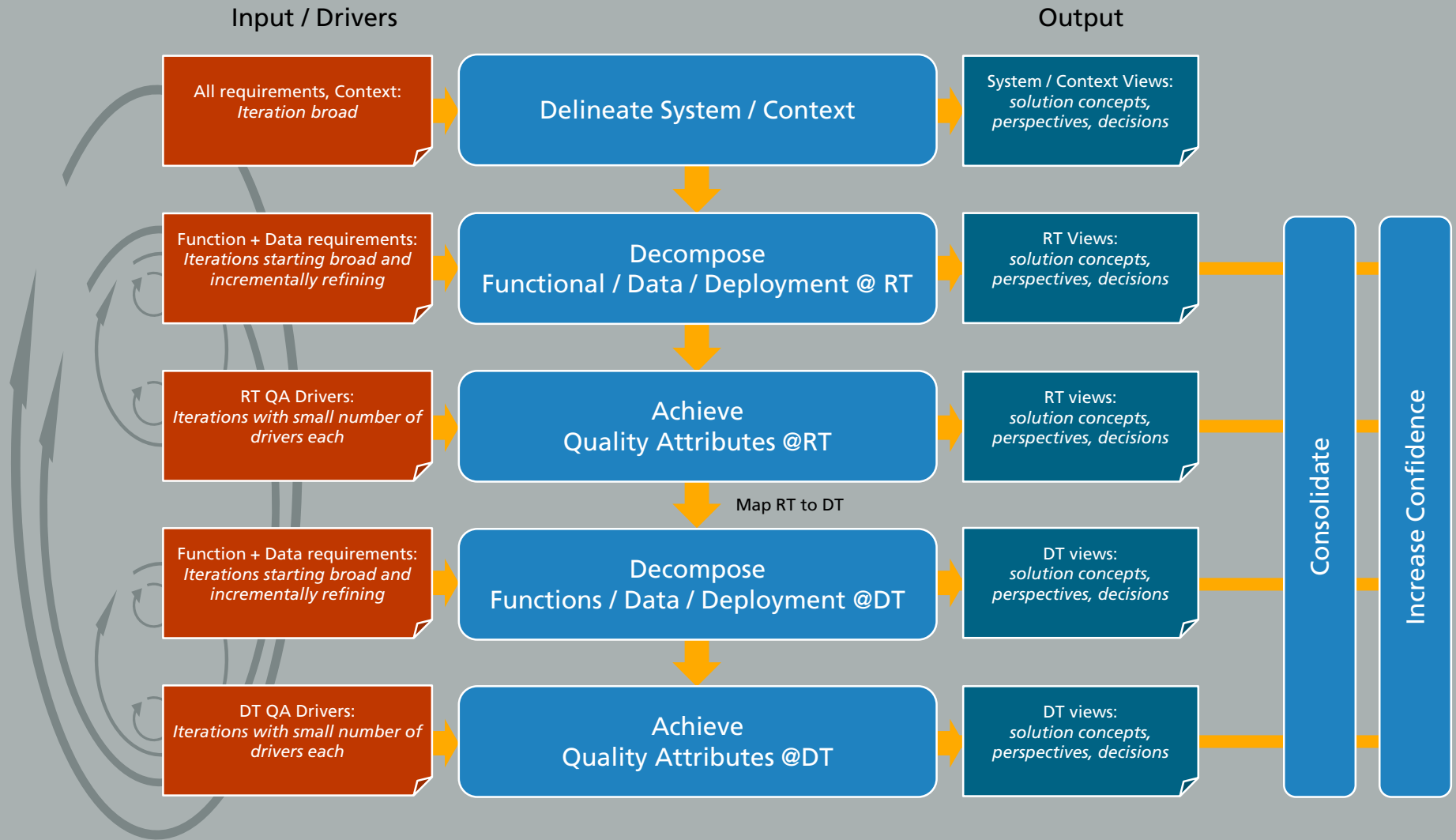


Runtime	Devtime
<ul style="list-style-type: none">• What technologies are used at runtime?• What technologies are used for communication between the system and external systems?• What are properties of candidate technologies (bandwidth, latency, throughput, processing power, energy consumption, etc.)	<ul style="list-style-type: none">• What tools are being used for development and delivery?• How are the tools connected/integrated?

Notation: Architecture Decomposition Framework



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

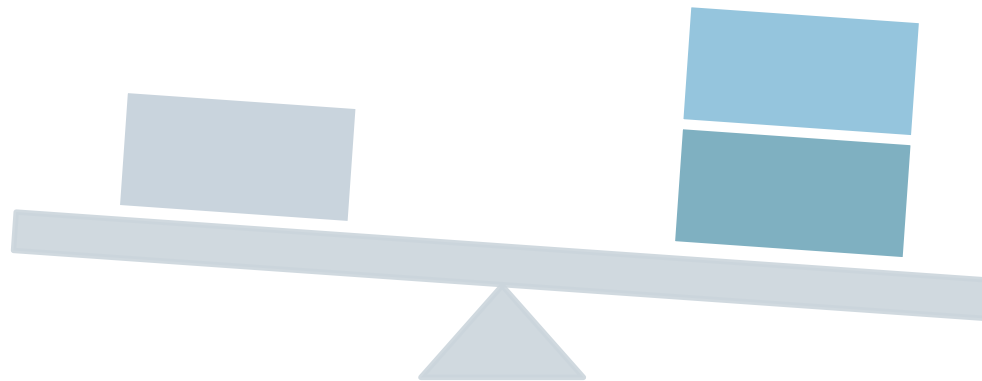
■ **Uniformity**

- 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	<i>Concise short name</i>
Design Decision ID	<i>Unique identifier</i>
Explanation	<i>Explanation of the decision rationale</i>

Pros & Opportunities <ul style="list-style-type: none"> ▪ <i>Points in favor</i> ▪ <i>Anticipations of future</i> 	Cons & Risks <ul style="list-style-type: none"> ▪ <i>Points against</i> ▪ <i>Unknown or open aspects</i>
--	---

Assumptions & Quantifications <ul style="list-style-type: none"> ▪ <i>Assumption made about the driver solution (or parts of it)</i> ▪ <i>Measurable effects applying to the driver solution (or parts of it)</i> 	Trade-Offs <ul style="list-style-type: none"> ▪ <i>Trade-offs to other design decisions, quality attributes, solutions concepts, architecture drivers</i> ▪ <i>Potentially impacted if this solution changes</i>
--	---

Manifestation Links	<i>Links to models, diagrams, additional documentation</i>
----------------------------	--

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

- Data loading time does not add on startup time

Cons & Risks

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

Assumptions & Quantifications

- Data can be loaded in 5s

Trade-Offs

- Maintainability, understandability

Manifestation Links

Driver Solution Template

Driver Name	<i>Concise short name</i>	
Driver ID	<i>Unique identifier</i>	
Steps	<ol style="list-style-type: none"> 1. <i>Logical flow to explain driver solution (white box view explaining the design)</i> 2. <i>The glue between design decisions (accepted and discarded)</i> 3. <i>Putting all related design decisions in a combined and larger context</i> 	
Related Design Decisions	ACCEPTED <ul style="list-style-type: none"> ▪ <i>Link to design decision (detailed description) to enable traceability</i> 	DISCARDED <ul style="list-style-type: none"> ▪ <i>Link to design decision (detailed description) to enable traceability</i>
Pros & Opportunities		
<ul style="list-style-type: none"> ▪ <i>Points in favor</i> ▪ <i>Anticipations of future</i> 		
Cons & Risks		
<ul style="list-style-type: none"> ▪ <i>Points against</i> ▪ <i>Unknown or open aspects</i> 		
Assumptions & Quantifications		
<ul style="list-style-type: none"> ▪ <i>Assumption made about the driver solution (or parts of it)</i> ▪ <i>Measurable effects applying to the driver solution (or parts of it)</i> 		
Trade-Offs		
<ul style="list-style-type: none"> ▪ <i>Trade-offs to other design decisions, quality attributes, solutions concepts, architecture drivers</i> ▪ <i>Potentially impacted if this solution changes</i> 		
Manifestation Links	<i>Links to models, diagrams, additional documentation</i>	

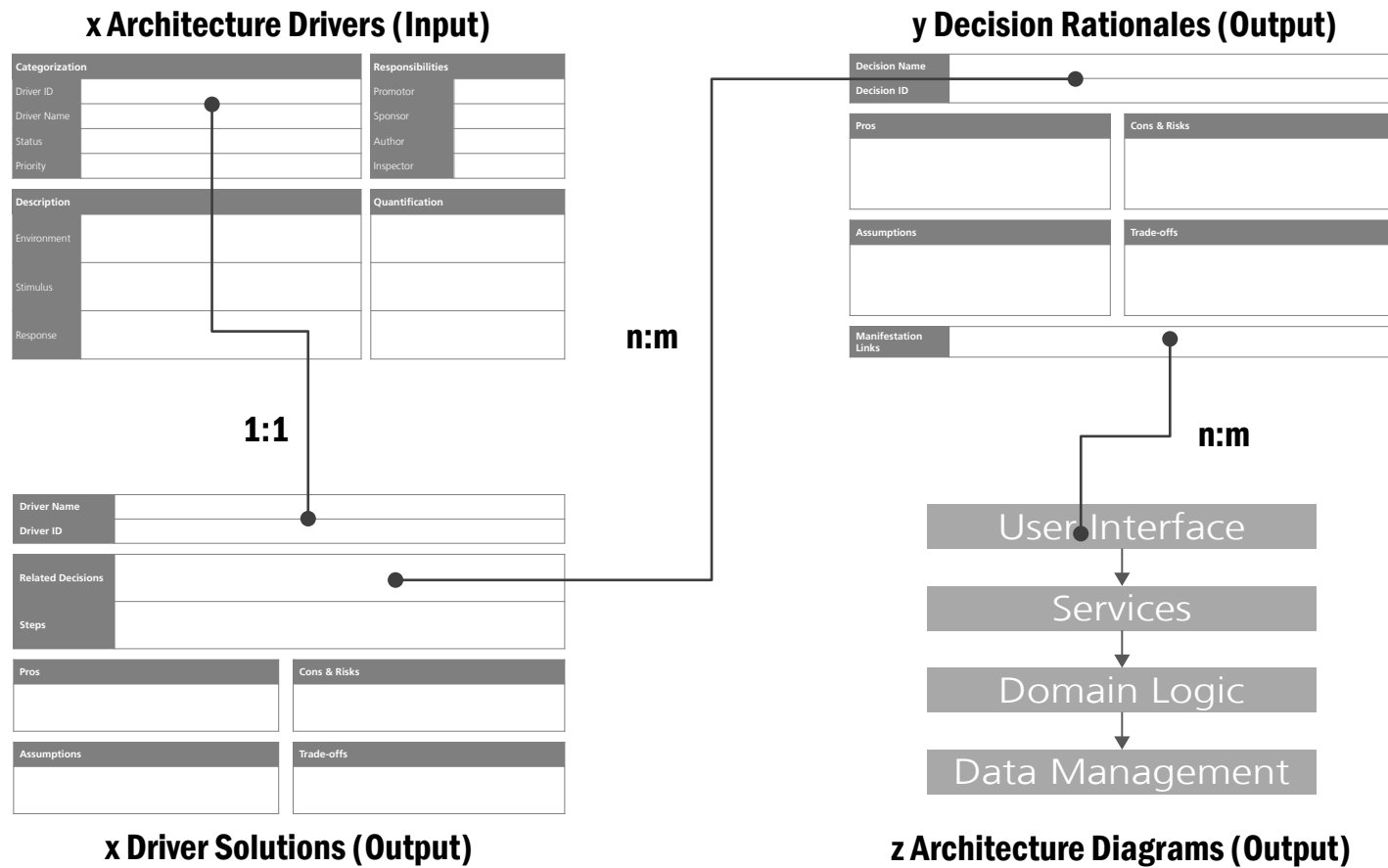
Driver Solution Example

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

Steps	<ol style="list-style-type: none">1. Application always stores preprocessed index-structures on updates of searchable items2. On startup, loading of search data is moved to a separate thread3. The UI is started and ready for user input while loading of search data is ongoing4. After loading the search data, searches can be done without the user noticing that search was not available before
Related Design Decisions	<ul style="list-style-type: none">▪ DD.01 Decoupled loading of search data▪ DD.12 Preprocessed index-structures of search data

Pros & Opportunities	Cons & Risks
<ul style="list-style-type: none">▪ Very fast startup time, application directly usable by user	<ul style="list-style-type: none">▪ More effort in realization▪ Loading in separate thread requires synchronization and makes implementation more difficult

Assumptions & Quantifications	Trade-Offs
<ul style="list-style-type: none">▪ Data can be loaded in 5s▪ User rarely sends a search in less than 4s after start is completed	<ul style="list-style-type: none">▪ Maintainability, understandability



x Architecture Drivers (Input)

Categorization		Responsibilities	
Driver ID		Promotor	
Driver Name		Sponsor	
Status		Author	
Priority		Inspector	
Description		Quantification	
Environment			
Stimulus			
Response			

INPUT

1:1

Driver Name	
Driver ID	
Related Decisions	
Steps	
Pros	
Assumptions	
Trade-offs	

OUTPUT

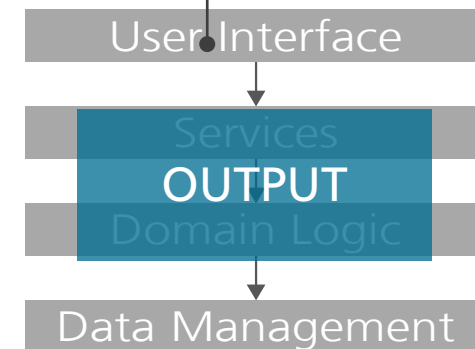
x Driver Solutions (Output)

y Decision Rationales (Output)

Decision Name	
Decision ID	
Pros	
Cons & Risks	
Assumptions	
Trade-offs	
Manifestation Links	

OUTPUT

n:m



z Architecture Diagrams (Output)

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

Documentation

Who uses the Architecture Documentation?



Developer



Architect



Projectleader



Manager



Quality Engineer



Marketing

Functions (ART)

System Interface

User Interface

System Interface

Registration

Audio

Inputs

Applications

Monitoring App

Diagnosis App

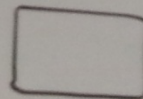
Medical Report

The minimalistic form

Data Management

----- System Boundary

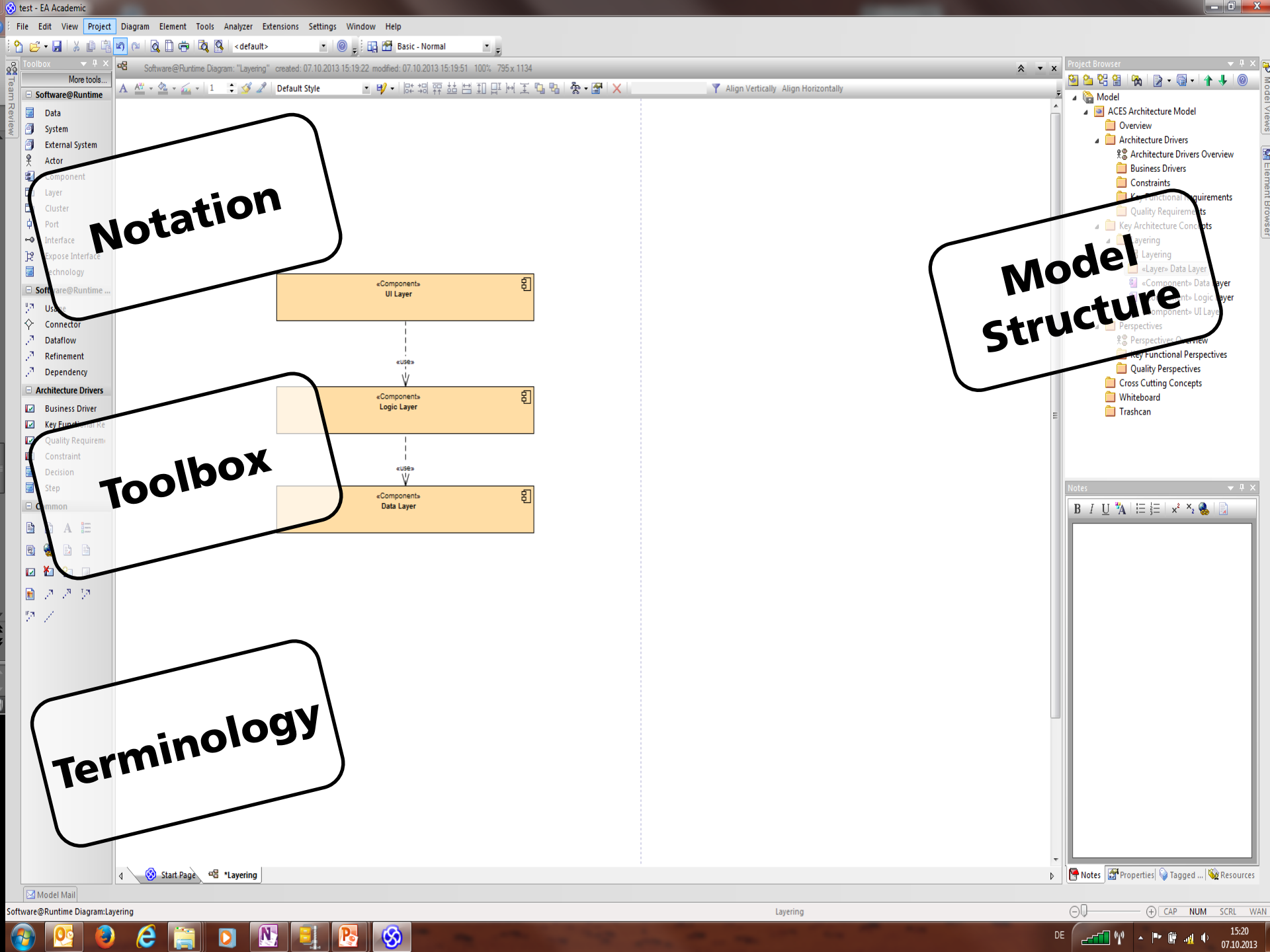
→ uses



Container



Component



Notation

Toolbox

Terminology

Model Structure

Software Architecture Document – Example

Example Architecture Document

Authors:
Matthias Naab
Dominik Rost

A publication by Fraunhofer IESE

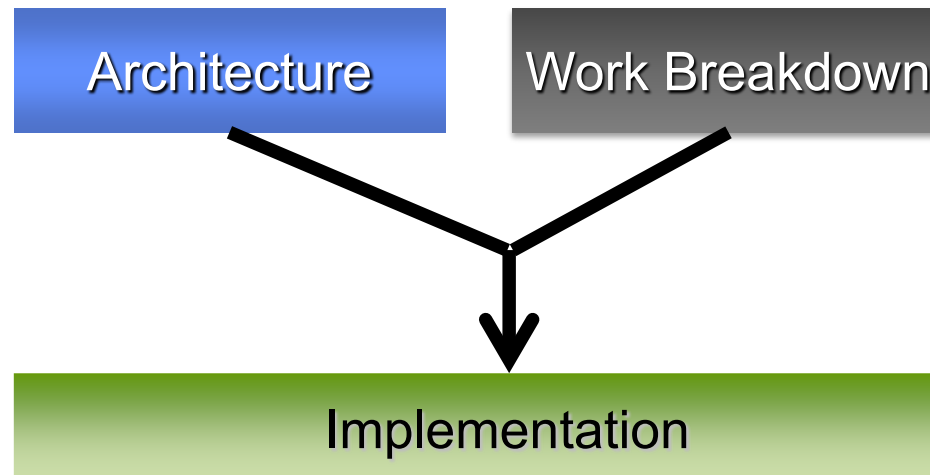
1	Introduction	1
1.1	Project Context & Document Goals	1
1.2	System Goals	1
1.3	Stakeholders	1
1.4	Stakeholder Reading Guide	2
2	Architecture Drivers	3
2.1	Business Goals	3
2.2	Key Functional Requirements	5
2.3	Quality Requirements	7
2.4	Constraints	10
3	Architecture Overview and Key Architecture Concepts	12
3.1	Context Delineation	12
3.2	Component Overview	13
3.3	Platform-Product-Relationship	16
3.4	Layering	16
4	Detailed Solutions for Architectural Drivers	18
4.1	Deployability	18
4.2	Decentralization of Computation	19
4.3	Performance	21
5	Solution Concepts for Cross Cutting Concerns	22
5.1	Internationalization	22
5.2	Logging	22
5.3	Multi Tenancy	22
5.4	License Models	22
5.5	...	22
6	Methodology	23
6.1	Fraunhofer ACES ADF	23
7	Glossary	24

Tool Support for Architecture Modeling

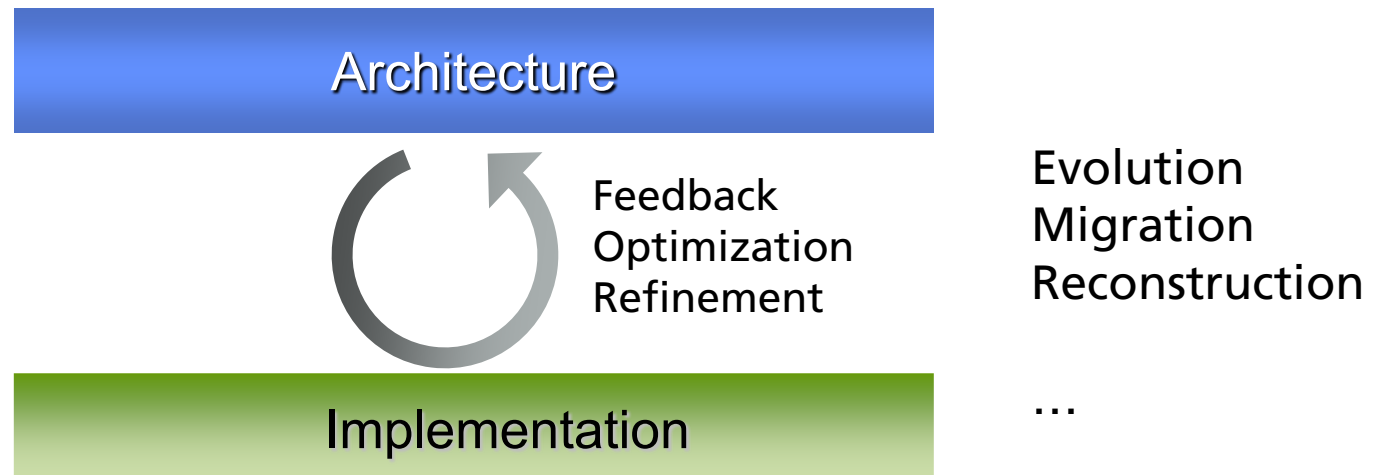
- First use paper
- Then use some Modeling tool (Enterprise Architect for example)
- Use word for documenting things

Prediction and Control

You can start your implementation



In the real world the story only starts...



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



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