# Safety Architectures Competence @ Fraunhofer IESE

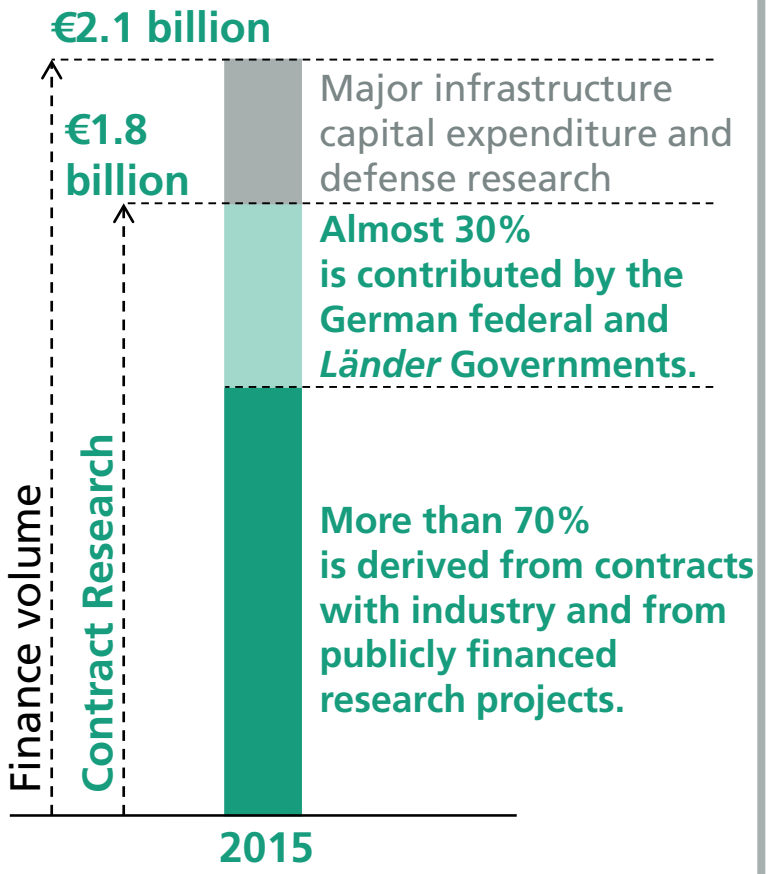

**Pablo Oliveira Antonino**
**April 11, 2016**

# The Fraunhofer-Gesellschaft at a Glance

**The Fraunhofer-Gesellschaft undertakes applied research of direct utility to private and public enterprise and of wide benefit to society.**

**24,000** staff


**67** institutes and research units








Fraunhofer IESE

# Fraunhofer IESE

## The institute for software and systems engineering methods

- Founded in 1996, headquartered in Kaiserslautern
- Over 155 full-time equivalents (FTEs)
- Our solutions can be scaled flexibly and are suitable for companies of any size

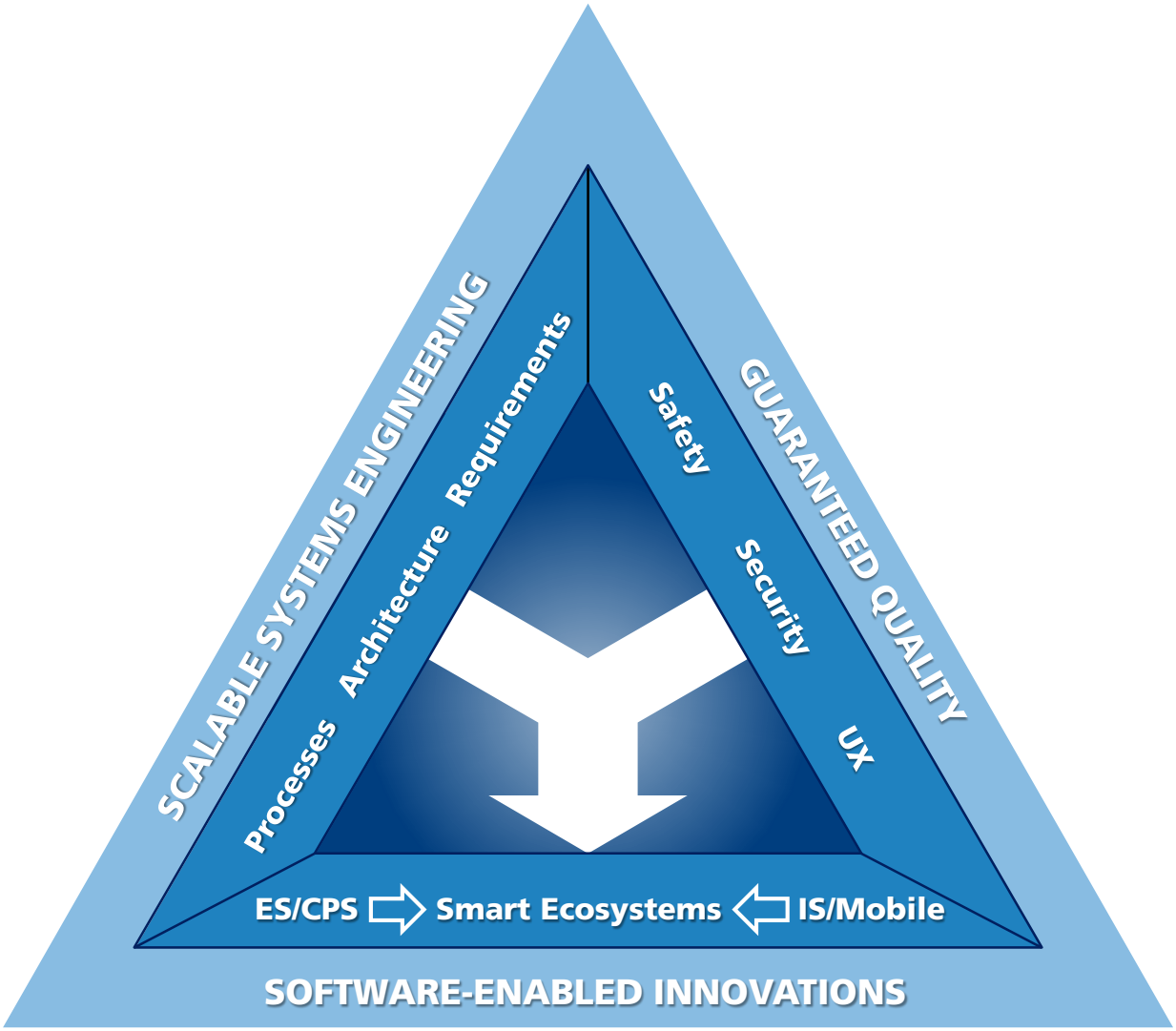
- Our most important business areas:
  - Automotive and Transportation Systems
  - Automation and Plant Engineering
  - Health Care
  - Information Systems
  - Energy Management
  - E-Government





Fraunhofer IESE

# Our Competencies – for Your Benefit








4

Fraunhofer IESE

# IESE Organizational Chart

**Executive Director**
Prof. Dr.-Ing. Peter Liggesmeyer

**Deputy Director**
Prof. Dr. Frank Bomarius

**Director Business Development**
Prof. Dr. Dieter Rombach

**Division Embedded Systems (ES)**
**Dr. Mario Trapp**

- **Embedded Systems Engineering (ESY)** — Dr. Martin Becker
- **Embedded Software Engineering (ESW)** — Dr. Thomas Kuhn
- **Embedded Systems Quality Assurance (ESQ)** — Dr. Daniel Schneider

**Division Process Management (PM)**
**Dr. Jens Heidrich**

- **Data Engineering (DE)** — Dr. Andreas Jedlitschka
- **Process Engineering (PE)** — Rolf van Lengen

**Division Information Systems (IS)**
**Dr. Jörg Dörr**

- **User Experience and Requirements Engineering (UXR)** — Dr. Marcus Trapp
- **Architecture Centric Engineering (ACE)** — Dr. Matthias Naab
- **Security Engineering (SE)** — Christian Jung

**Central Services**
**Ilona Würtz (acting)**

**Corporate Communications & Technology Marketing**
**Nicole Spanier-Baro**

**Business Areas**
**Ralf Kalmar** — **Michael Ochs**

- **Automotive & Transportation Systems**
- **Automation & Plant Engineering**
- **Energy Management**
- **Information Systems**
- **eGovernment**
- **Health Care**





Fraunhofer IESE

# Top Industry Customers in 2015









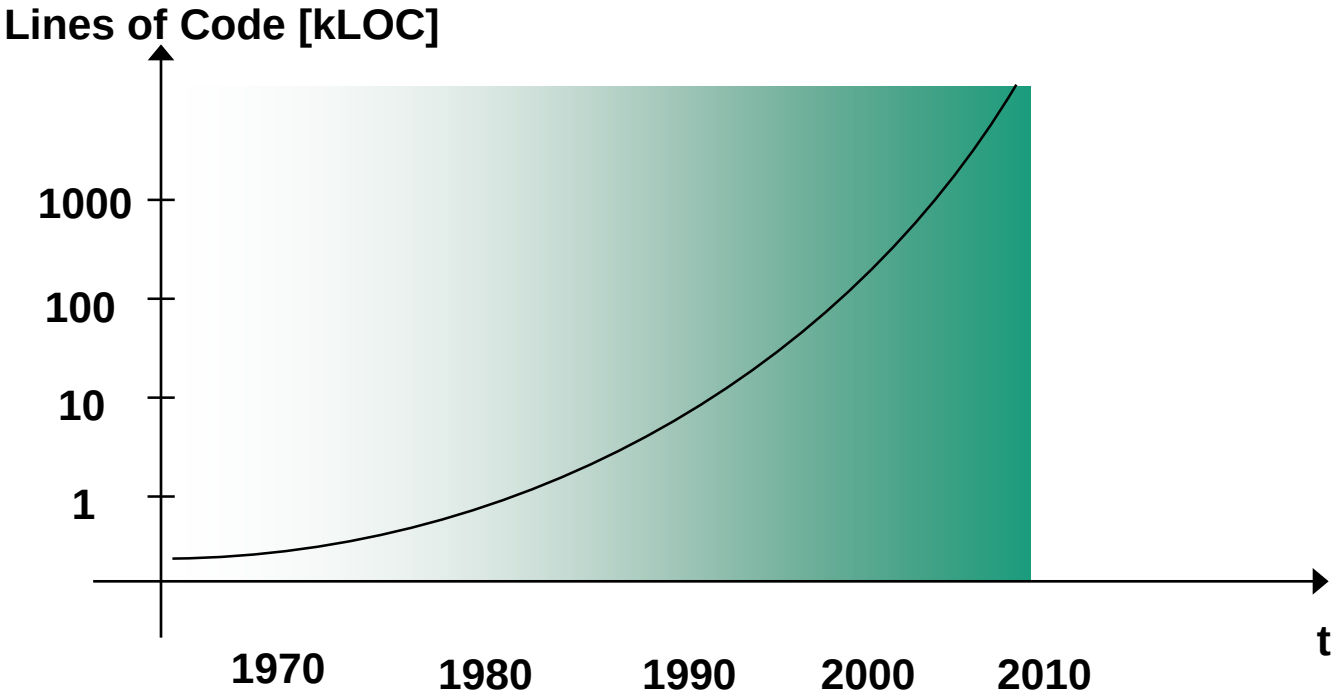





6

# Hello, Architecture!

# **Engineering Challenge:** Large-Scale Systems




- Examples
  - Car window opener 10.000 LoC
  - Car control unit 15.000.000 LoC
  - Windows XP 40.000.000 LoC

# **Engineering Challenge:** Large Development Teams

- Large teams have to collaborate.
- Teams
  - Distributed over buildings, countries, continents;
  - Distributed over departments, organizations.
- Decomposition of work for parallelization is essential.

# **Engineering Challenge:** High Quality

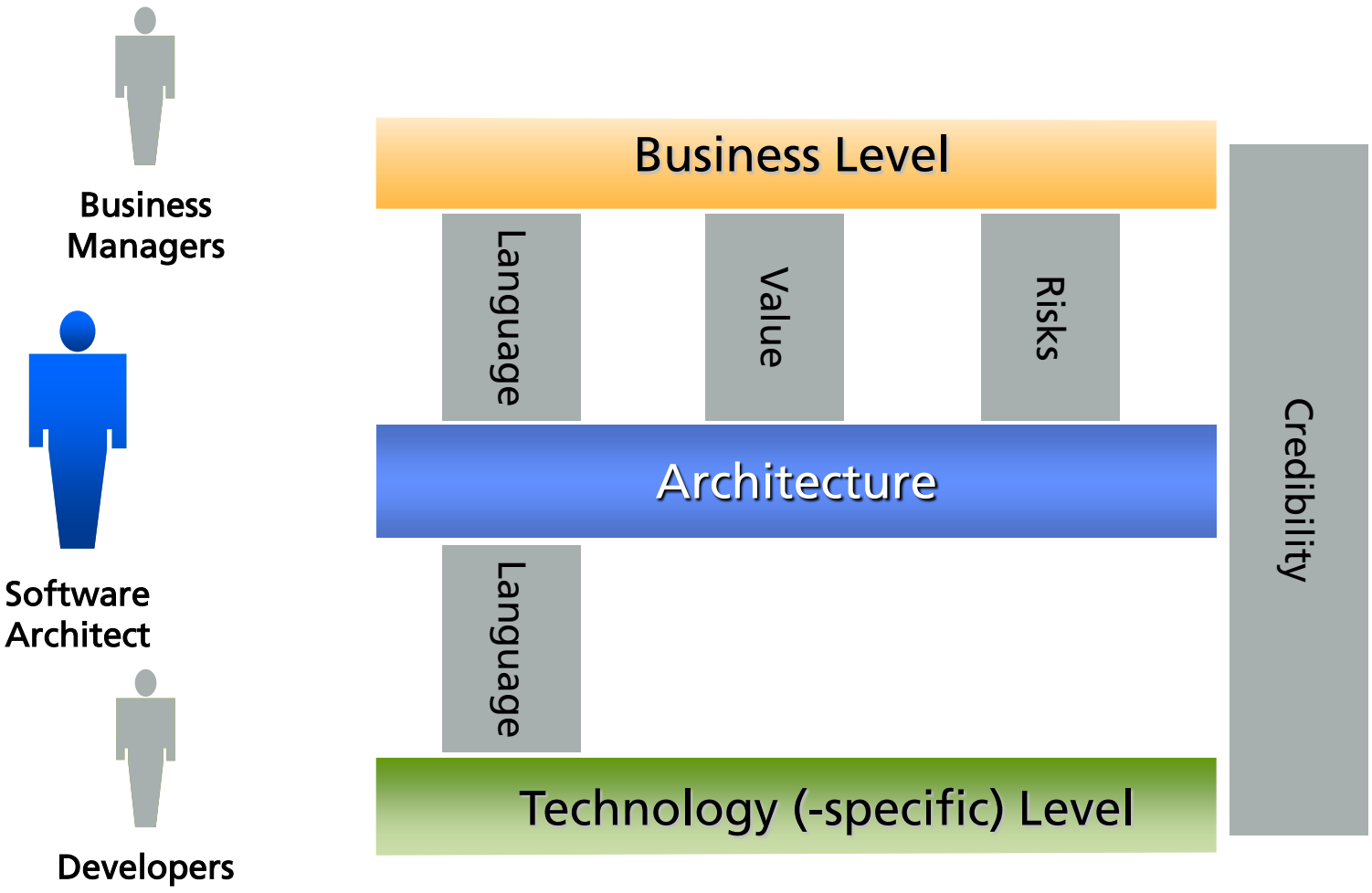
Quality is not only about **correctness of functionality**

Successful software systems have to assure additional properties

- Performance
- Security
- Safety
- Availability
- Maintainability
- …

These properties are the so-called **Quality Attributes**

# Architecture as a Mediator and Communicator



# Architectures…

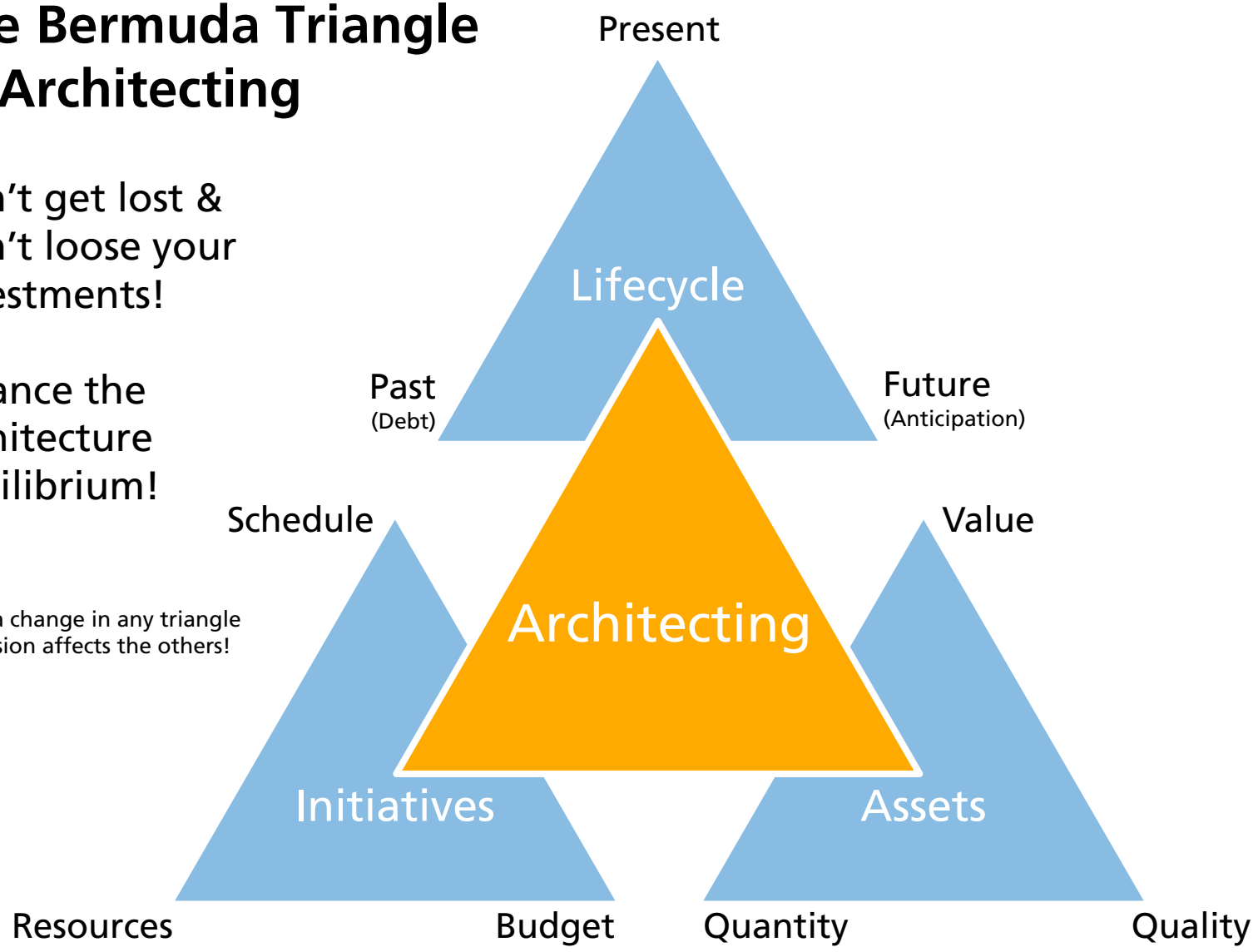
- **… provide guidance**
  - Plan for constructing a system
  - Technical leadership and coordination
  - Standards and consistency
- **… enable communication**
  - Clear technical vision and roadmap
  - Explicit documentation for communication
- **… balance technical risks**
  - Identification and mitigation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

# The Bermuda Triangle of Architecting

Don’t get lost &
Don’t loose your investments!

Balance the architecture equilibrium!

Note: a change in any triangle dimension affects the others!

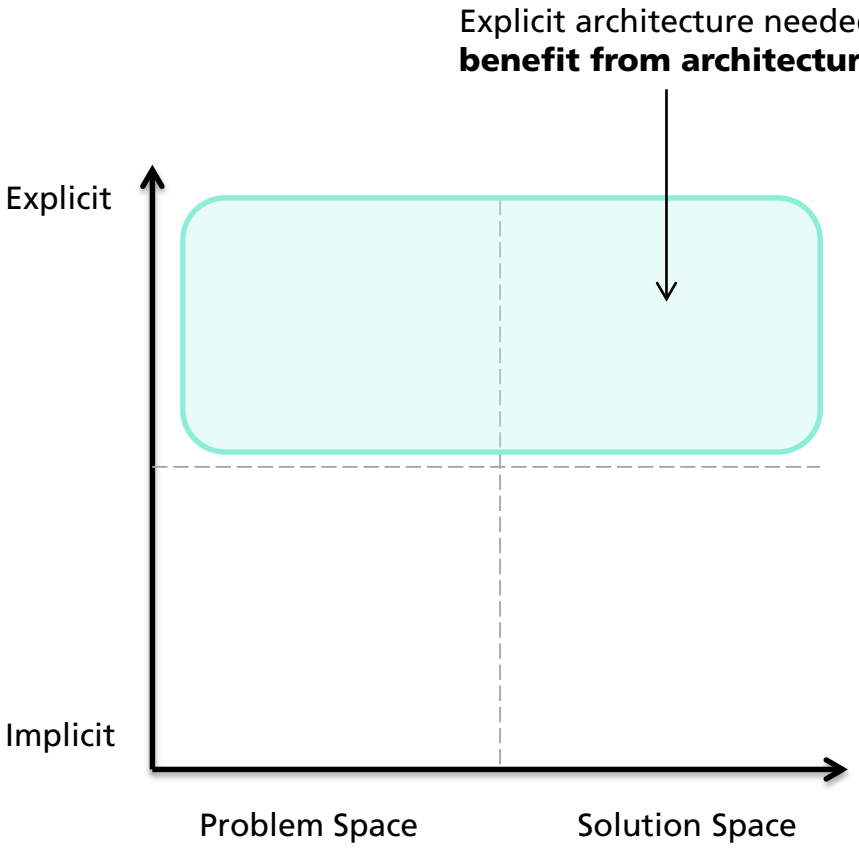







Fraunhofer IESE

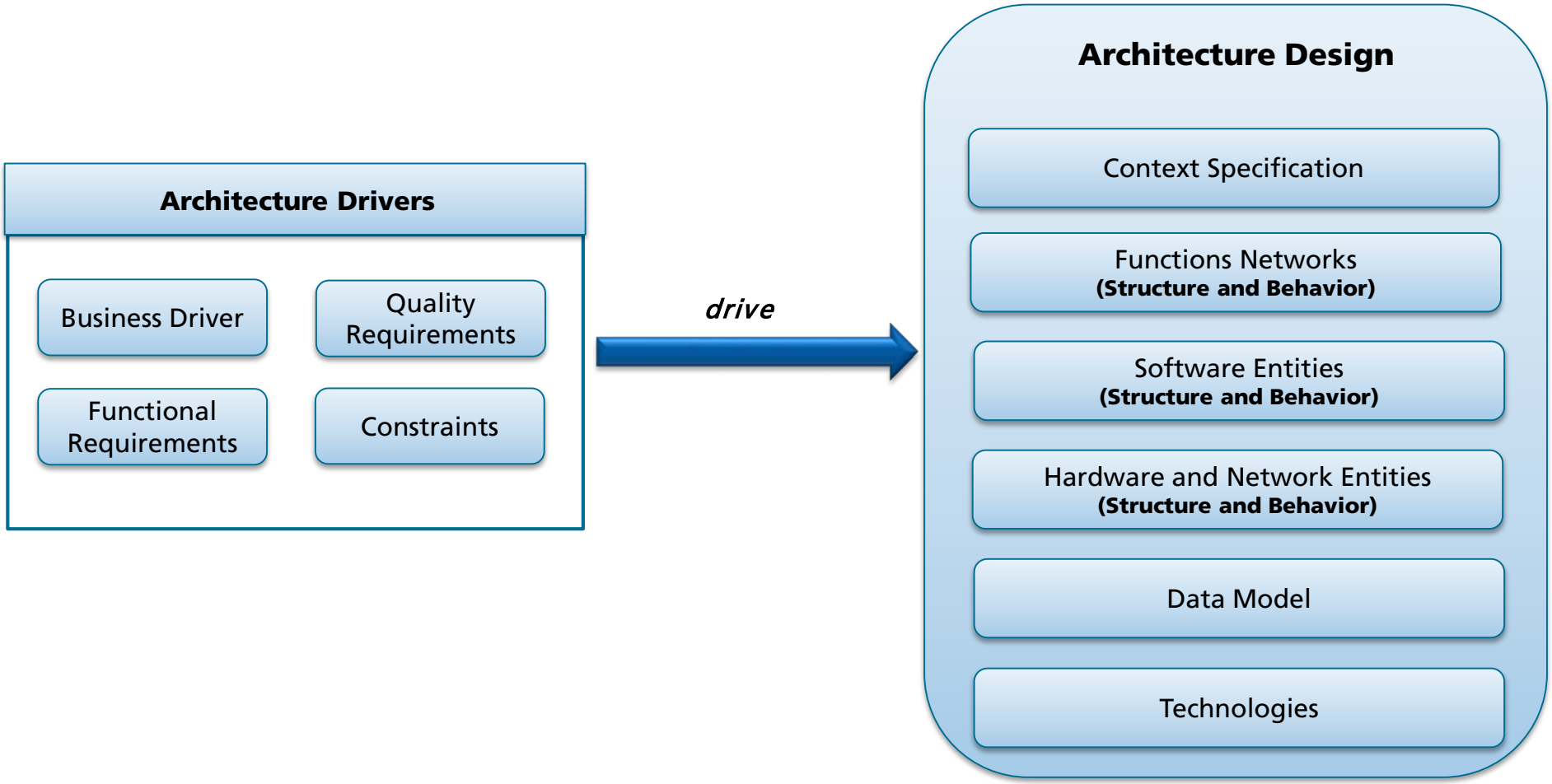
# What do We Need in Terms of Architecture?




- Implicit architecture
  - Fuzzy ideas in minds of engineers;
  - Only exists at implementation level;
  - Can only be communicated verbally.

- Explicit architecture
  - Modeled / documented;
  - Contains the information needed;
  - Can be intended or implemented architecture (“real world”).

# Architecture Drivers and Architecture Design



# Architecture Drivers

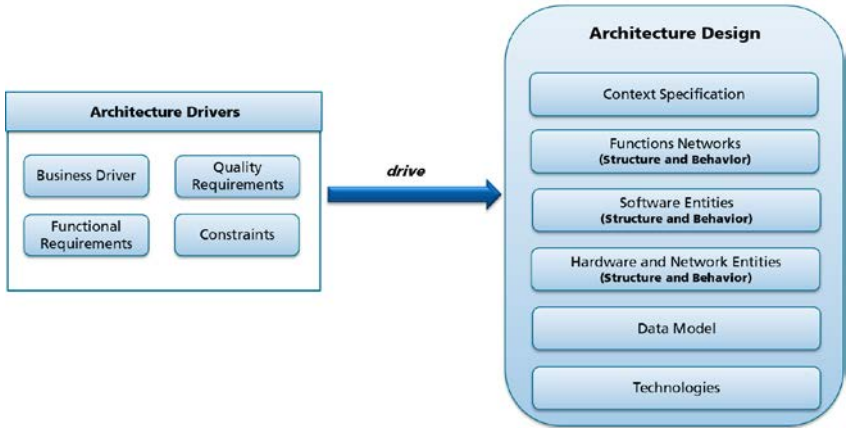
# What Drives my Architecture?

- Whatever is…
  - Costly to change
  - Risky
  - New

**With respect to stakeholders' concerns**

# Architectural Drivers

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



# Compensation of Architectural Drivers

**What we typically find in practice as architects**

- *Business goals*: often found, but not well understood
- *Functional requirements*: often found
- *Runtime quality attributes*: often found, but not specific enough
- *Devtime quality attributes*: rarely found, seldom specific
- *Operation quality attributes*: rarely found
- *Constraints*: often found, but not always really fix

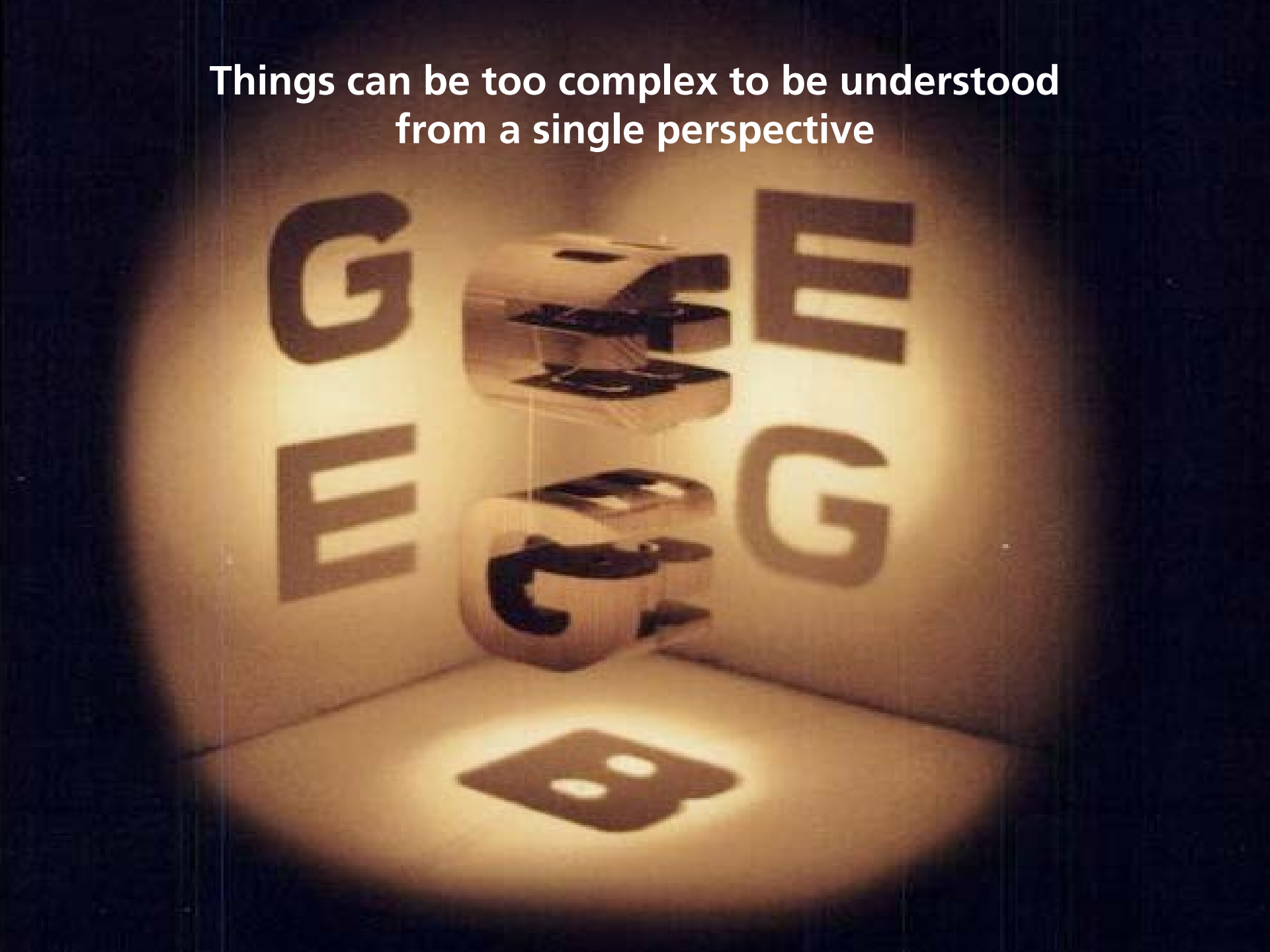
→ Architects have spend work for compensation of architectural drivers

# Architectural Drivers – Examples

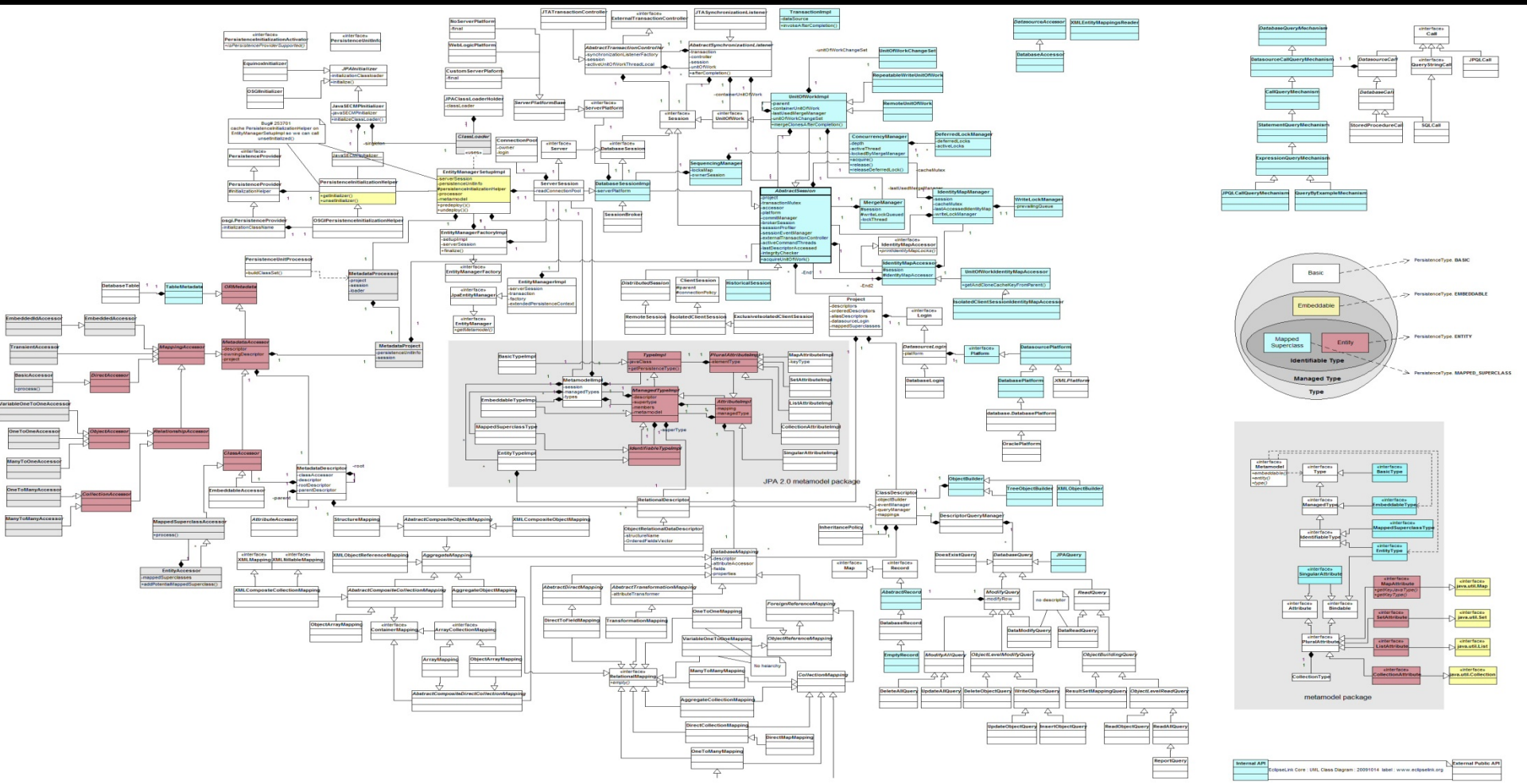
- „A user wants to update the system. The update is triggered with a maximum of 3 clicks. “
- „During operation, a single sensor fails. All ongoing operations are unaffected by the failure“
- „Each user input generates a visual response within 0.2 s“
- „A new feature is to be implemented. A team of 5 people is able to realize the feature within three days“
- „We are not allowed to use Open Source software at all“
- „All our components have to be AUTOSAR compliant“

# Architecture Design

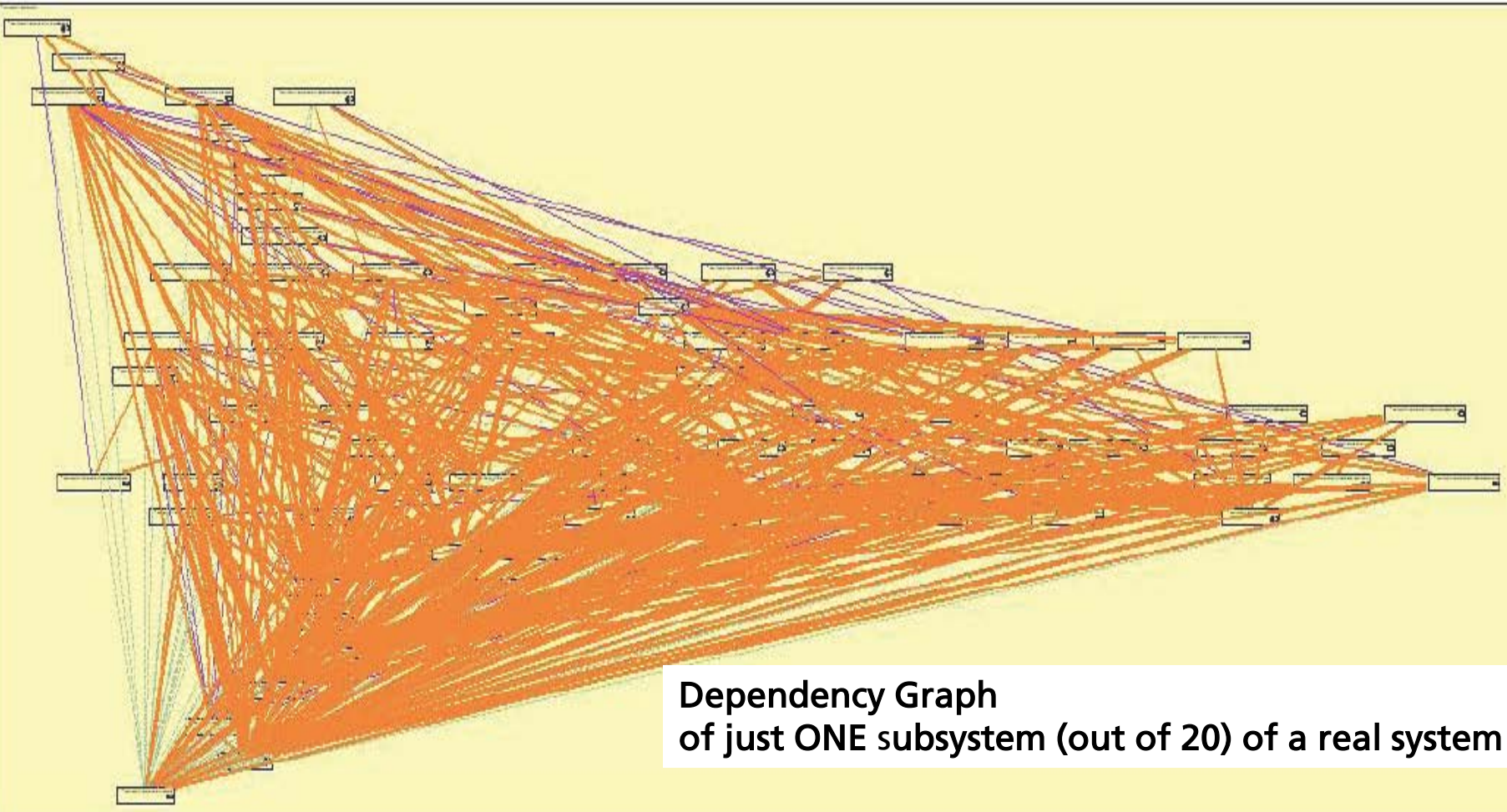
Things can be too complex to be understood from a single perspective



# But some try nevertheless …



# … and fail

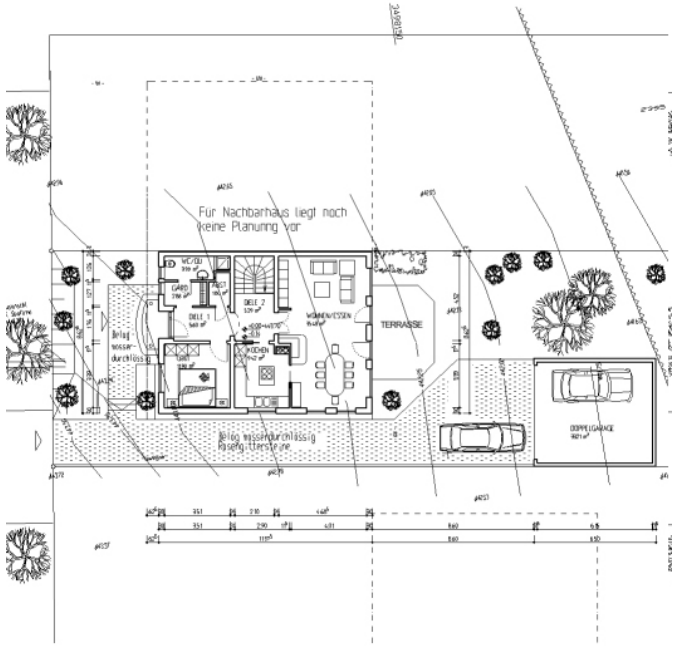


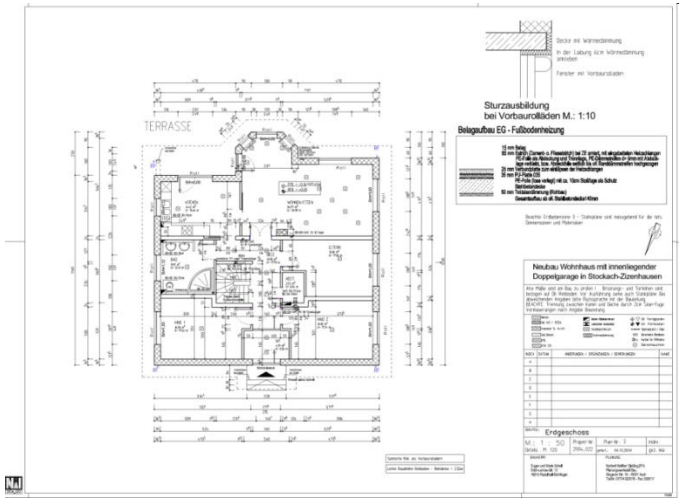
Dependency Graph
of just ONE subsystem (out of 20) of a real system

**"It is not possible to capture the functional features and quality properties of a complex system in a single comprehensible model that is understandable by and of value to all stakeholders"**

[Rozanski, Woods, 2005]

# Analogy – Views on a Building


http://www.planungswerkstatt-bau.de





Fraunhofer IESE

# What Determines the Views in Building Architecture?

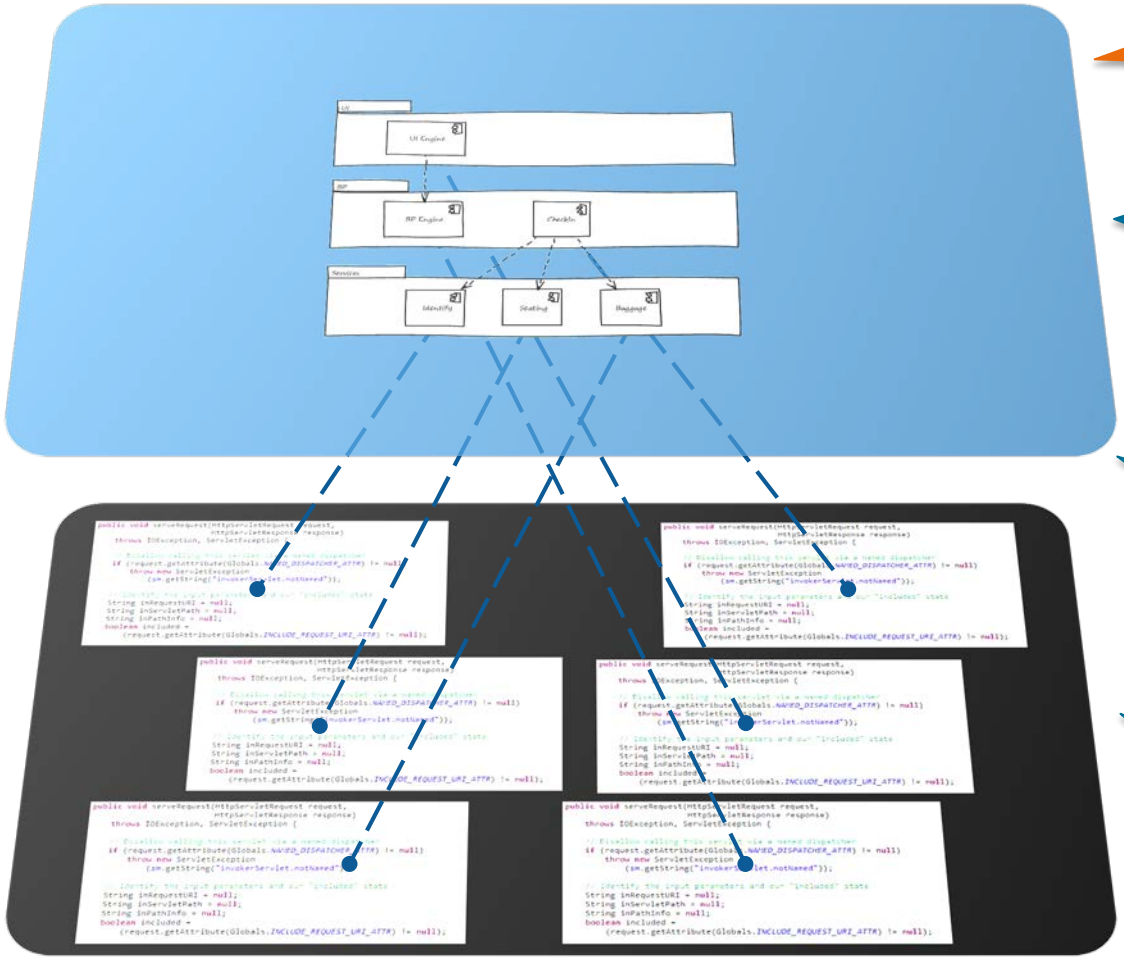
- 3-dimensional world and metrics
- Physics
- Crafts (plumbing, electricity, …)

# What Determines the Views in Software Architecture?

# Abstraction
# In the end, it's about the Code… but

Architecture

Implementation



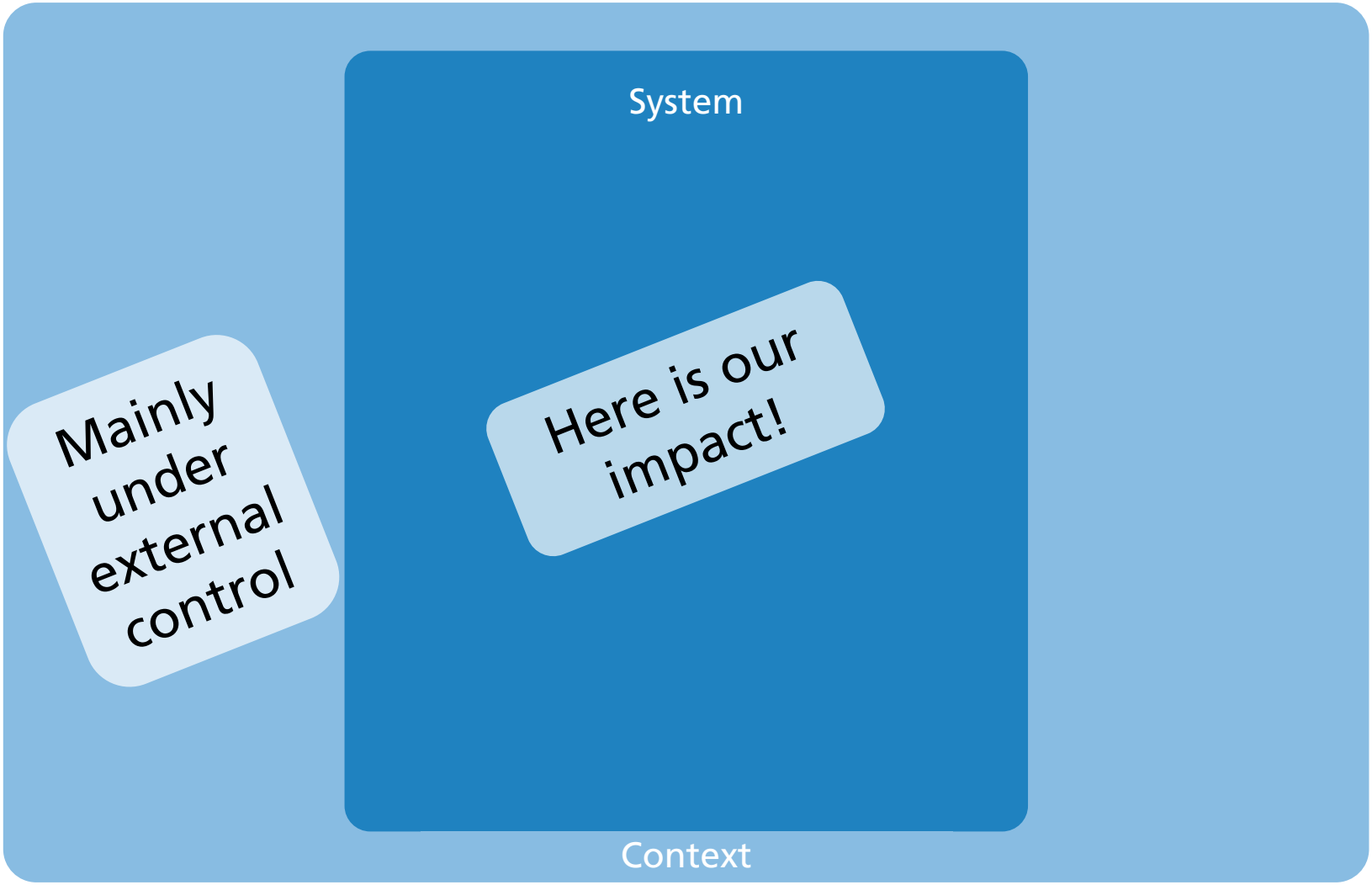
Investment

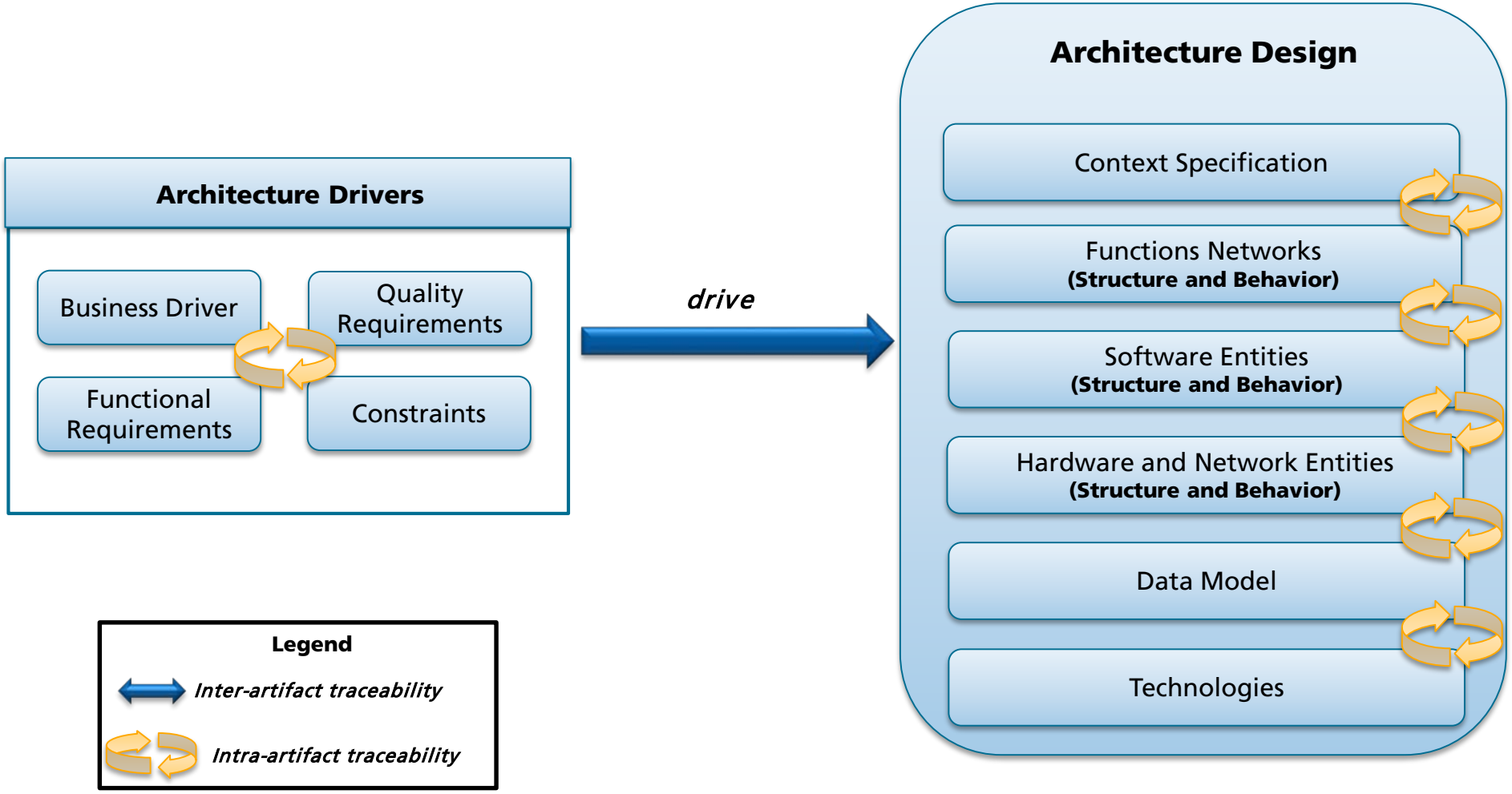
Model is Abstraction
(easy to change)

Prediction
(early)

Governance
(late)

# Architectural Scope






Refine and specify the decomposition by addressing **different aspects**

# The Embedded Modeling Profile

# Modelling Profile for Embedded Systems Development

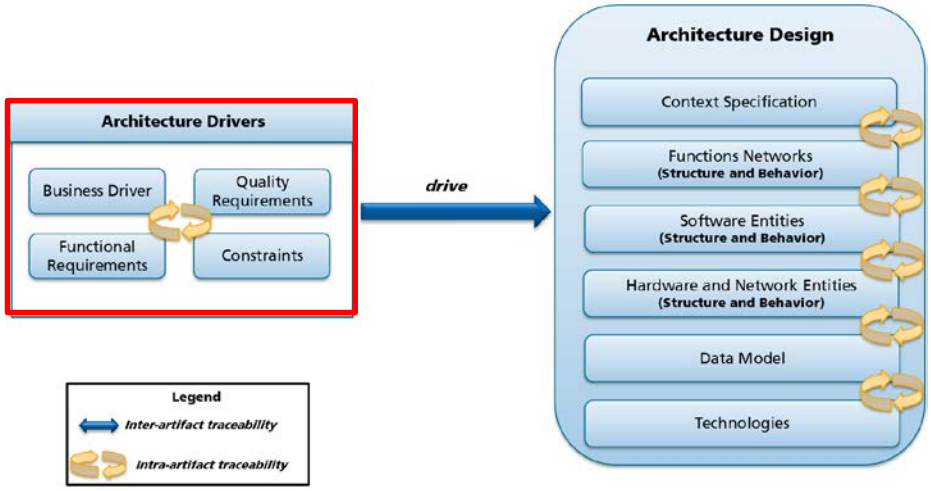
## Tailoring of UML/SysML

- Add support for modeling of system concepts for embedded systems
- Based on results of SPES 2020 and SPES XT project

## SPES

- Innovation alliance with 21 Partners from Industry and academia
- Development of Software Development Platform for Embedded Systems

# Architecture Drivers



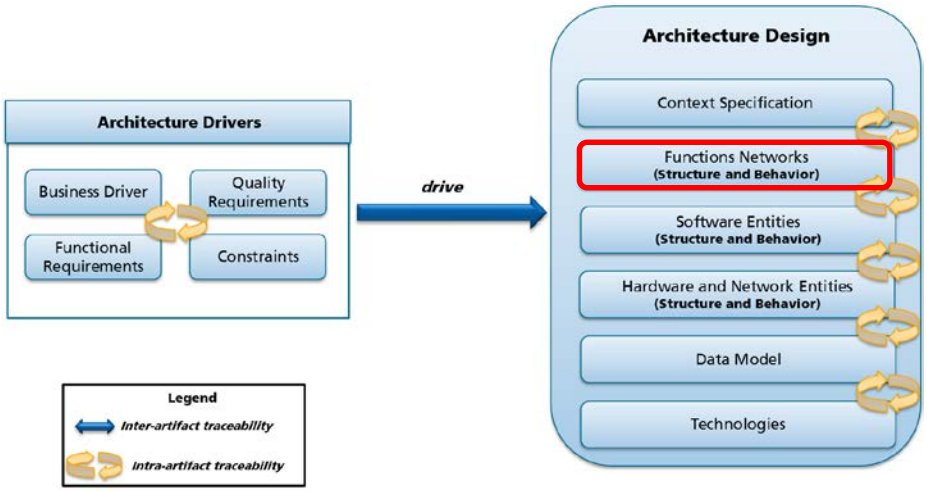


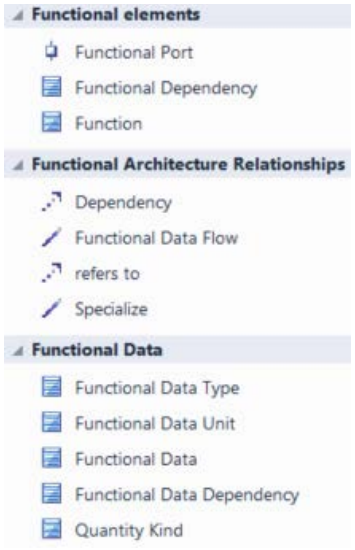




Fraunhofer IESE

# Function Networks



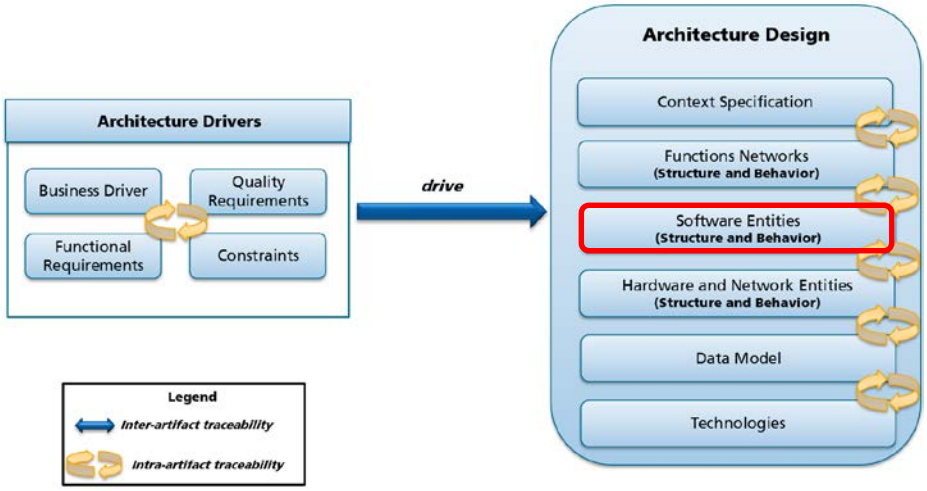








Fraunhofer IESE

# Software Entities



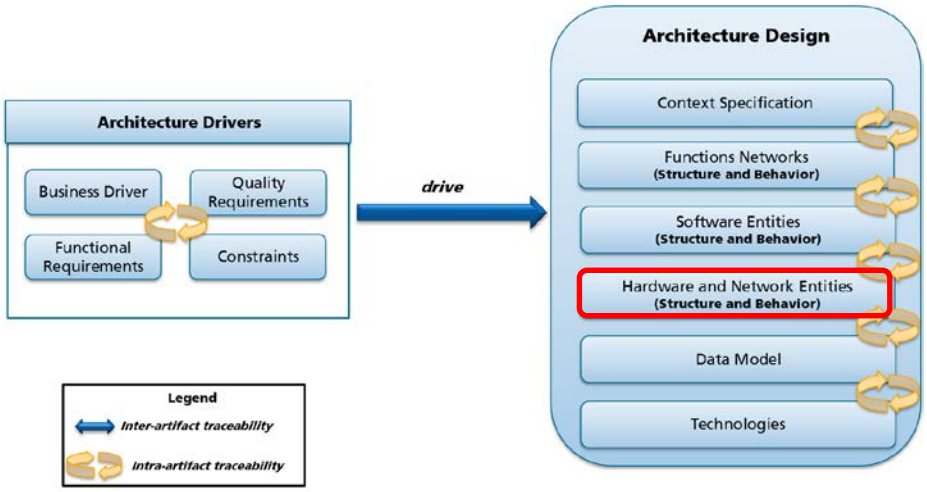


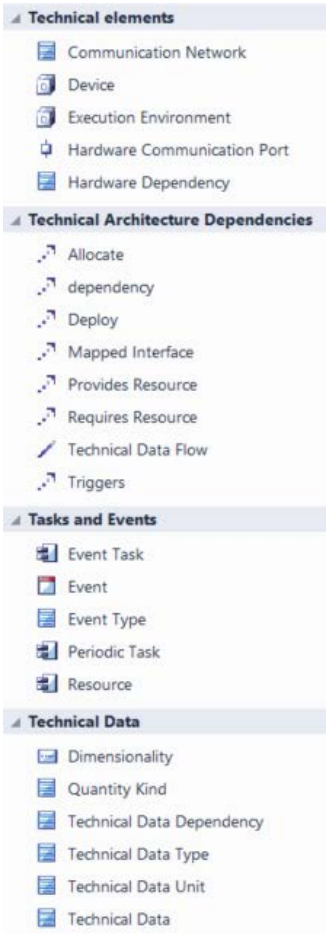




Fraunhofer IESE

# Hardware and Network Entities



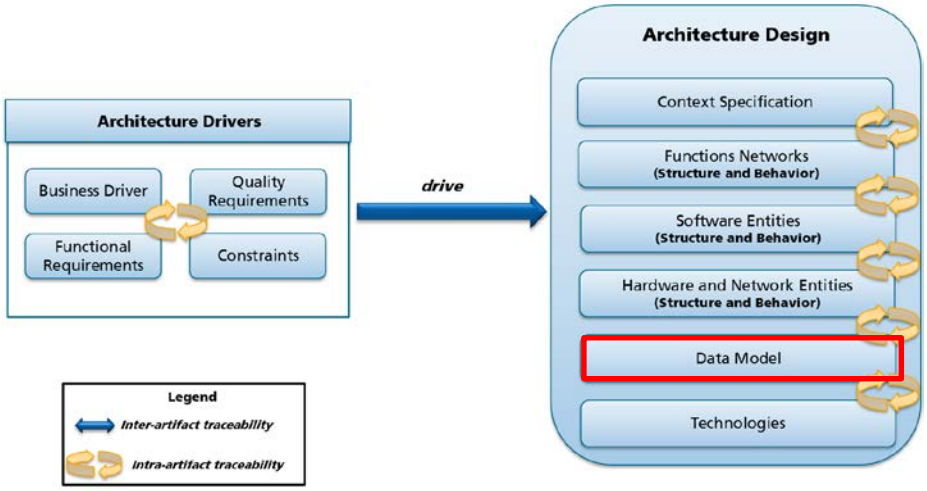



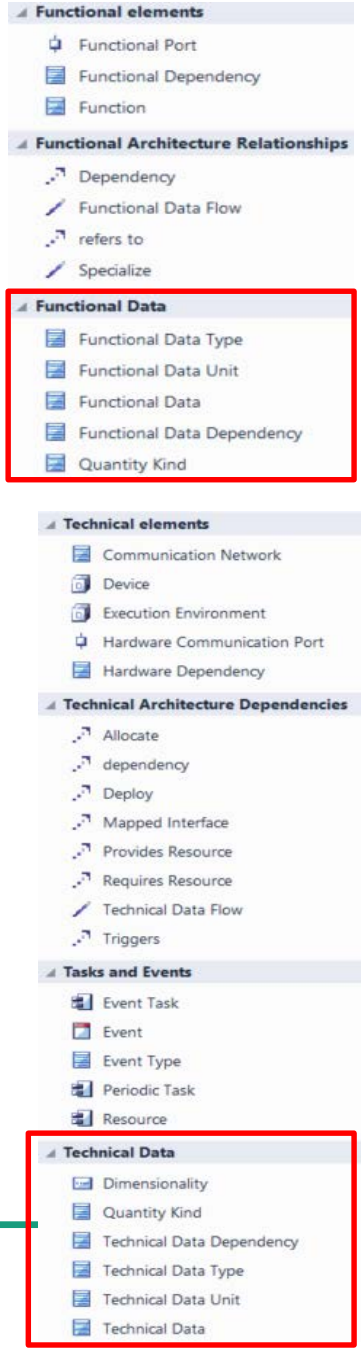





Fraunhofer IESE

# Data Model



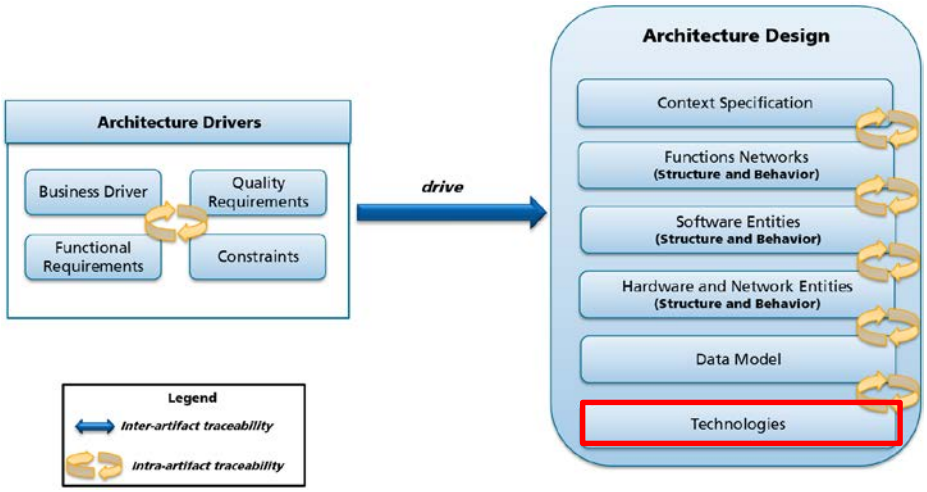






Fraunhofer IESE

# Data Model




- **C Realization**
  - C Contained Stereotype
  - C Function
  - C Global Variable
  - C Structure
  - C Type
  - Interrupt Handler
  - Memory Mapped Register
  - Translation Unit
- **C Realization Dependencies**
  - Invokes
  - Provides Interface
  - Reads Variable
  - Realizes Component
  - Requires Interface
  - Writes Variable
- **C++ Realization**
  - C++ Attribute
  - C++ Class
  - C++ Operation
  - C++ Structure
  - C++ Visibility Element
  - Namespace
- **C++ Relationships**
  - C++ Specialization
- **Simulink Realization**
  - Simulink Block
  - Simulink Port
- **Simulink Dependencies**
  - Provides Interface
  - Realizes Component
  - Requires Interface
  - Simulink Connection





Fraunhofer IESE

# Architecture and Safety

# What is so special about safety?


# GET THE FACTS

- "For the 34 (safety) incidents analyzed, 44% had inadequate specification as their primary cause."

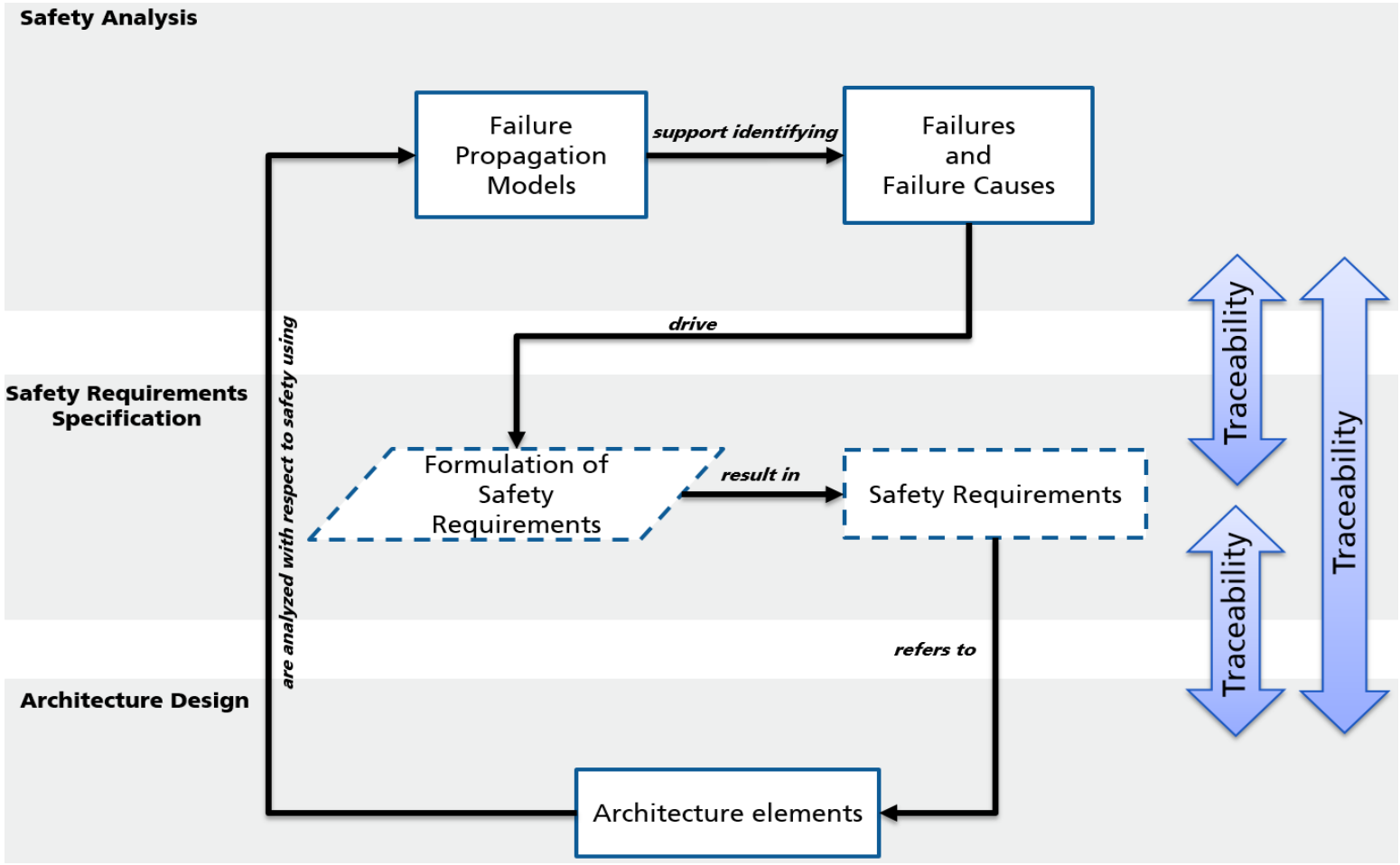
  *Out of Control: Why Control Systems Go Wrong and How to Prevent Failure.*

  Health and Safety Executive (HSE), 2015.

- "Almost all accidents related to software components in the past 20 years can be traced to flaws in the requirements specifications, such as unhandled cases."

  Safety-Critical Requirements Specification and Analysis using SpecTRM.

  Safeware Engineering, 2014.

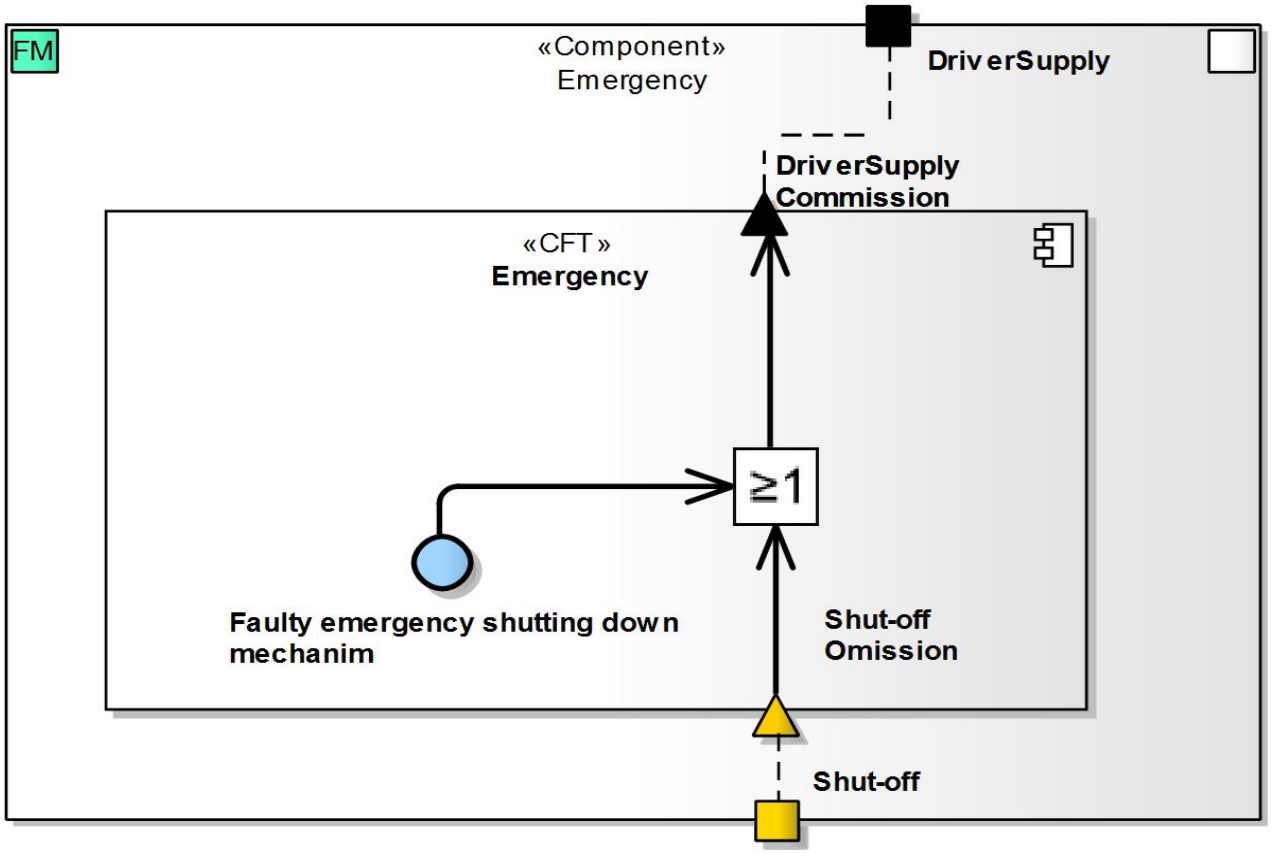




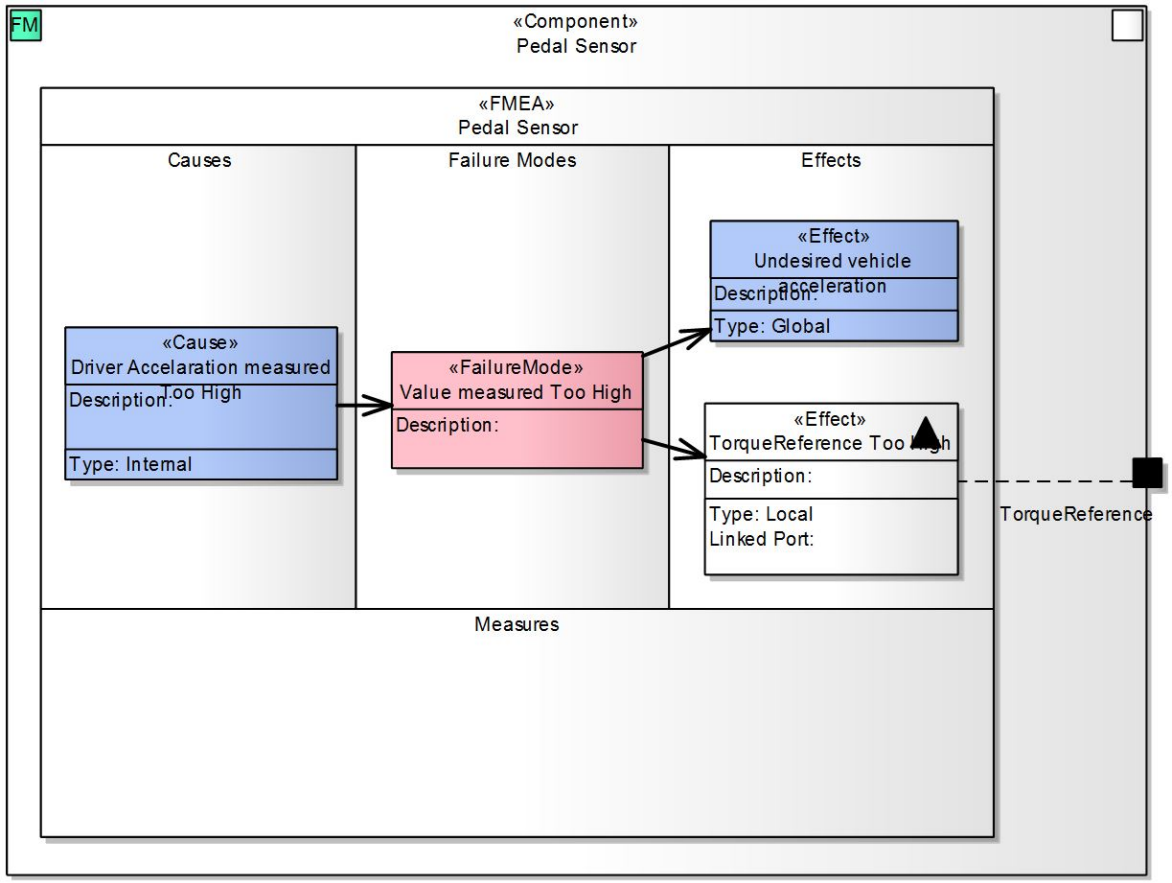



Fraunhofer IESE

# Component Fault Trees - CFTs



# Failure Modes and Effect Analysis - FMEA

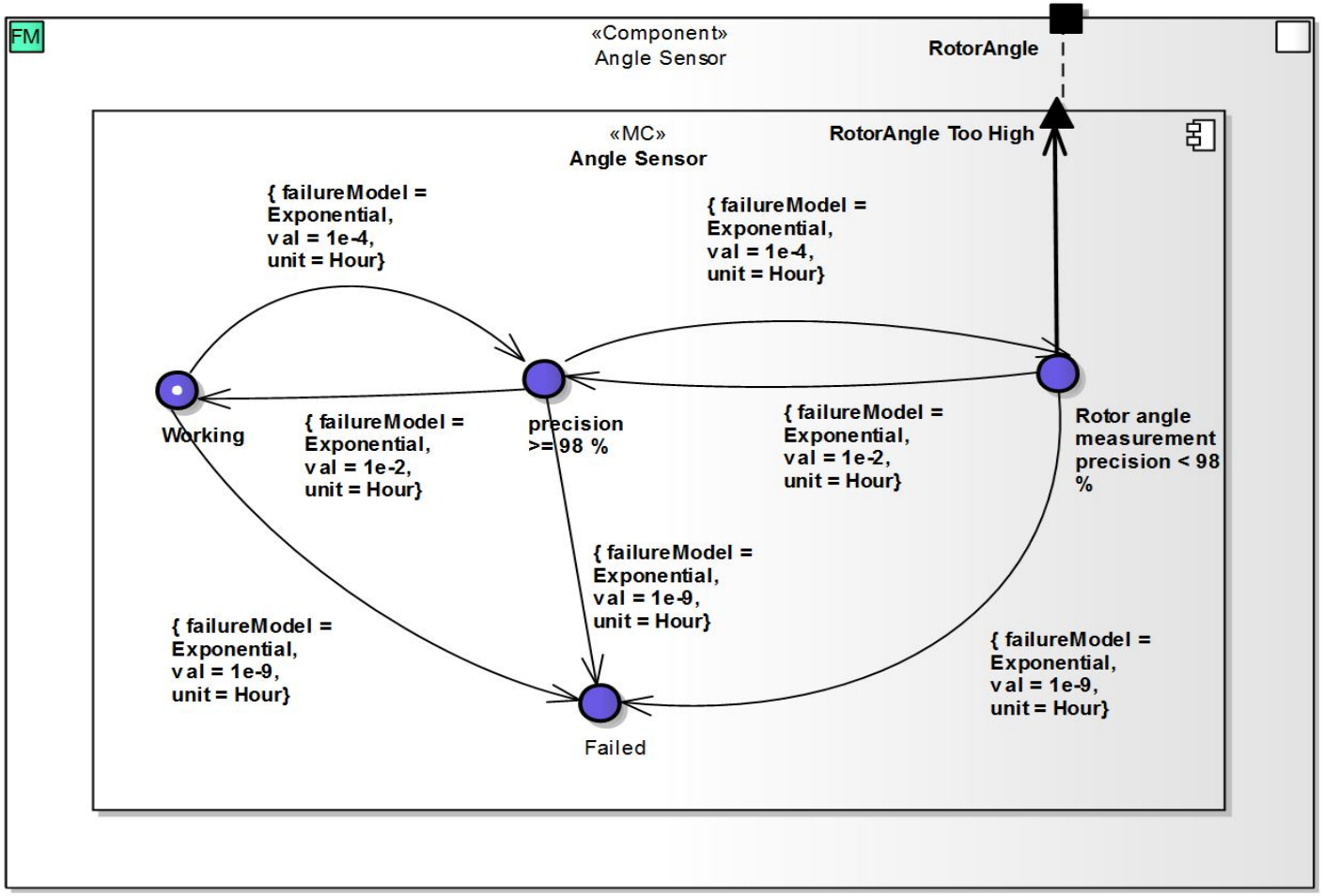


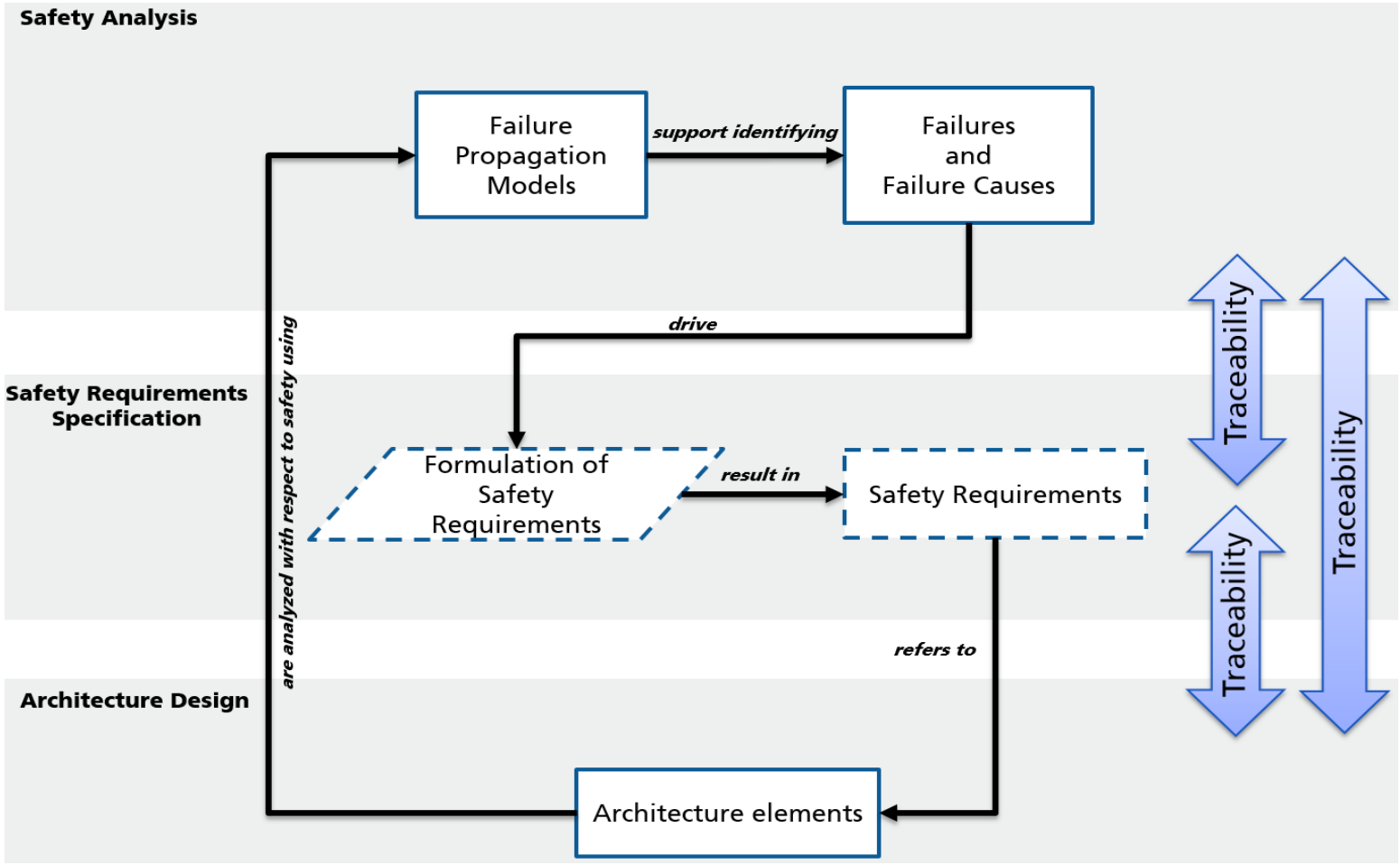





Fraunhofer IESE

# Markov Chains










Fraunhofer IESE




- **IEC 61508 –** *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*;
- **ISO 26262 –** Road vehicles -- Functional safety;
- **IEC 62061, ISO 13849, ISO 15998** (Earth-moving Machinary)**, ISO 25119** (Agriculture Vehicles) **–** Machinery Safety;
- **EN 50126/8/9 –** Railway;
- **DO-254, DO-178C, ARP 4754, ARP 4761** – Aerospace.

# GET THE FACTS

- Traceability among hazards, safety requirements, and architecture of equipments submitted to FDA are usually incomplete, incorrect, and conflicting.

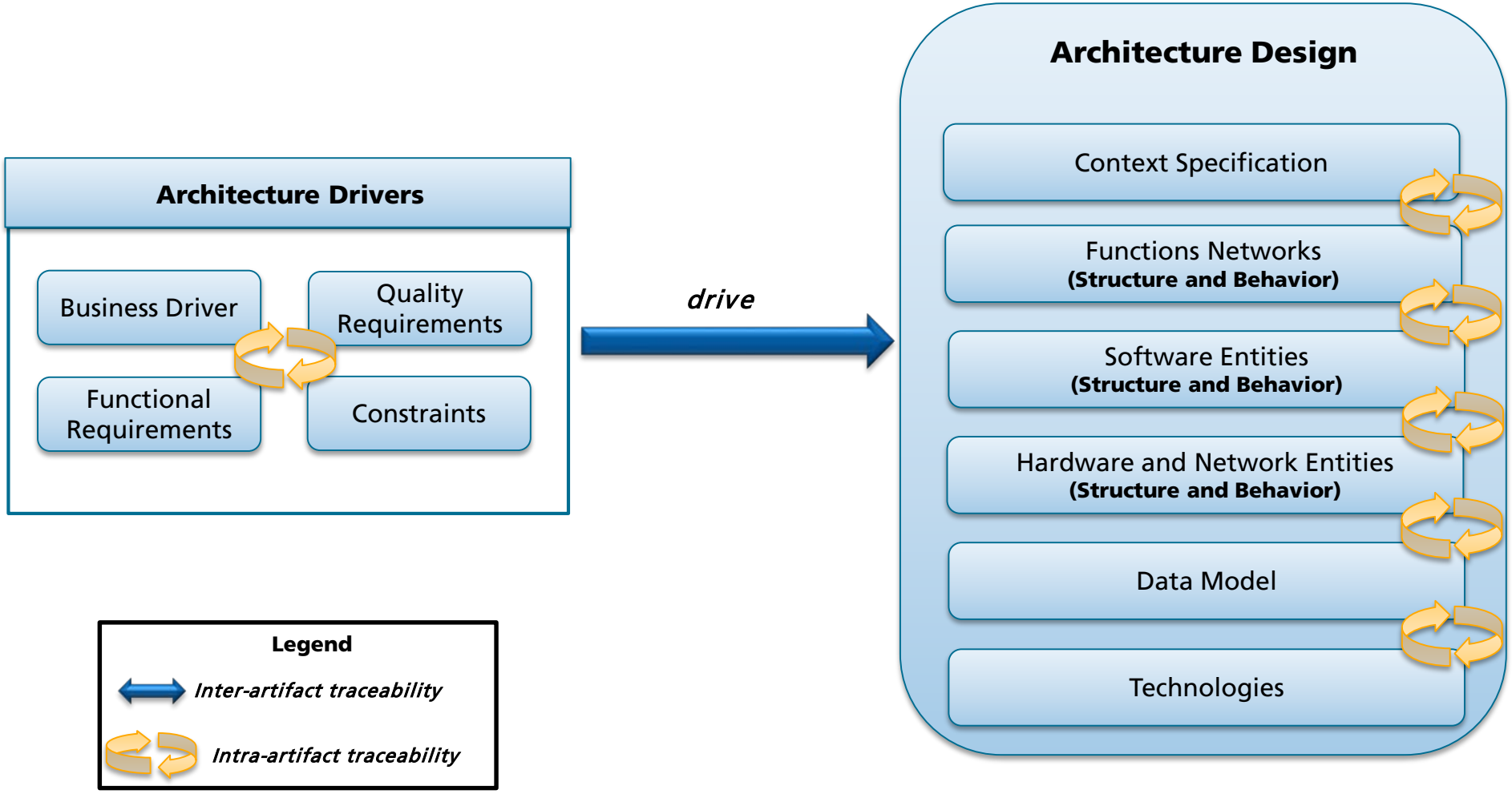
  FDA, 2014.

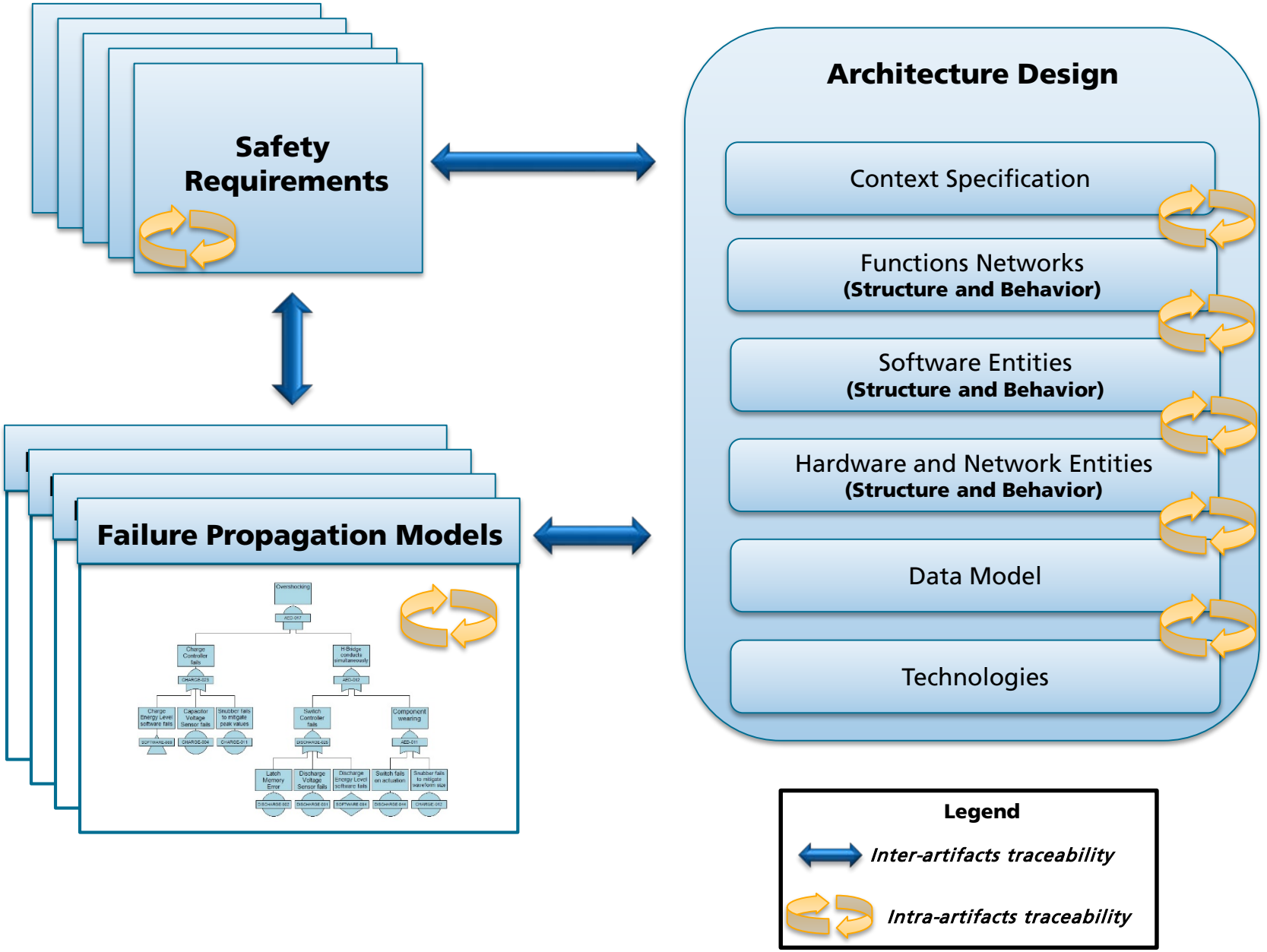
- Creating and documenting traceability immediately prior to certification is a common proceeding.

  Mäder et al., 2014.

- “None of the existing traceability approaches described in the literature are appropriate to meet this demand of the safety-critical domain .”

  CoEST - Center of Excellence for Software Traceability, 2012.

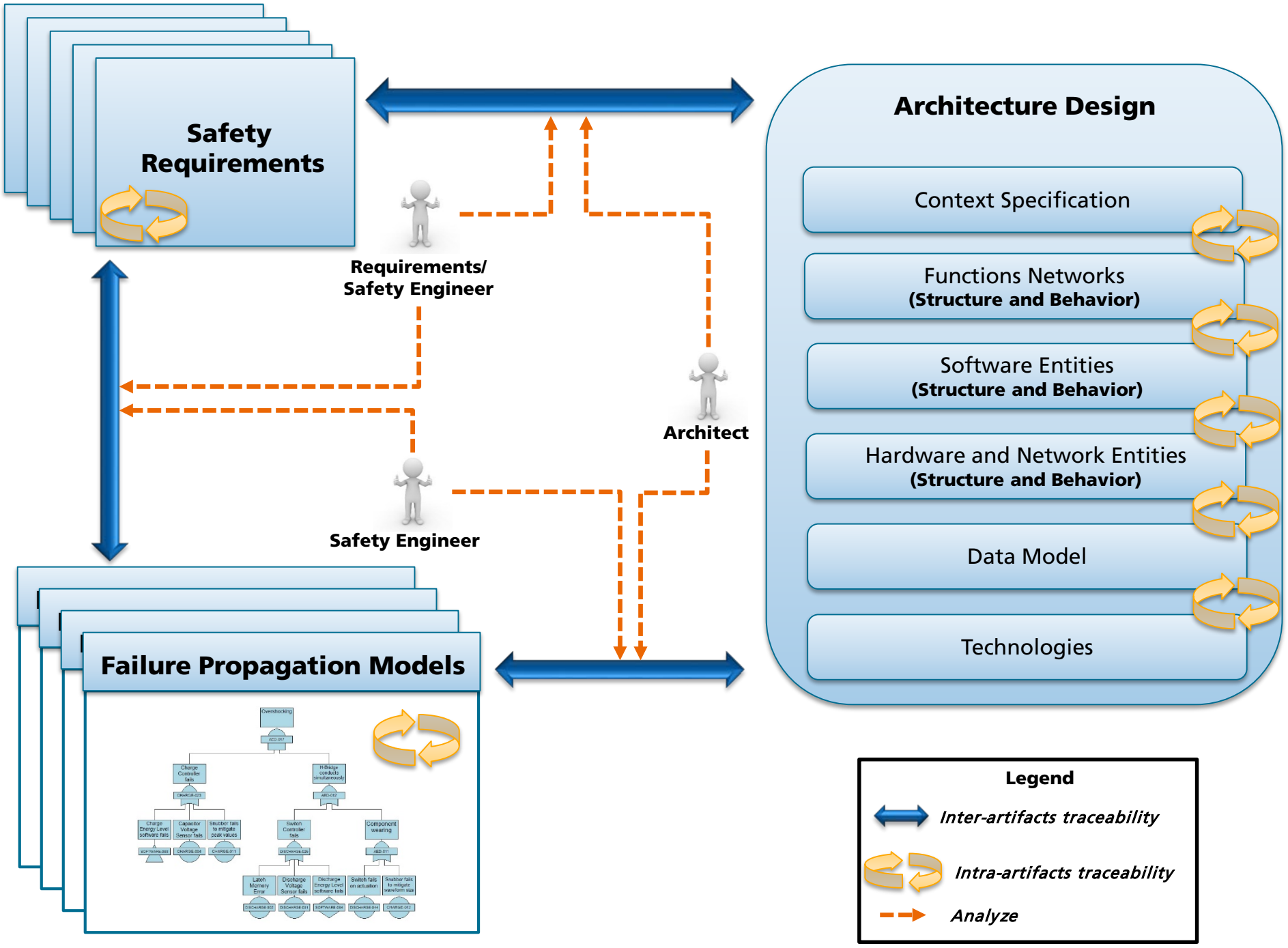




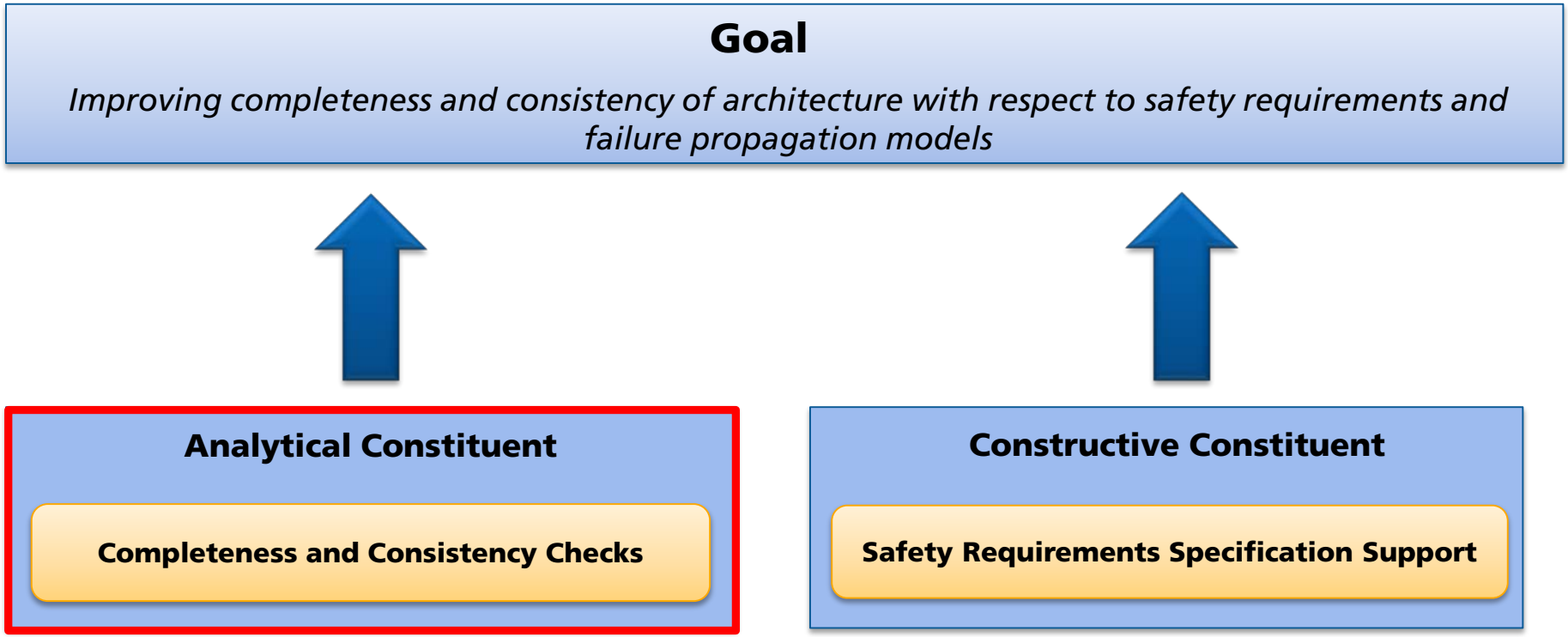





Fraunhofer IESE


Safety Requirements
Requirements/ Safety Engineer
Architect
Safety Engineer
Failure Propagation Models
Architecture Design
Context Specification
Functions Networks (Structure and Behavior)
Software Entities (Structure and Behavior)
Hardware and Network Entities (Structure and Behavior)
Data Model
Technologies
Legend
Inter-artifacts traceability
Intra-artifacts traceability
Analyze

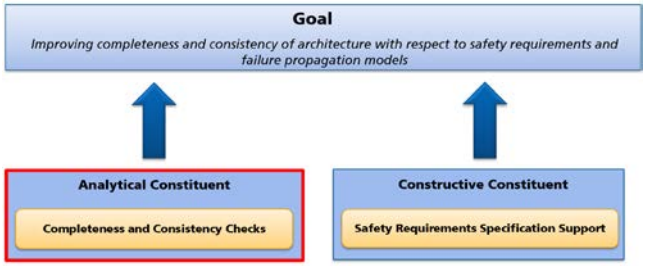
# Fraunhofer IESE Approach to deal with Safety Architectures



# Designing the Automated Completeness and Consistency Checks

# Meet Safety Engineering Goals

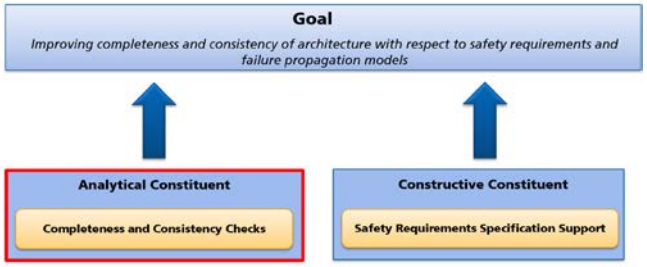
- All failures described in the failure propagation models are covered by safety requirements;
- All safety-related requirements are satisfied by elements of the architecture;
- Determine the potential impact of changing a requirement on its associated safety-related artifacts.



# Automation and Instantiation by Different Technology Platforms

- Non-automated approaches to dealing with large-scale software are unpractical and unrealistic to be considered in industrial software development environments.

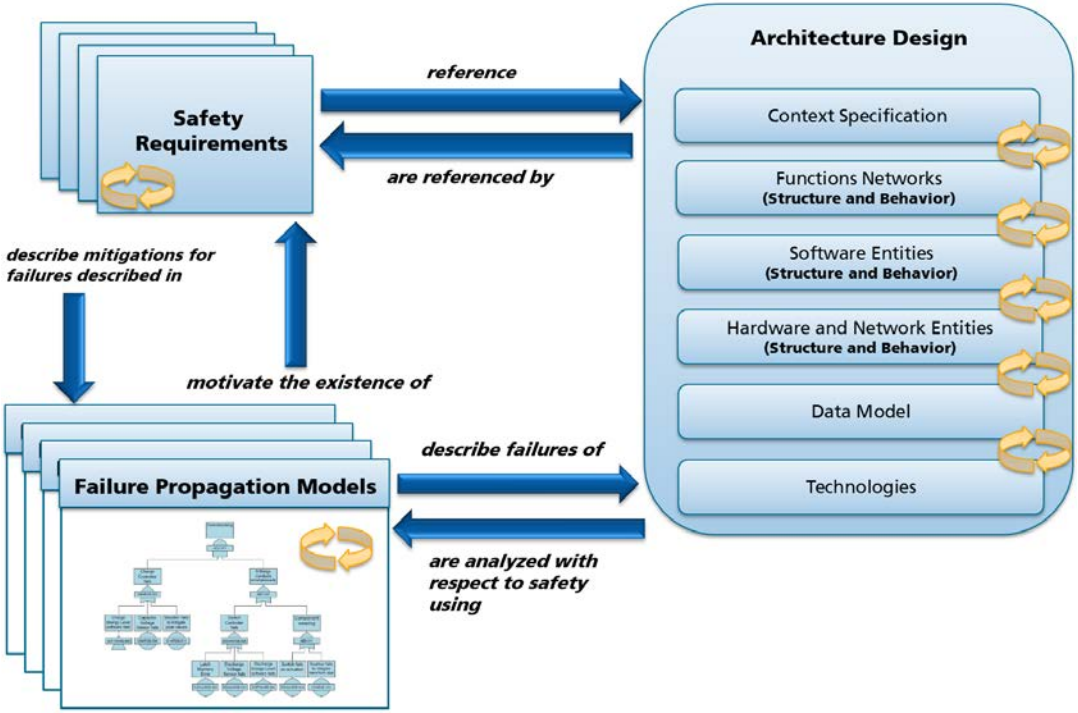
- Basis for implementation with (i) formal proofs, *(ii)* model checking, *(iii)* query languages, and *(iv)* specialists computer programs. .



# Completeness Checks

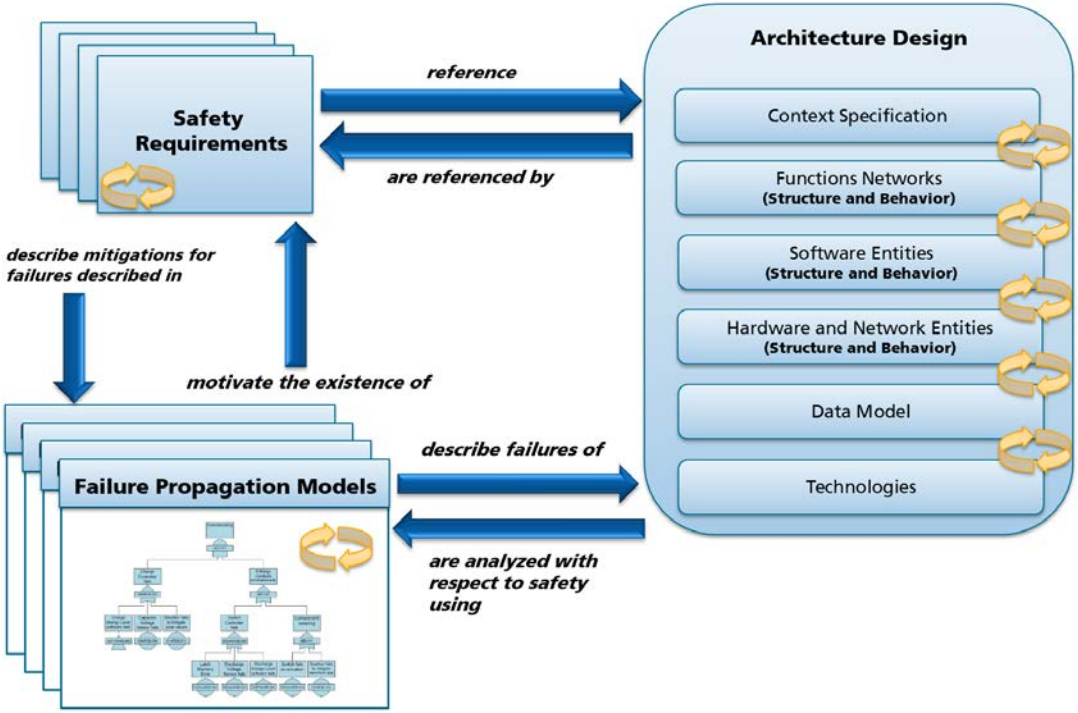
# Notion of Completeness

- Completeness is a quality attribute that is ensured when the definition and justification of a problem is found within the specification.



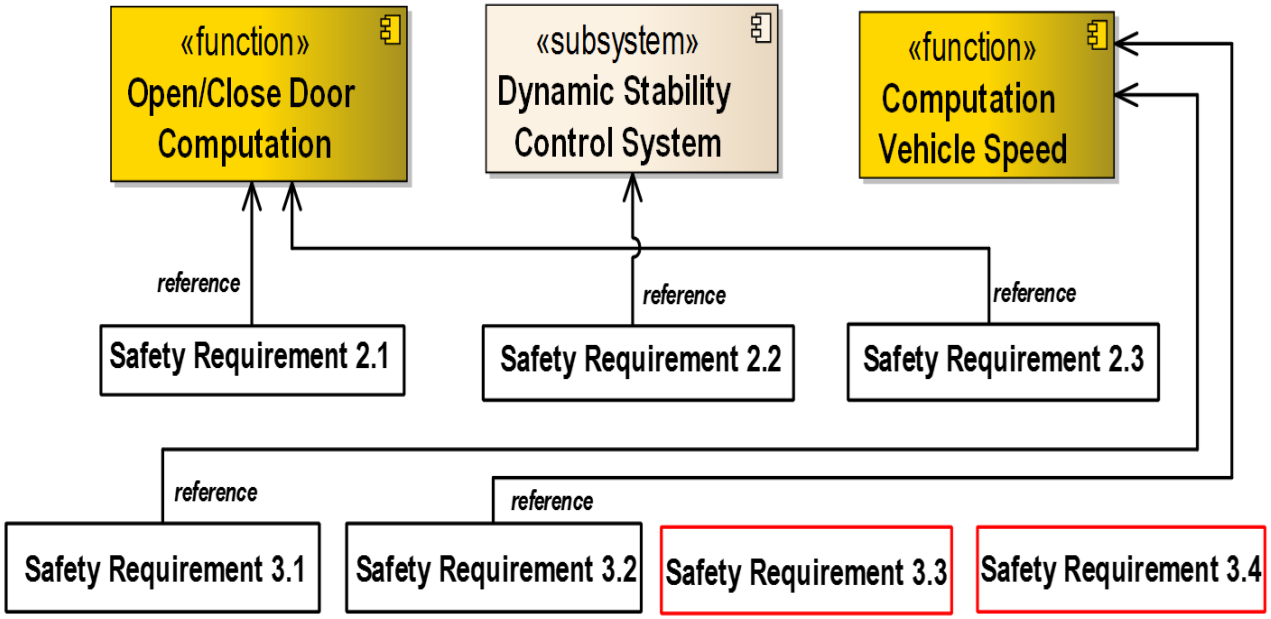
# Notion of Completeness

- Completeness is a quality attribute that is ensured when the definition and justification of a problem is found within the specification.



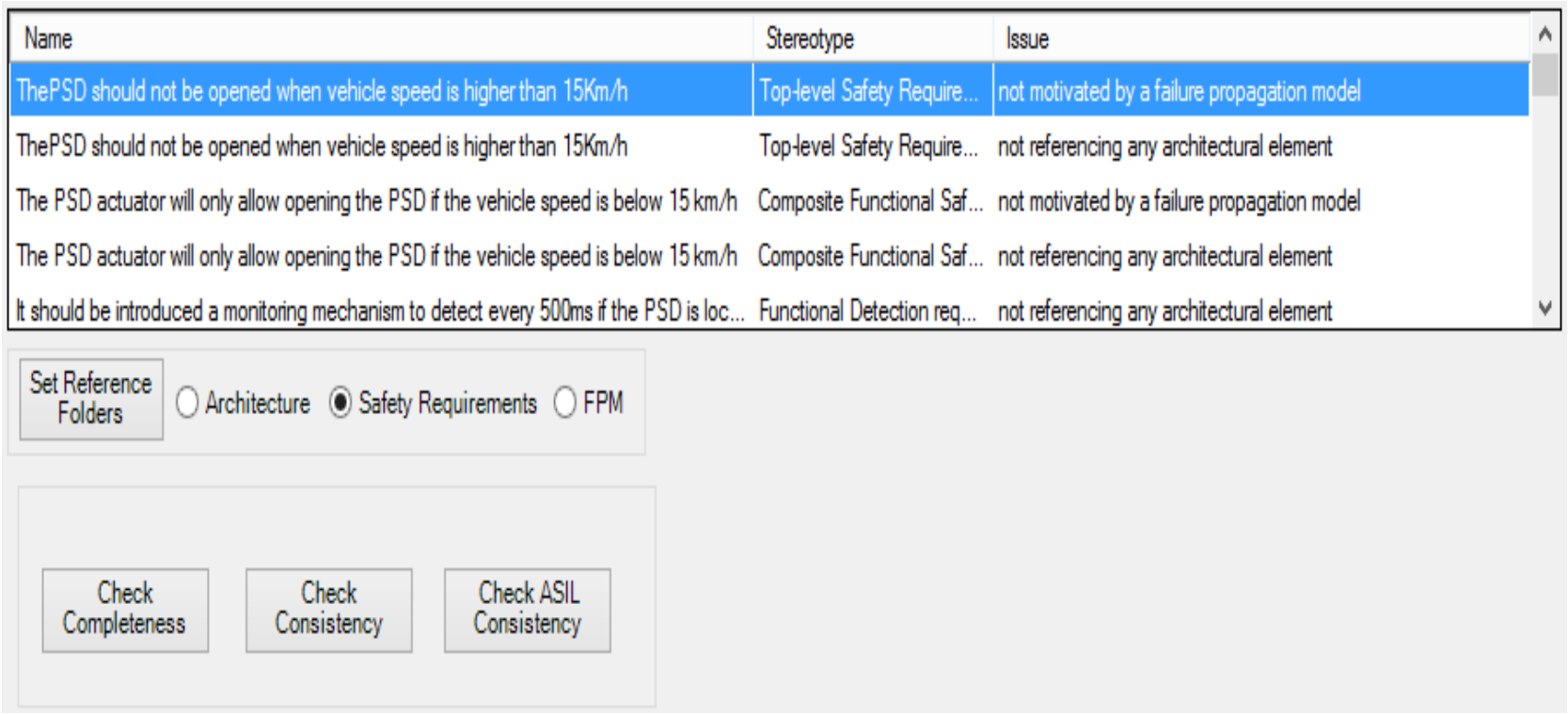

- **SRCompC3:** Every safety requirement describes failures mitigations referencing, at least, one safety-critical architecture element.
- **TransSRCompC3:** Every safety-critical architecture element addresses, at least, one safety requirement.

# Example Completeness Check




**Violation of the SRCompC3:** Every safety requirement describes failures mitigations referencing, at least, one safety-critical architecture element.

# I-SafE Completeness Checks Output Example

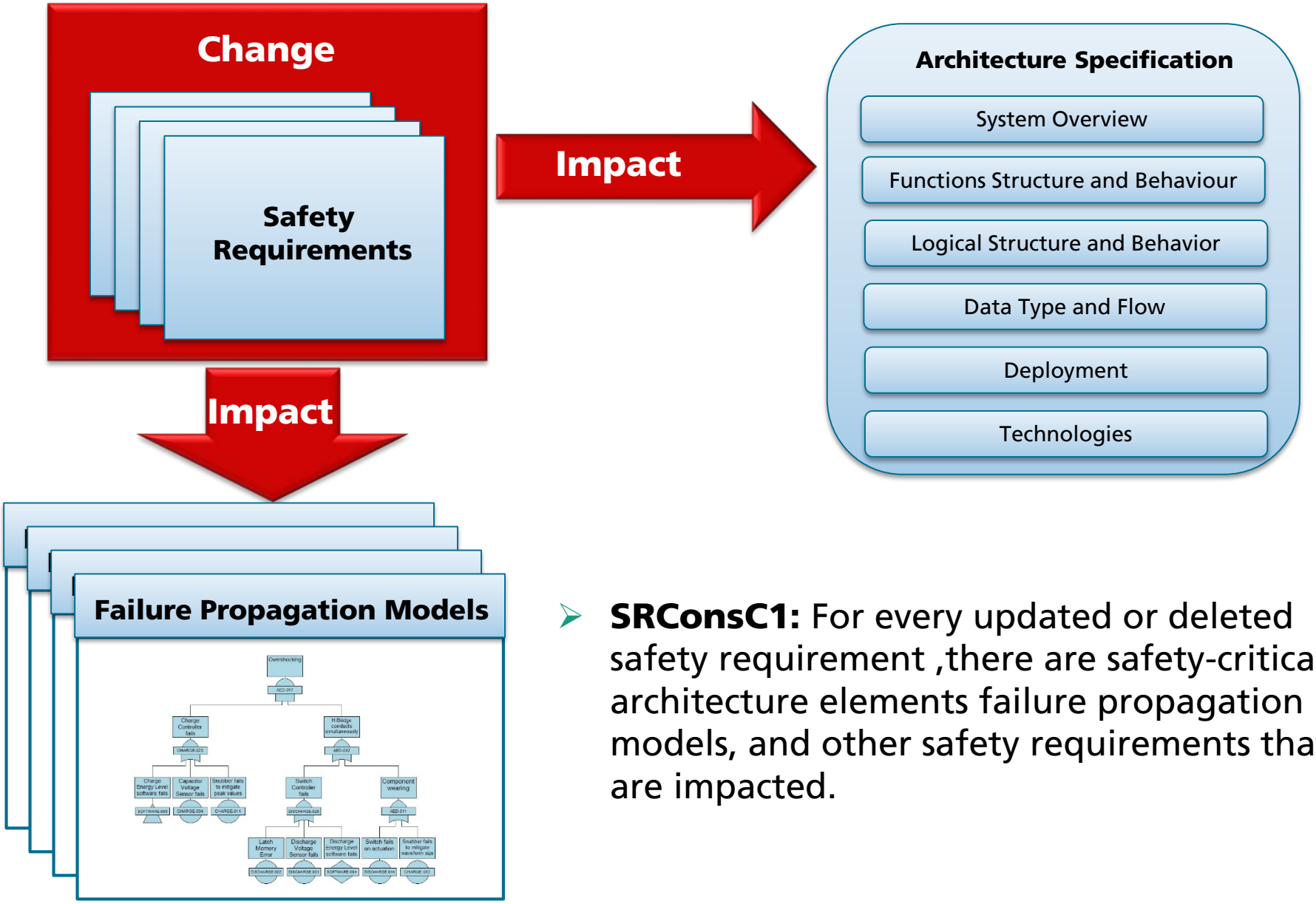


# Cosistency Checks

# Notion of Consistency

- Consistency is achieved when two or more artifacts obey relationships that should exist between them.

- A safety requirement is consistent as long as there are no contradictions among safety requirements, safety-critical architecture elements, and failure propagation models.

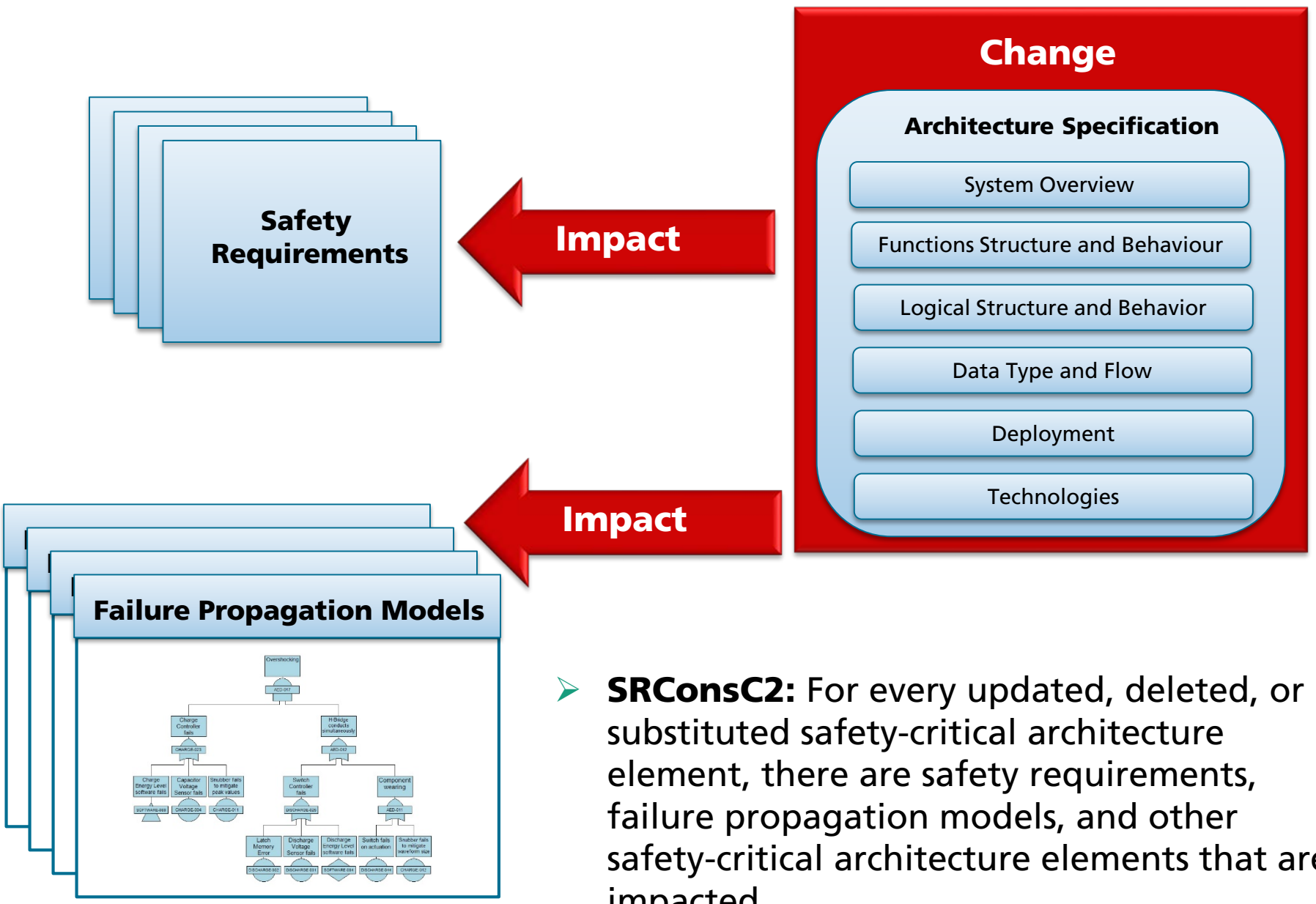



- **SRConsC1:** For every updated or deleted safety requirement ,there are safety-critical architecture elements failure propagation models, and other safety requirements that are impacted.

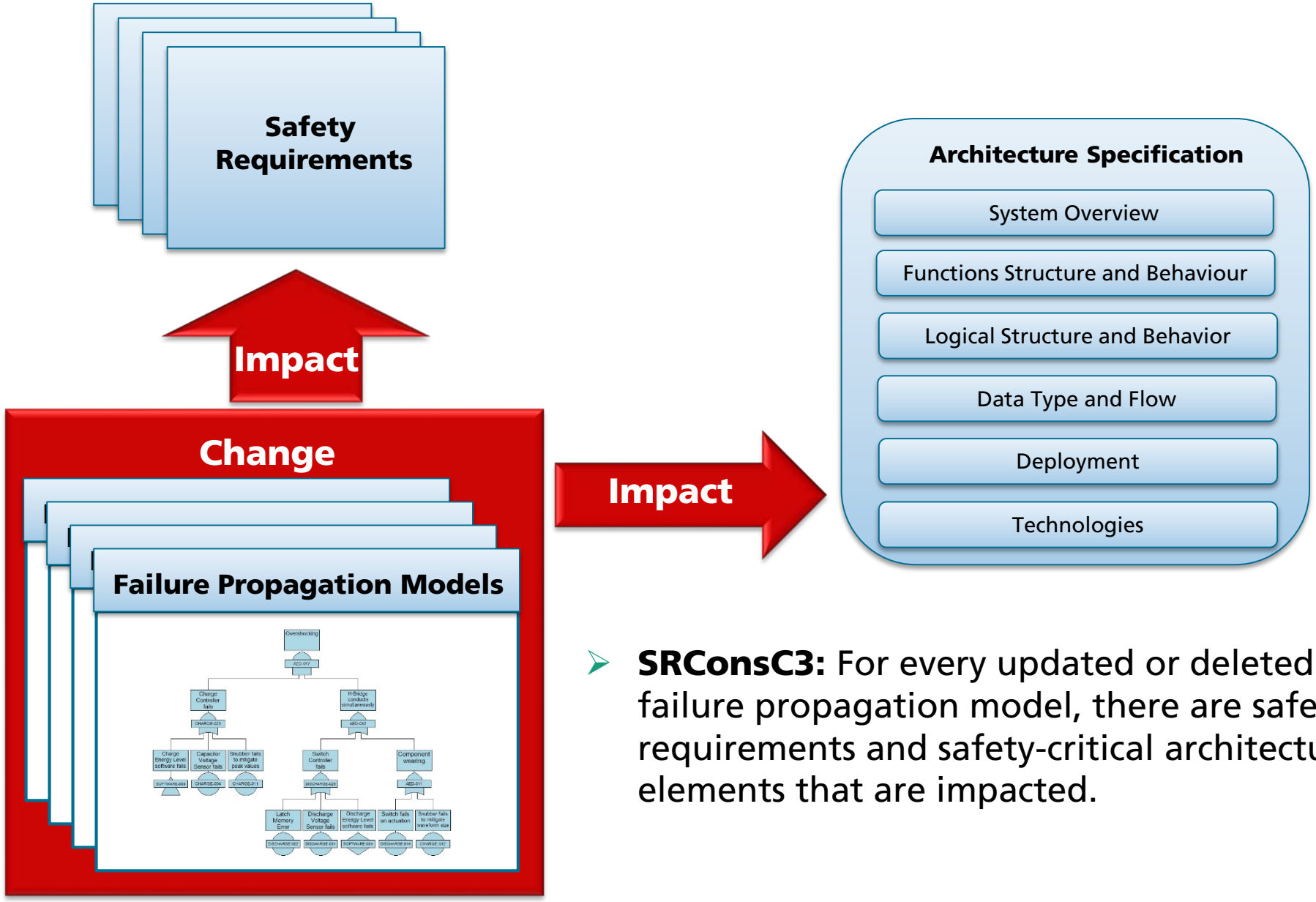




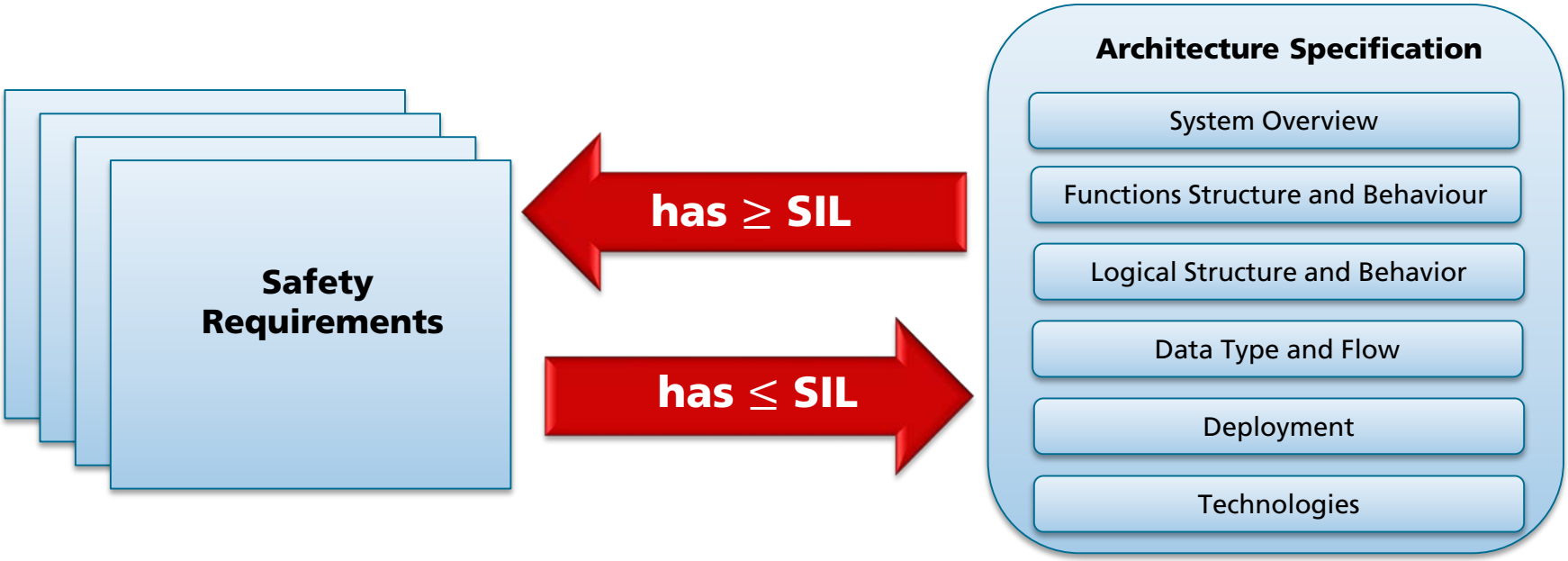
Fraunhofer IESE




- **SRConsC2:** For every updated, deleted, or substituted safety-critical architecture element, there are safety requirements, failure propagation models, and other safety-critical architecture elements that are impacted.

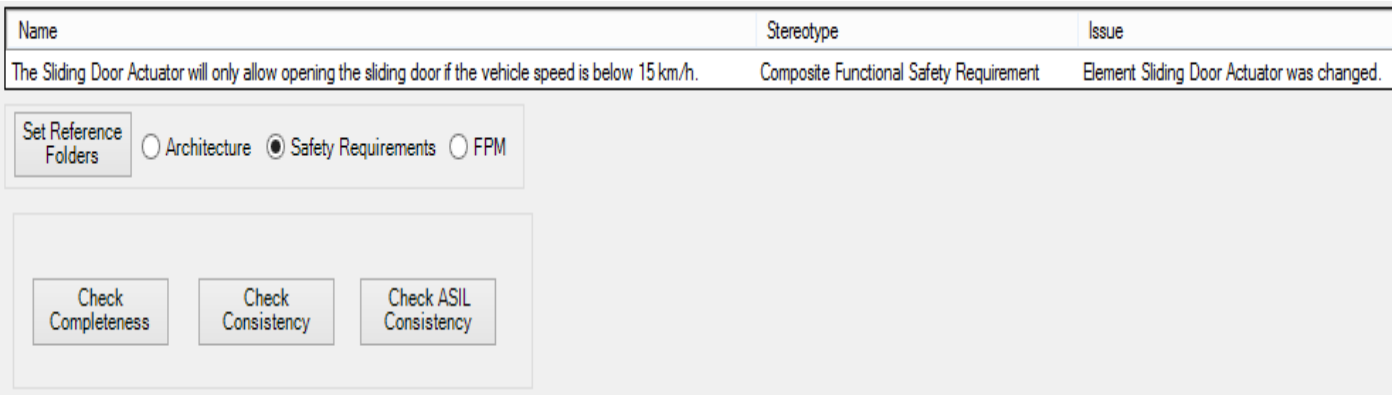


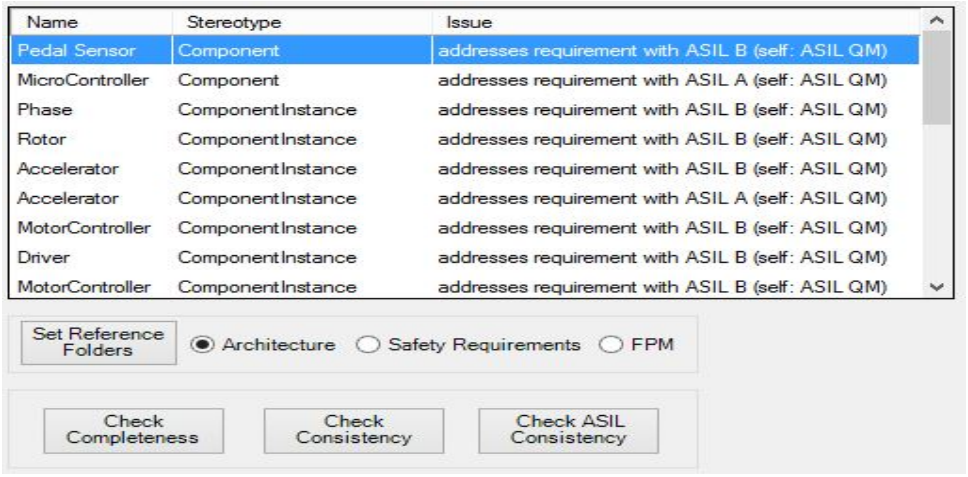

- **SRConsC3:** For every updated or deleted failure propagation model, there are safety requirements and safety-critical architecture elements that are impacted.



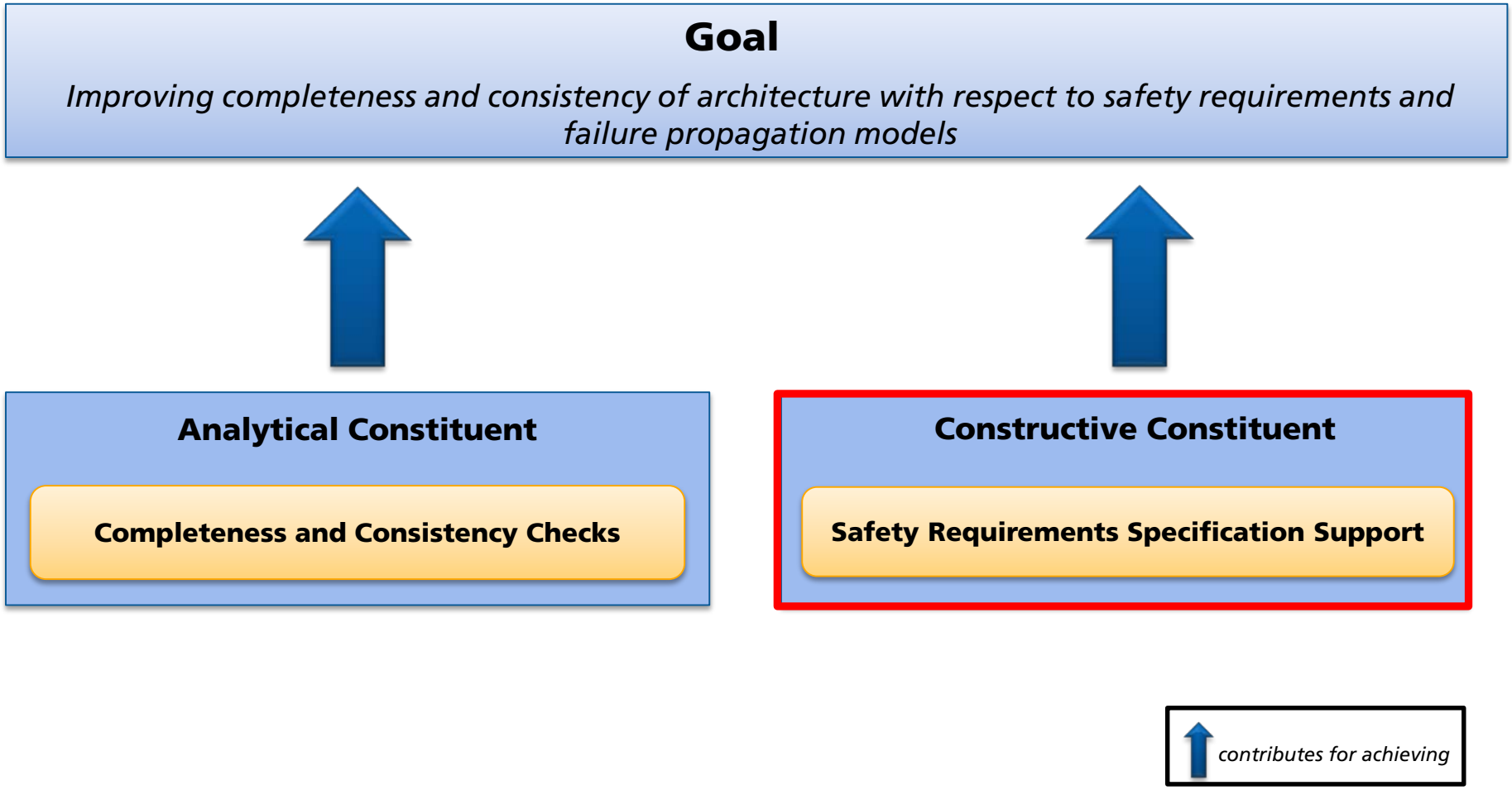

- **SRConsC4:** The safety requirements are addressed by safety-critical architecture elements with an equal or more stringent safety integrity level.
- **TransSRConsC4:** Safety-critical architecture elements address safety requirements that have an equal or less stringent safety integrity level.

# I-SafE Consistency Checks Output Examples

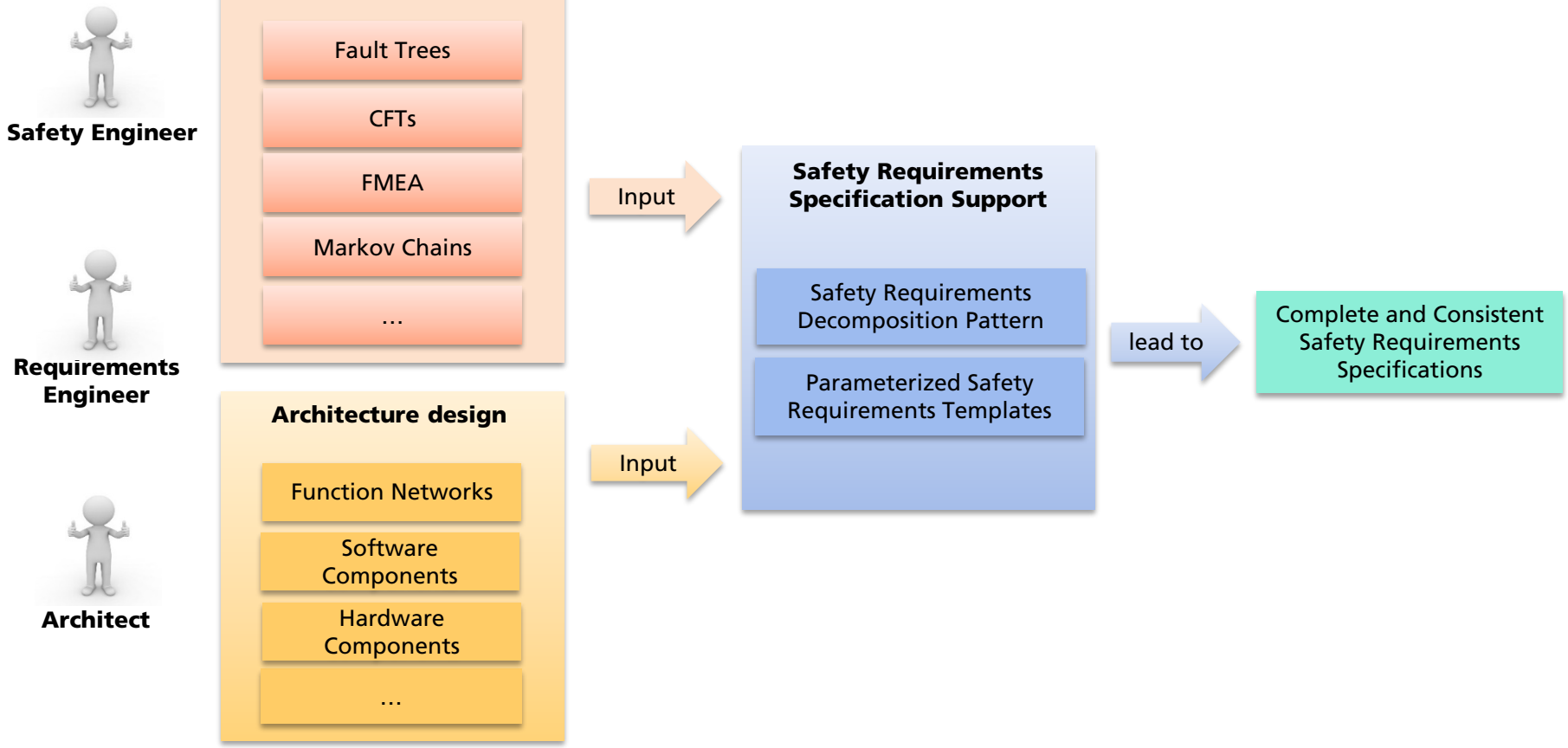


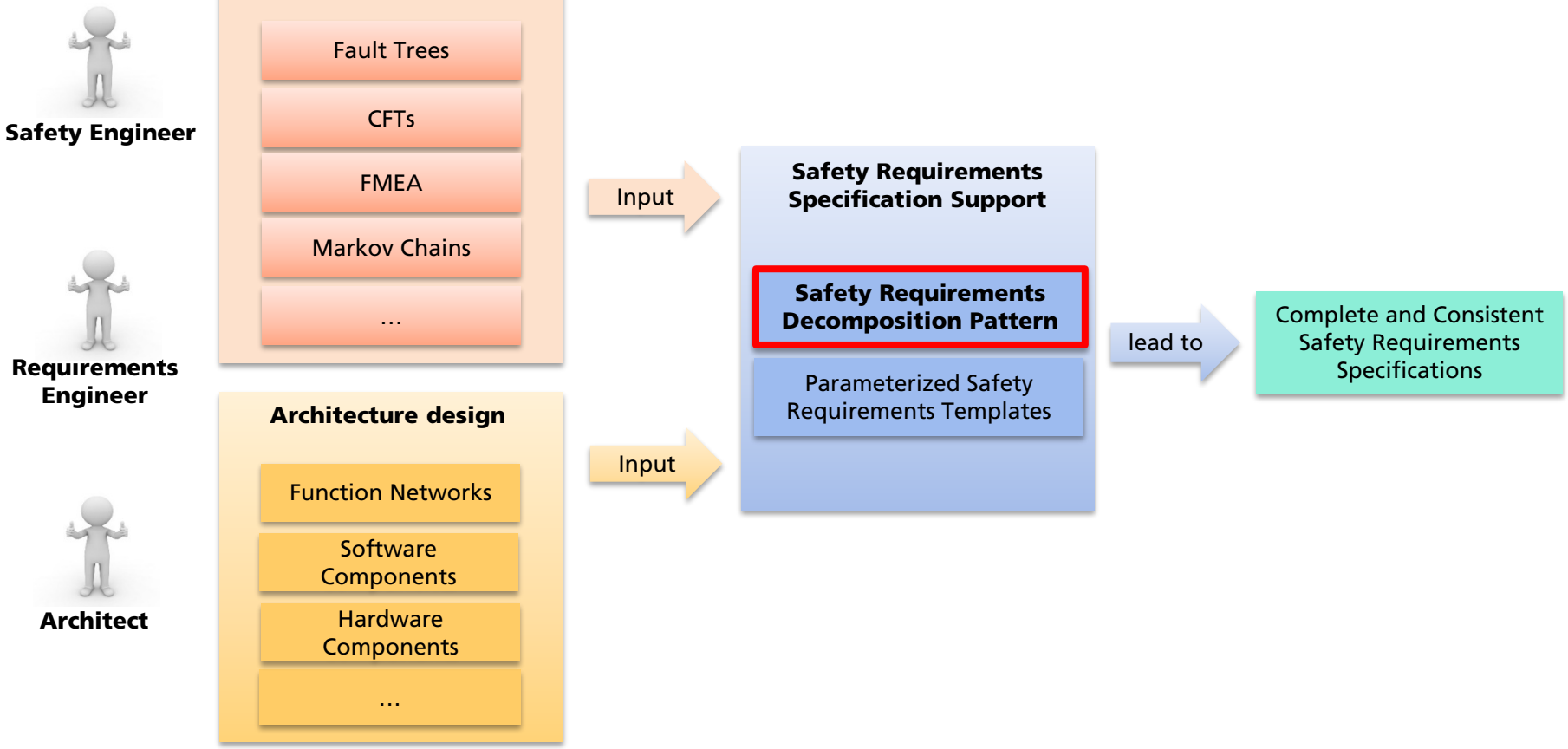
# Fraunhofer IESE Approach to deal with Safety Architectures



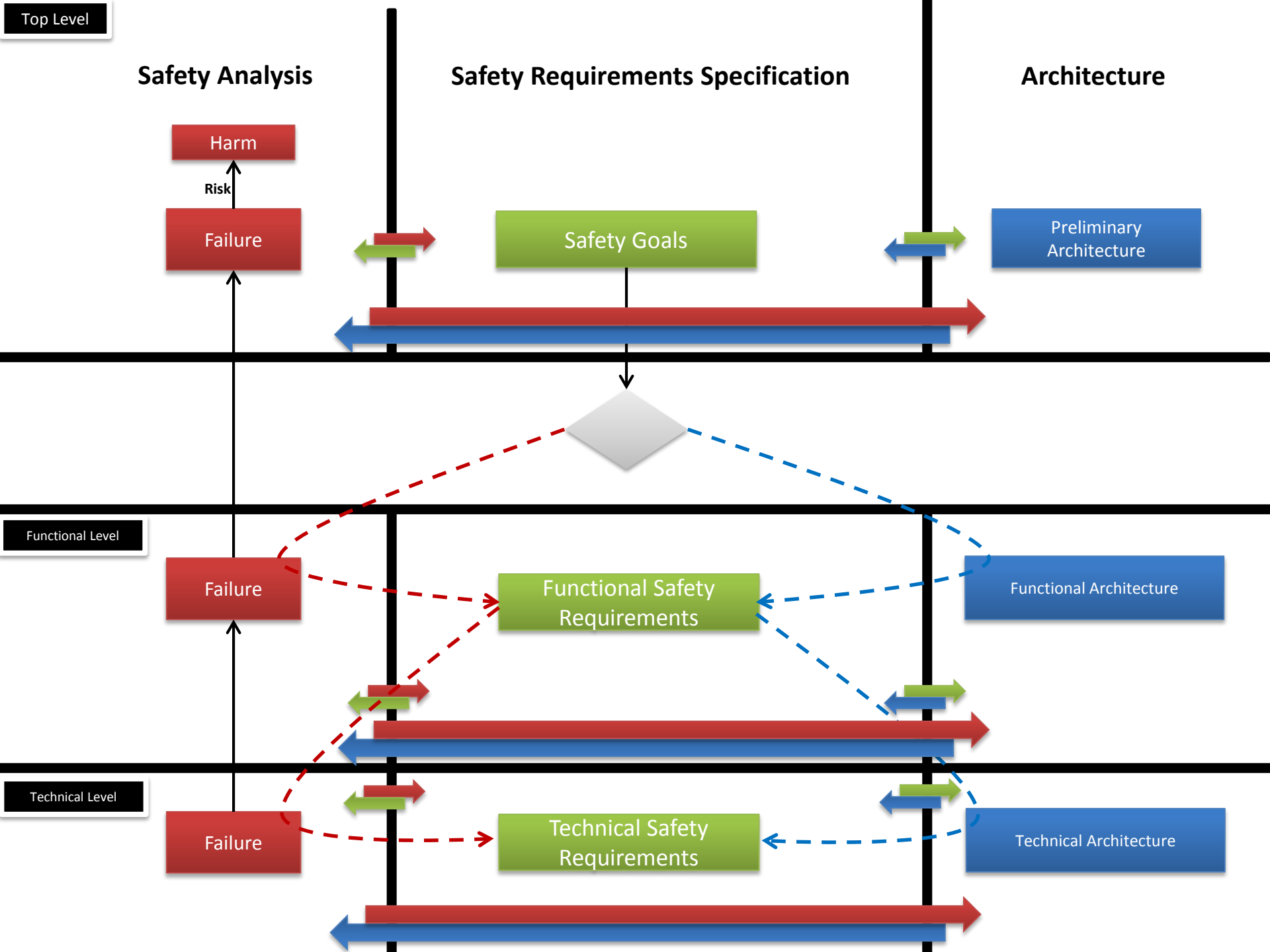
# Safety Requirements Specification Support



# Safety Requirements Specification Support

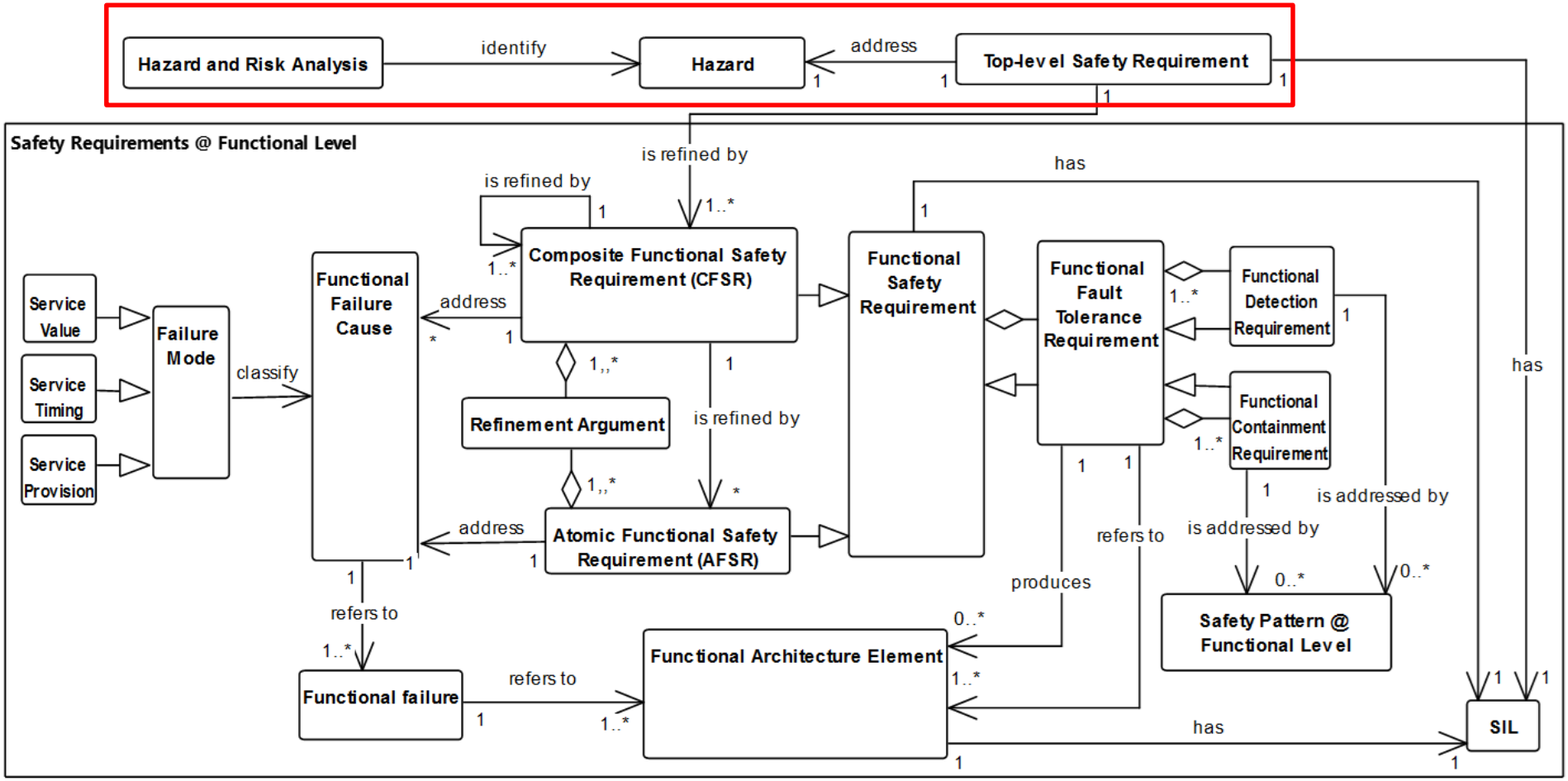


# The Safety Requirements Decomposition Pattern


Top Level
Safety Analysis
Safety Requirements Specification
Architecture
Harm
Risk
Failure
Safety Goals
Preliminary Architecture
Functional Level
Failure
Functional Safety Requirements
Functional Architecture
Technical Level
Failure
Technical Safety Requirements
Technical Architecture

# The Safety Requirements Decomposition Pattern @ the Functional Level

# Safety Requirements Decomposition Pattern @ Functional Level

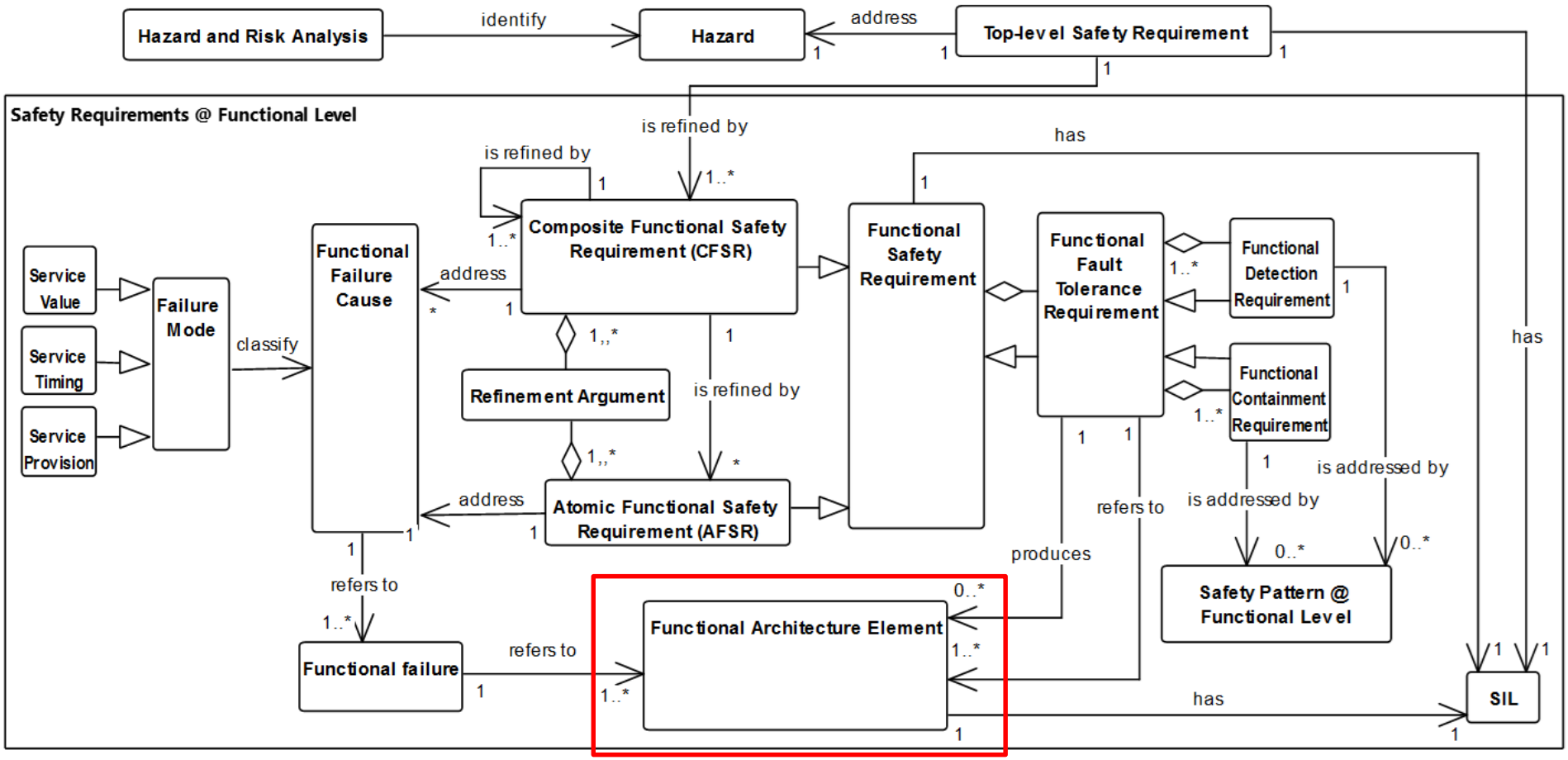







Fraunhofer IESE

# Safety Requirements Decomposition Pattern @ Functional Level

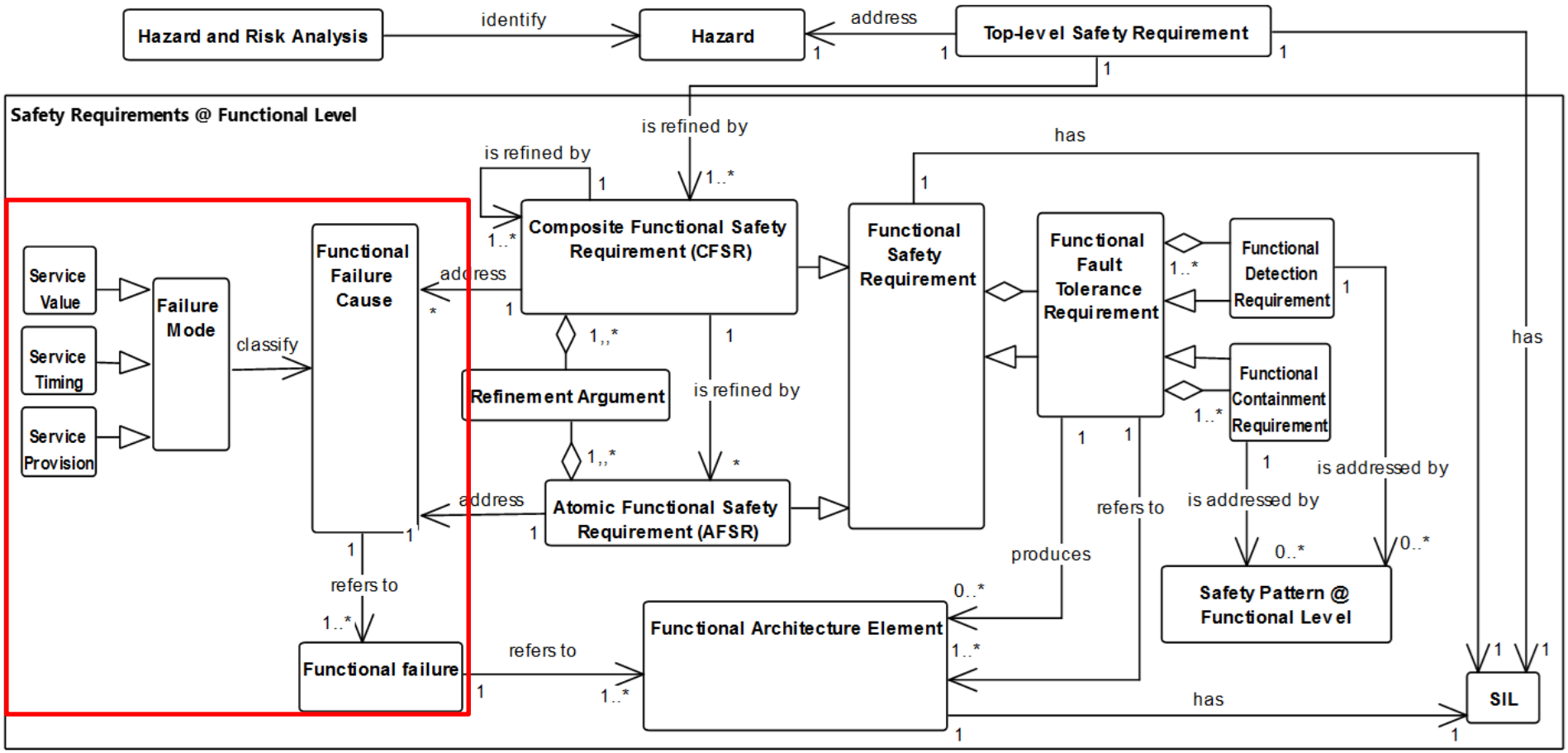







Fraunhofer IESE

# Safety Requirements Decomposition Pattern @ Functional Level

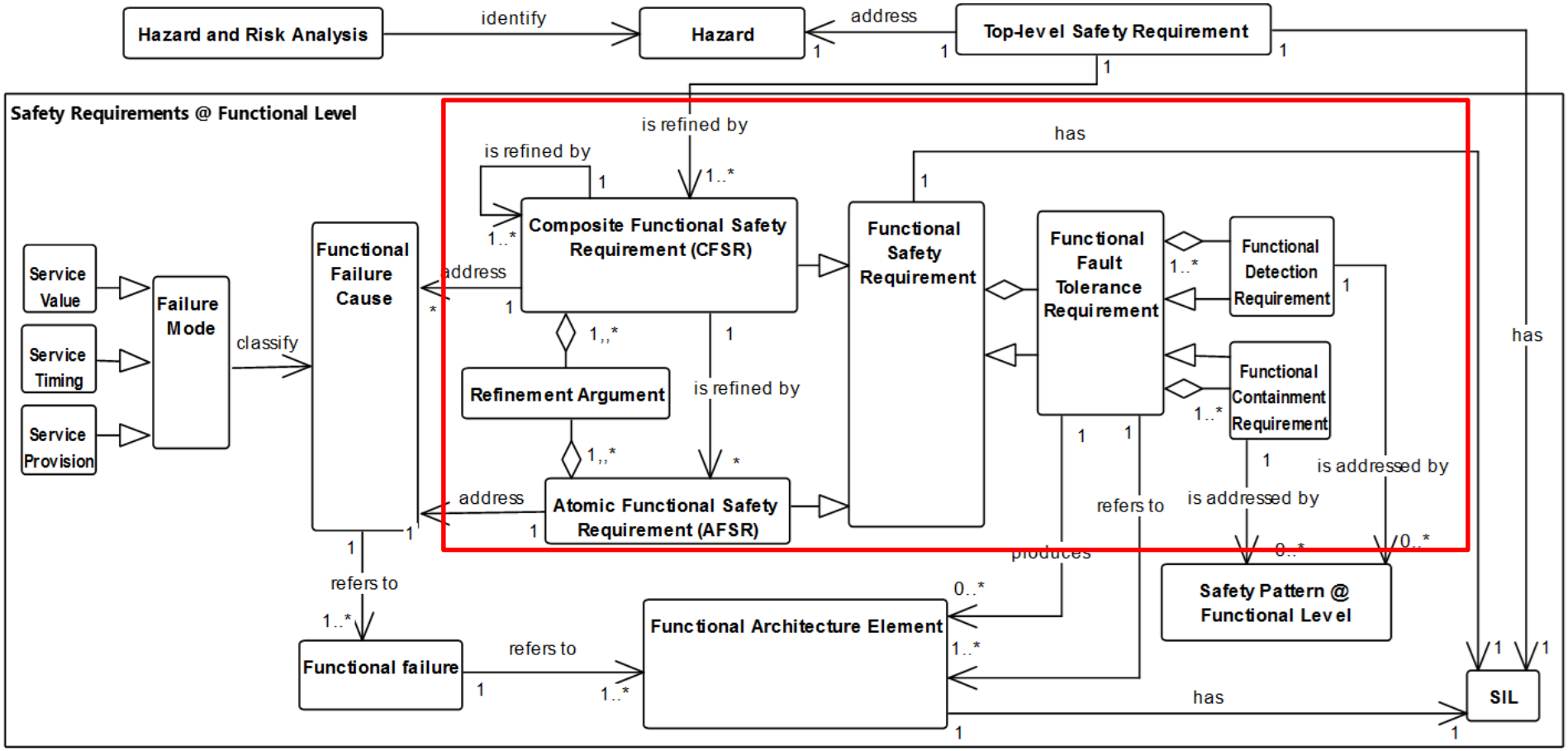







Fraunhofer IESE

# Safety Requirements Decomposition Pattern @ Functional Level



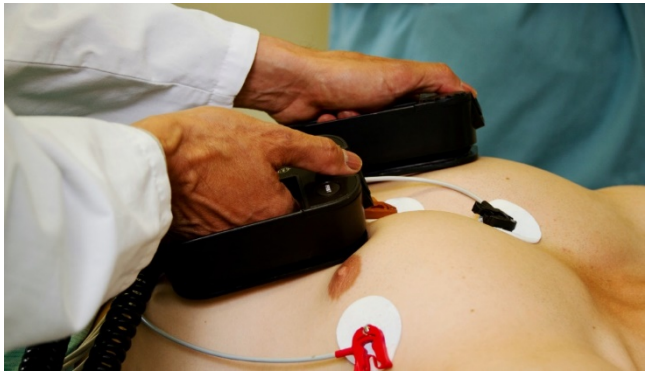





Fraunhofer IESE

# Automated External Defibrillator

## Traditional External Defibrillator

## Automated External Defibrillator (AED)



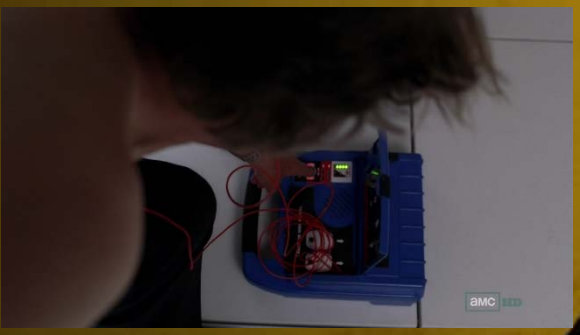



Fraunhofer IESE


AED
AUTOMATISIERTER EXTERNER DEFIBRILLATOR
amc HD


Mr. Schuler, open this door.
amc HD


amc HD

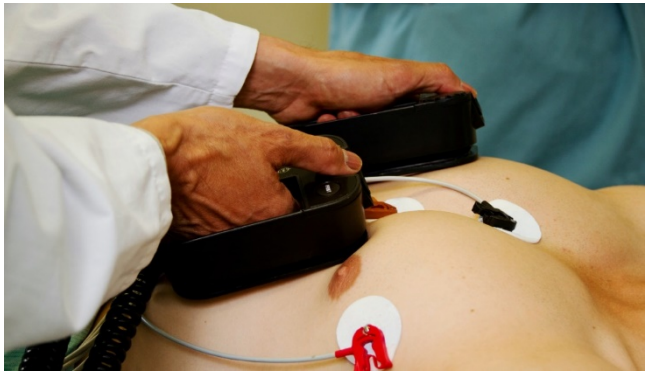

amc HD


amc HD

## Traditional External Defibrillator


## Automated External Defibrillator (AED)

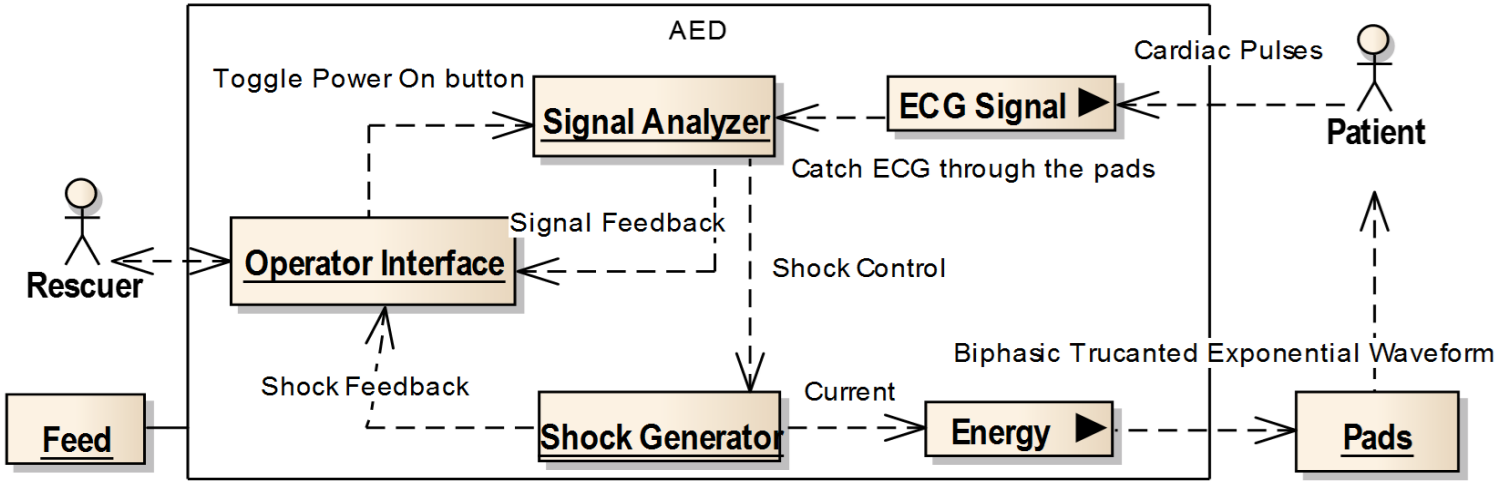

**Overshocking!!**







Fraunhofer IESE



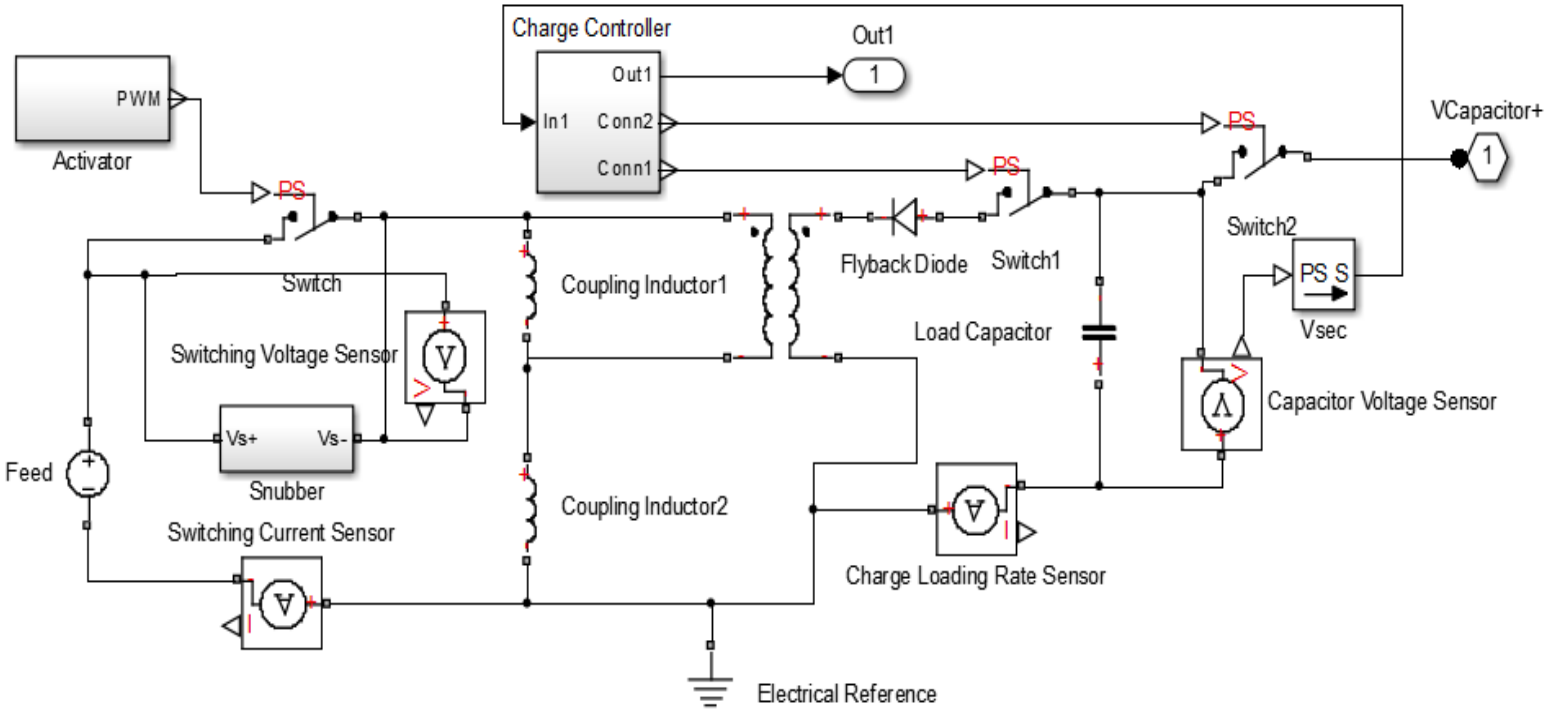

http://nutes.uepb.edu.br/

http://www.lifemed.com.br





Fraunhofer IESE



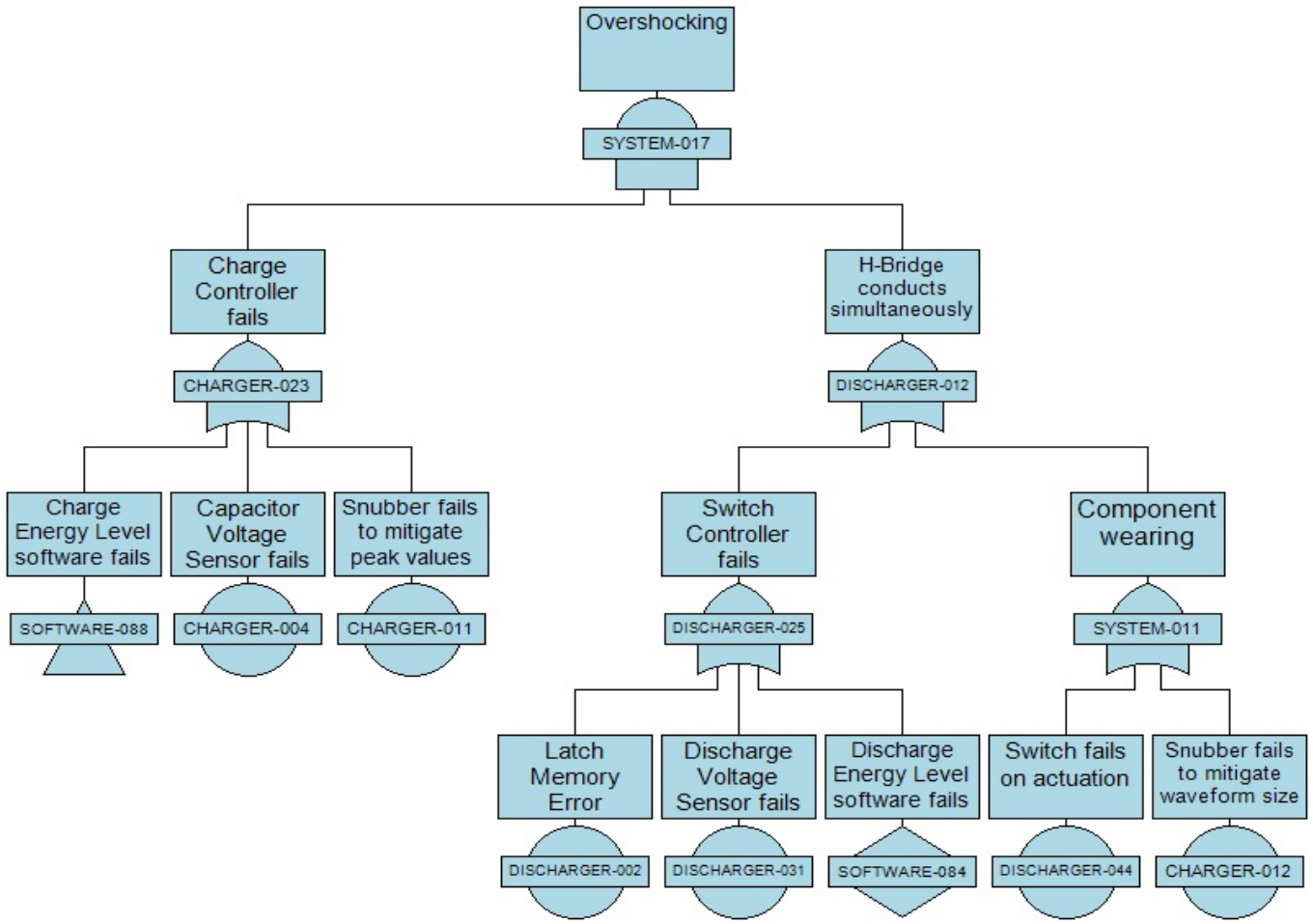

http://nutes.uepb.edu.br/

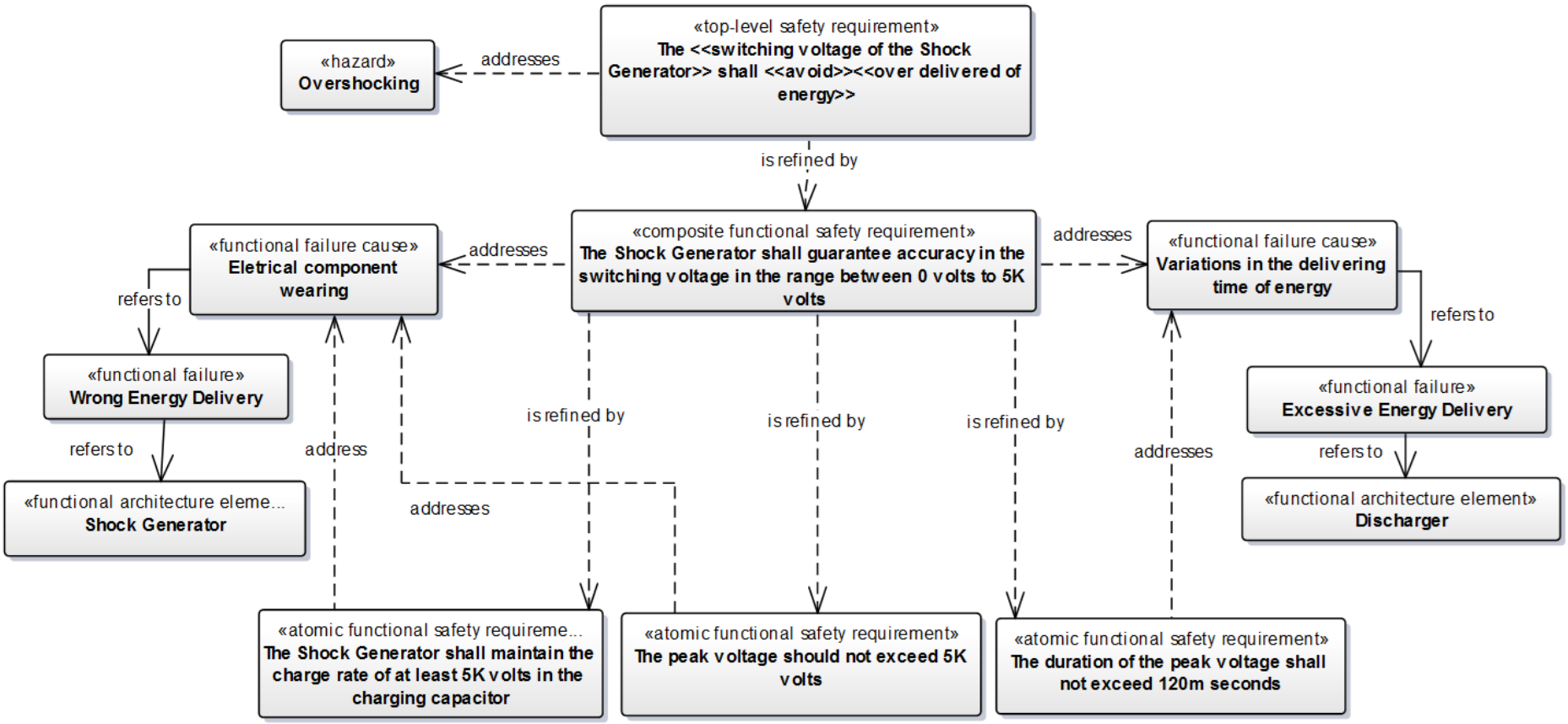
http://www.lifemed.com.br





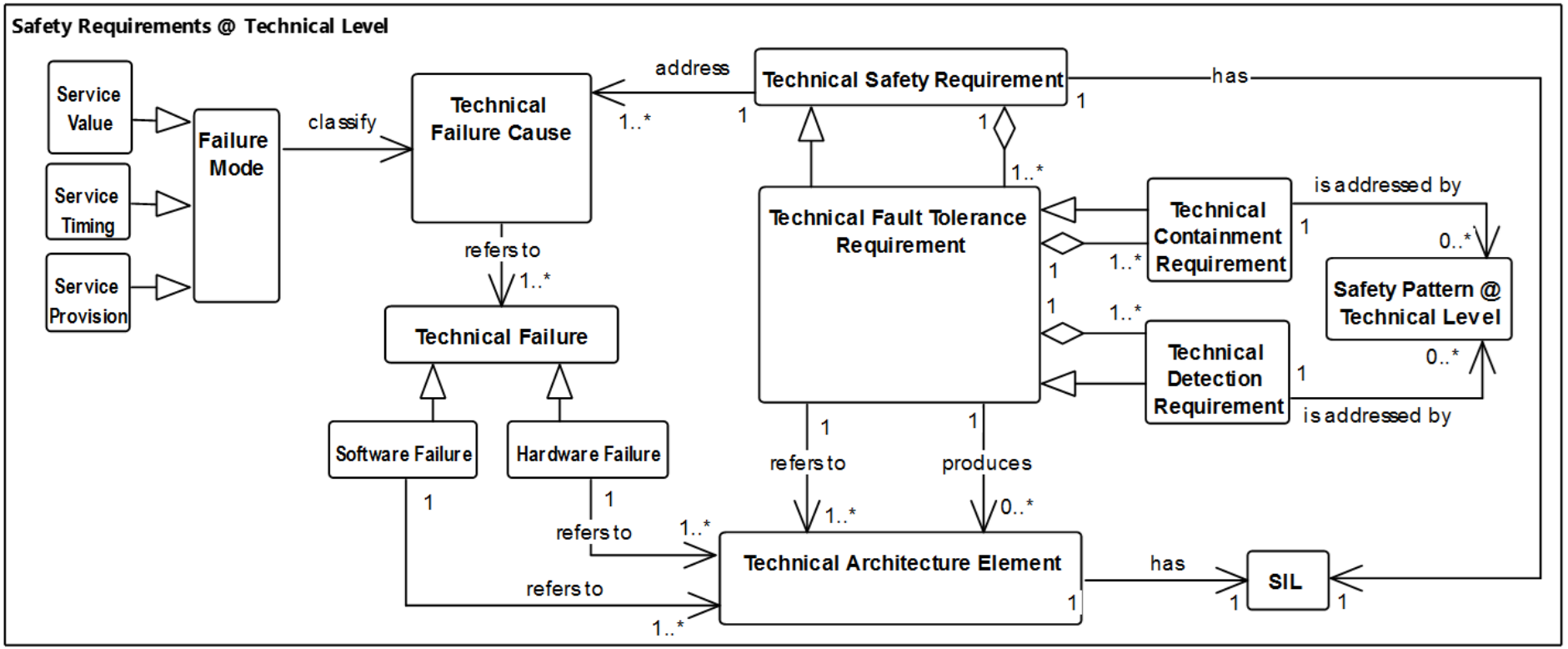
Fraunhofer IESE





# The Safety Requirements Decomposition Pattern @ the Technical Level

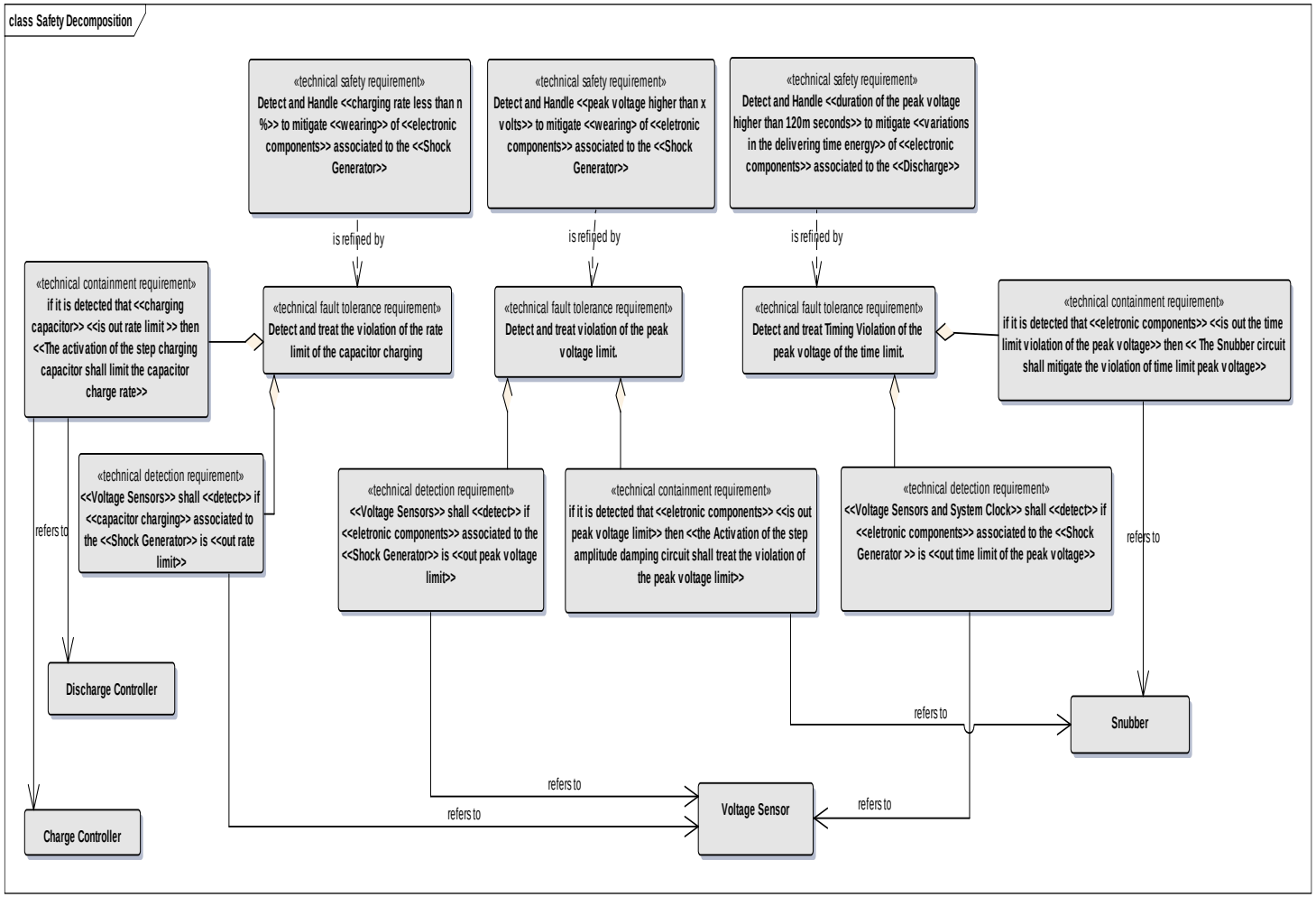
# Safety Requirements Decomposition Pattern @ Technical Level

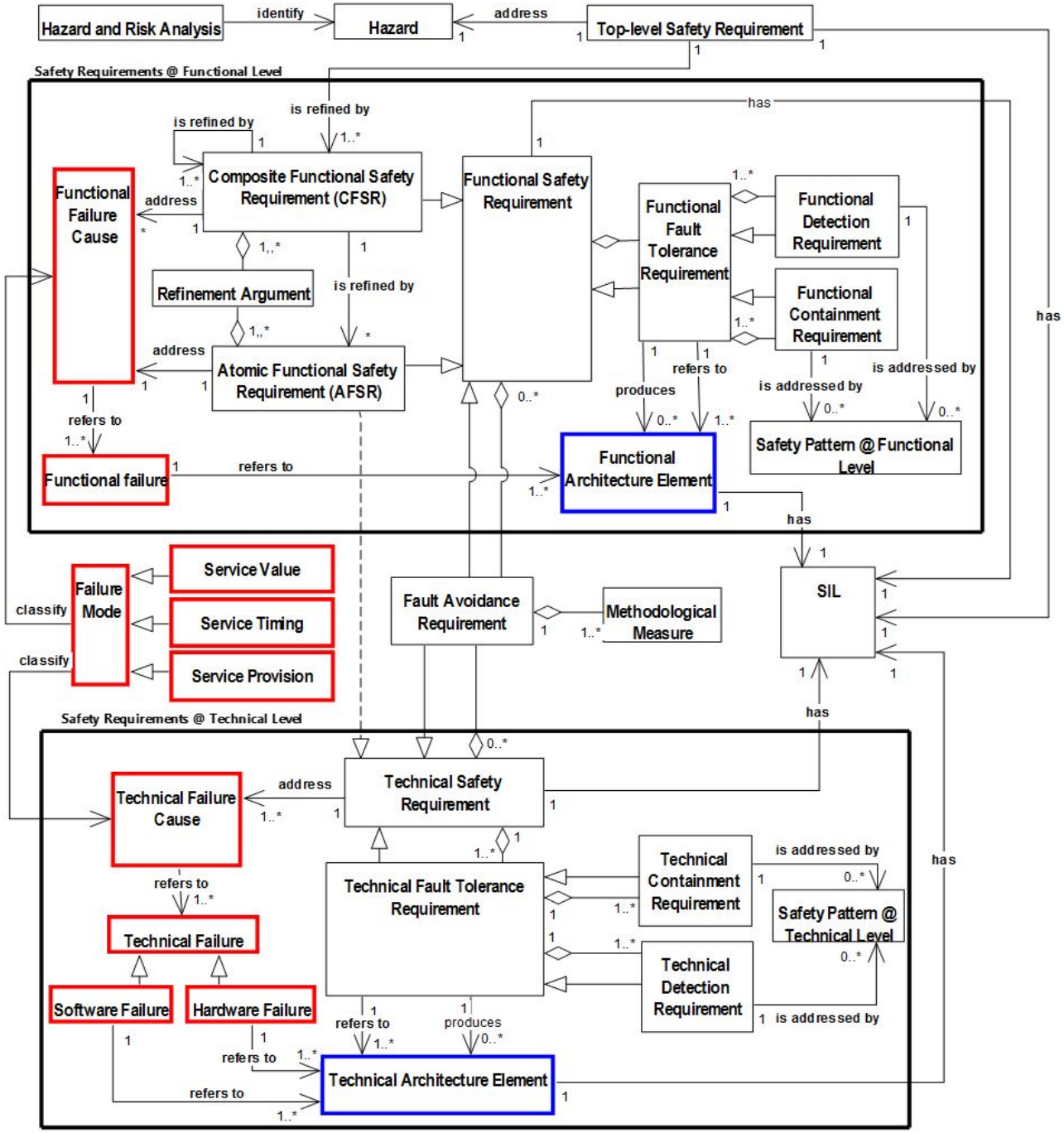




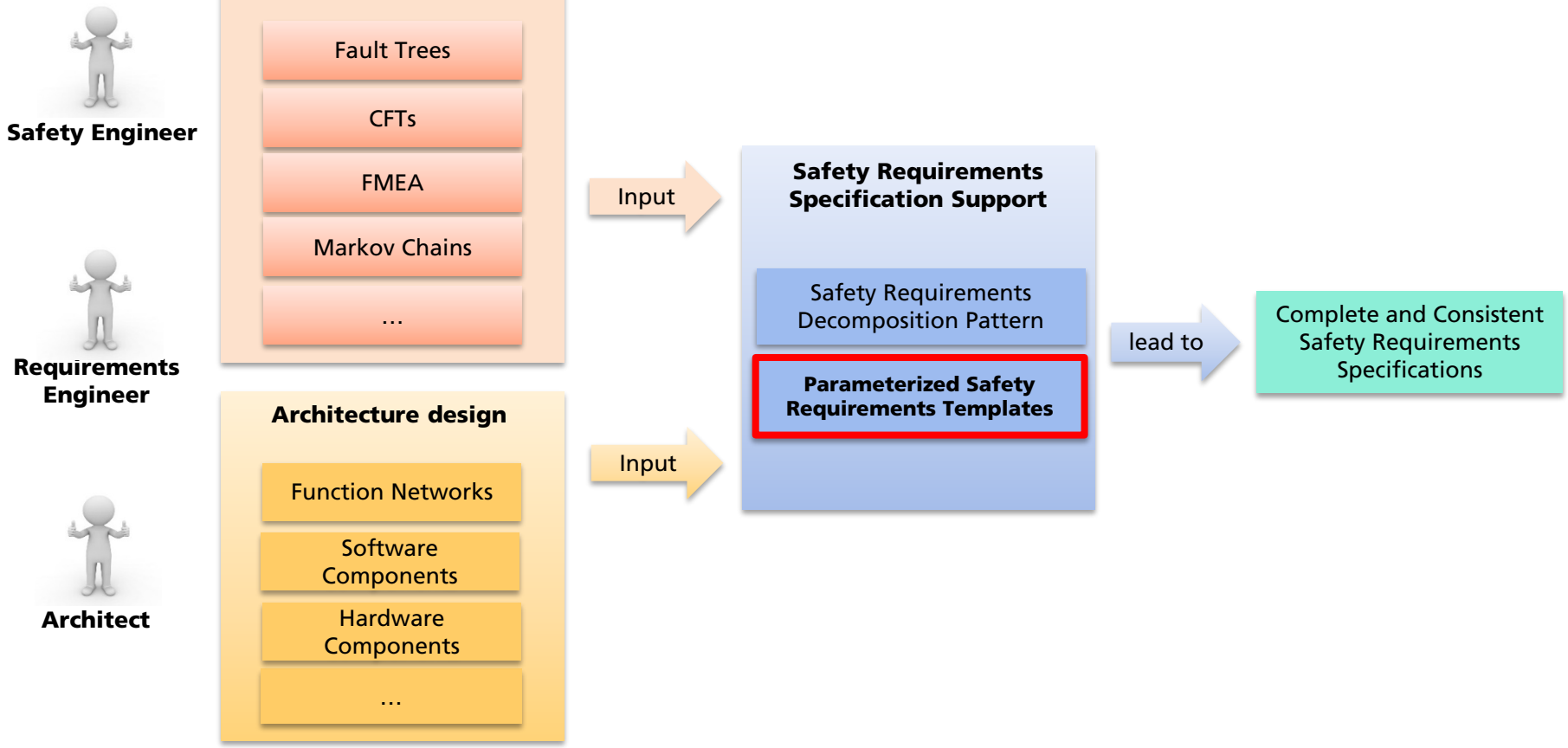



Fraunhofer IESE




Hazard and Risk Analysis
identify
Hazard
address
Top-level Safety Requirement
Safety Requirements @ Functional Level
is refined by
has
Functional Failure Cause
address
Composite Functional Safety Requirement (CFSR)
Functional Safety Requirement
Functional Fault Tolerance Requirement
Functional Detection Requirement
Functional Containment Requirement
Refinement Argument
is refined by
address
Atomic Functional Safety Requirement (AFSR)
refers to
produces
is addressed by
Safety Pattern @ Functional Level
Functional failure
refers to
Functional Architecture Element
has
SIL
Failure Mode
Service Value
Service Timing
Service Provision
classify
Fault Avoidance Requirement
Methodological Measure
Safety Requirements @ Technical Level
Technical Failure Cause
address
Technical Safety Requirement
Technical Fault Tolerance Requirement
Technical Containment Requirement
Technical Detection Requirement
Safety Pattern @ Technical Level
Technical Failure
Software Failure
Hardware Failure
refers to
produces
Technical Architecture Element

# Safety Requirements Specification Support



# Designing the Parameterized Safety Requirements Templates

# Designing the Parameterized Safety Requirements Templates (1/3)

**[Condition] [Subject] [Action] [Object] [Constraint]**

**EXAMPLE:** When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint].**

**Or**

**[Condition] [Action or Constraint] [Value]**

**EXAMPLE:** At sea state 1 **[Condition]**, the Radar System shall detect targets at ranges out to **[Action or Constraint]** 100 nautical miles **[Value]**.

**Or**

**[Subject] [Action] [Value]**

**EXAMPLE:** The Invoice System **[Subject]**, shall display pending customer invoices **[Action]** in ascending order **[Value]** in which invoices are to be paid.

*ISO/IEC/IEEE 29148:2011 Systems and software engineering - Life cycle processes - Requirements engineering.*

# Designing the Parameterized Safety Requirements Templates (2/3)

- Acceptable failure mode and rates;
- Qualitative requirements for failure modes;
- Elements of the architecture that address the safety requirements demands.

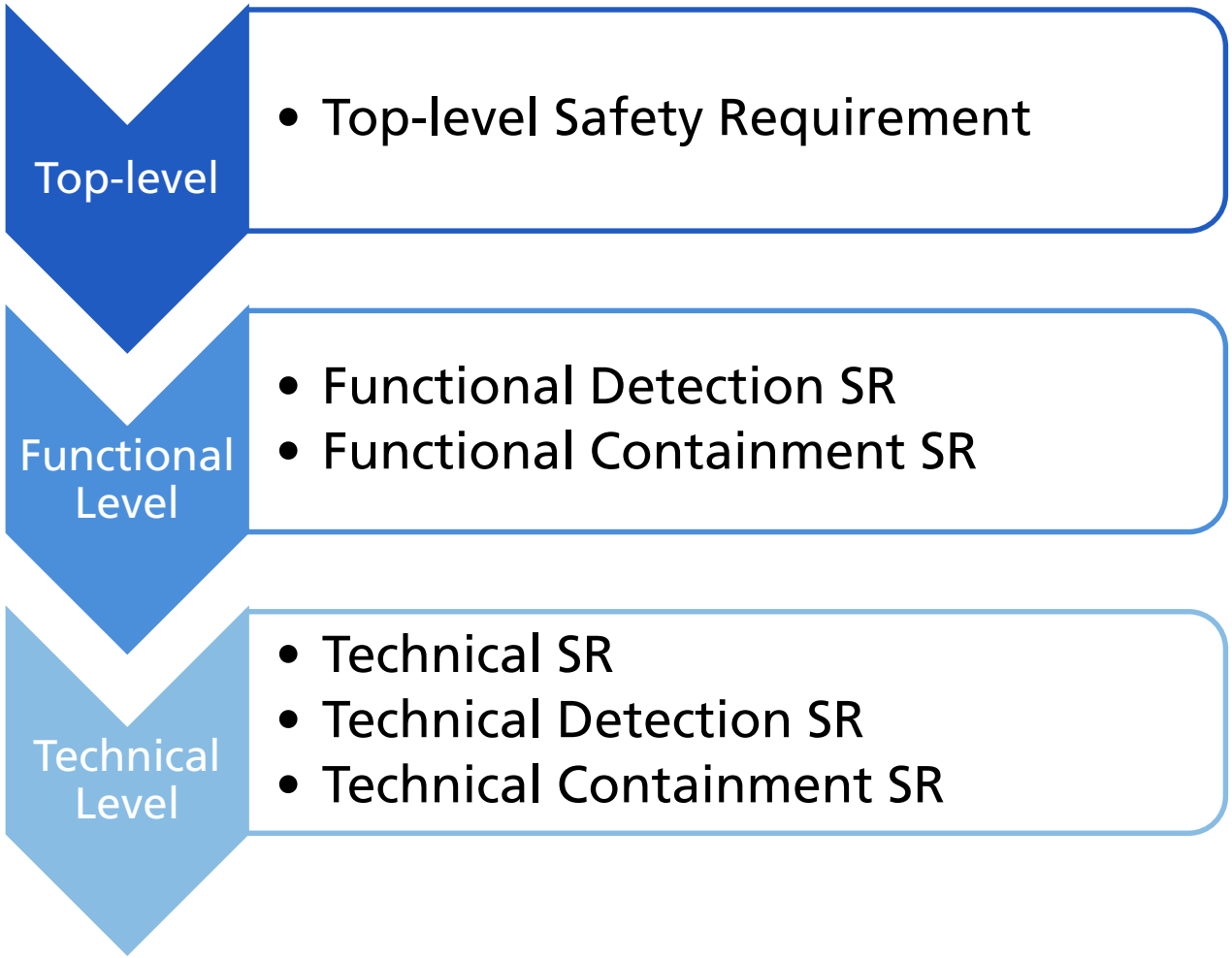
# Designing the Parameterized Safety Requirements Templates (3/3)

- “Requirements are mandatory binding provisions and use 'shall'.”;
- “It is best to avoid using the term ‘must', due to potential misinterpretation as a requirement.”;
- “Use positive statements and avoid negative requirements such as ‘shall not'.”;
- “Use active voice: avoid using passive voice, such as 'shall be able to select'.”;
- …

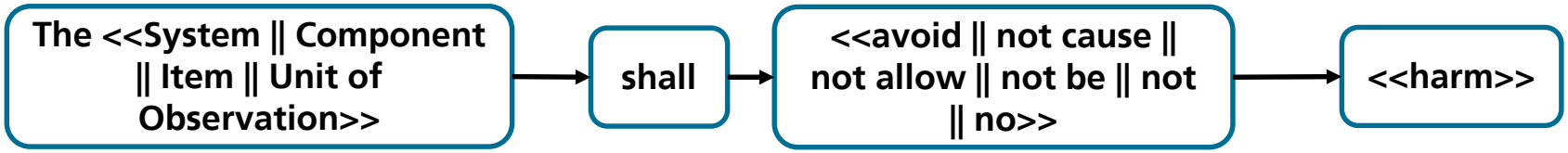
*ISO/IEC/IEEE 29148:2011 Systems and software engineering - Life cycle processes - Requirements engineering.*

# The Parameterized Safety Requirements Templates

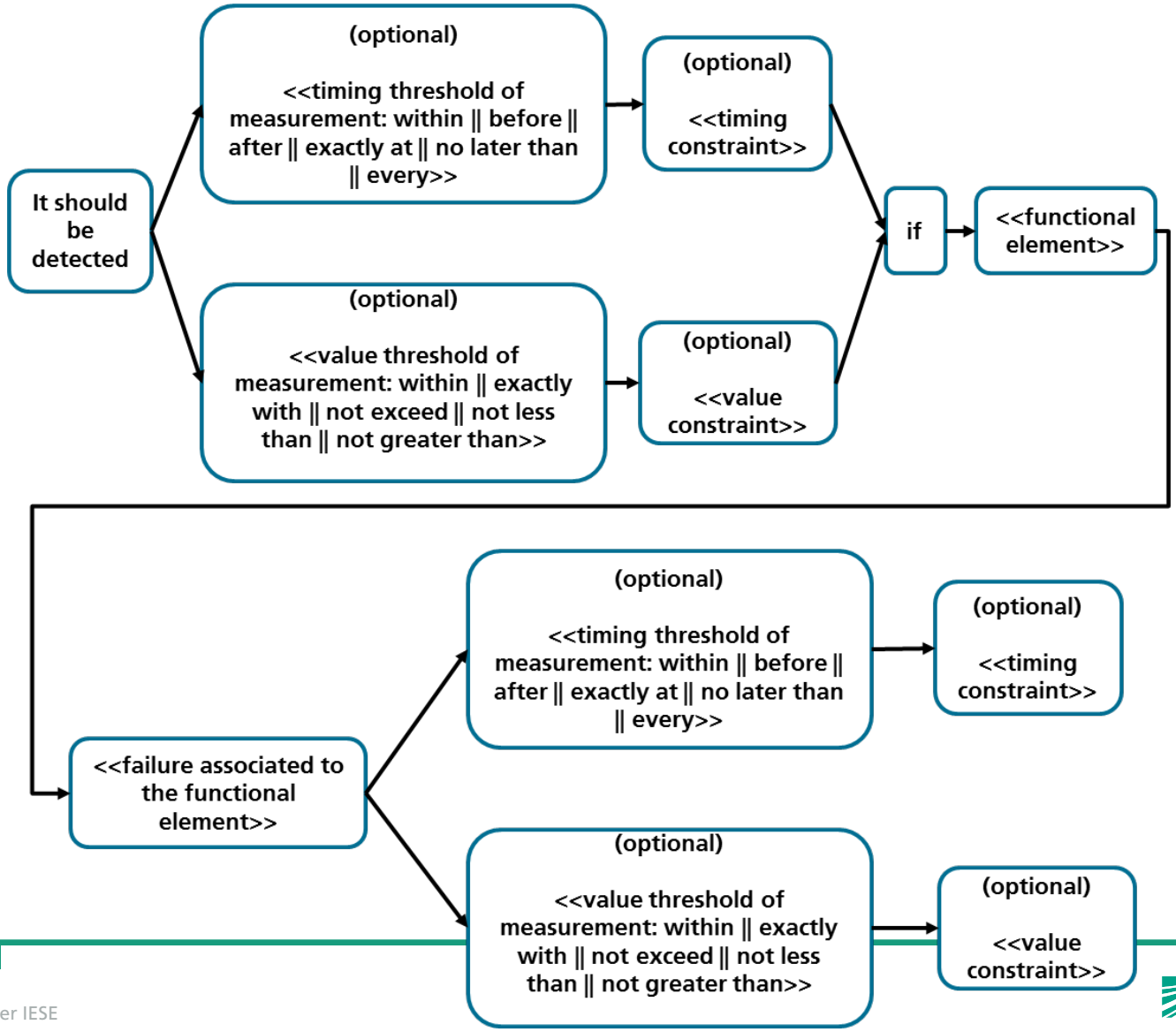
# Safety Requirements Decomposition Pattern elements with Templates



# Top Level Safety Requirement Template



# Functional Detection Requirements Template

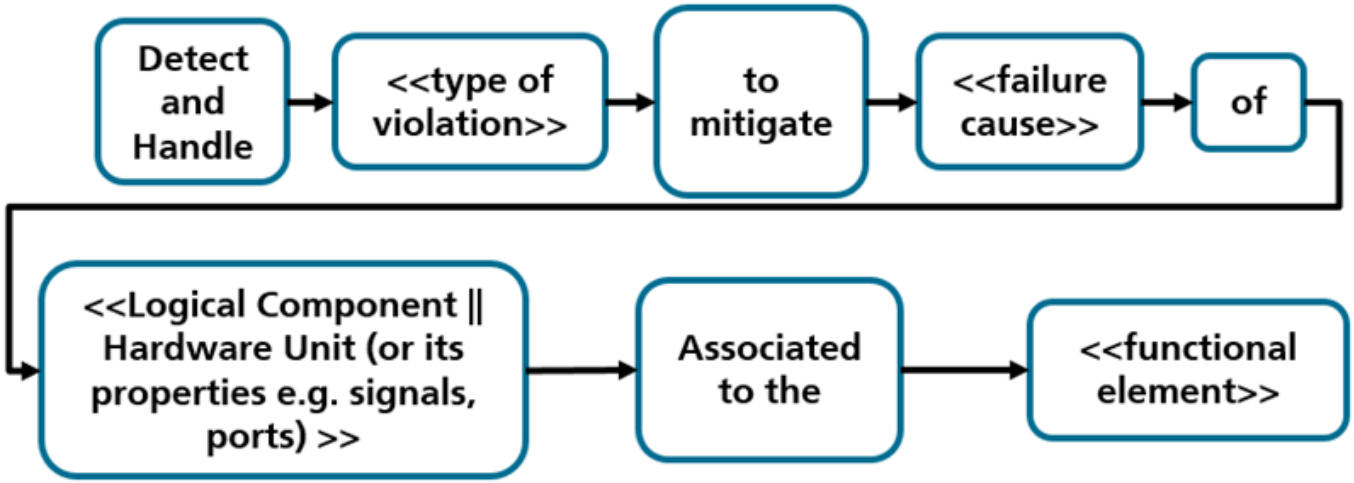




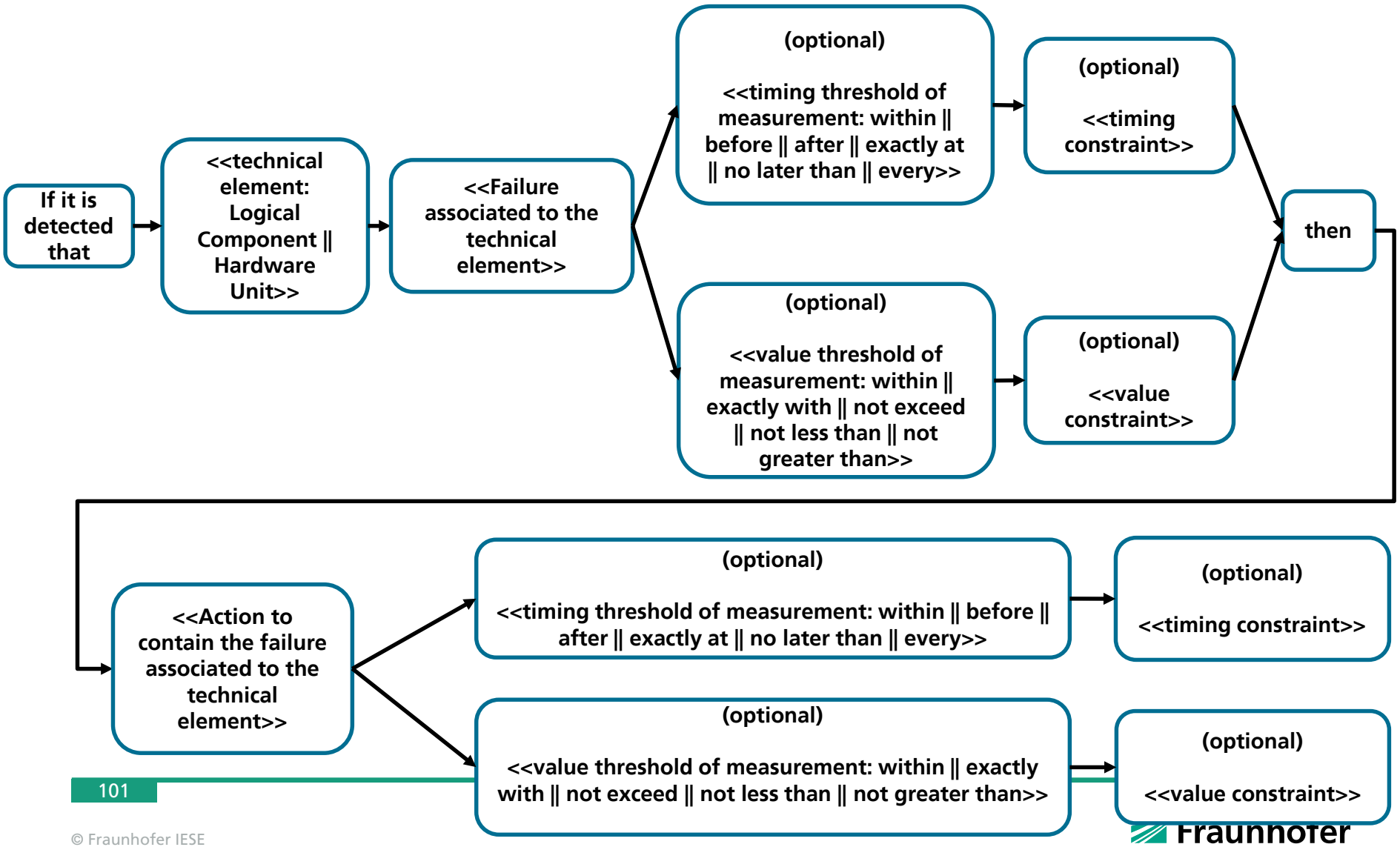



Fraunhofer IESE

# Technical Safety Requirement



# Technical Containment Safety Requirement Template



# Safety Patterns @ Functional and Technical Levels

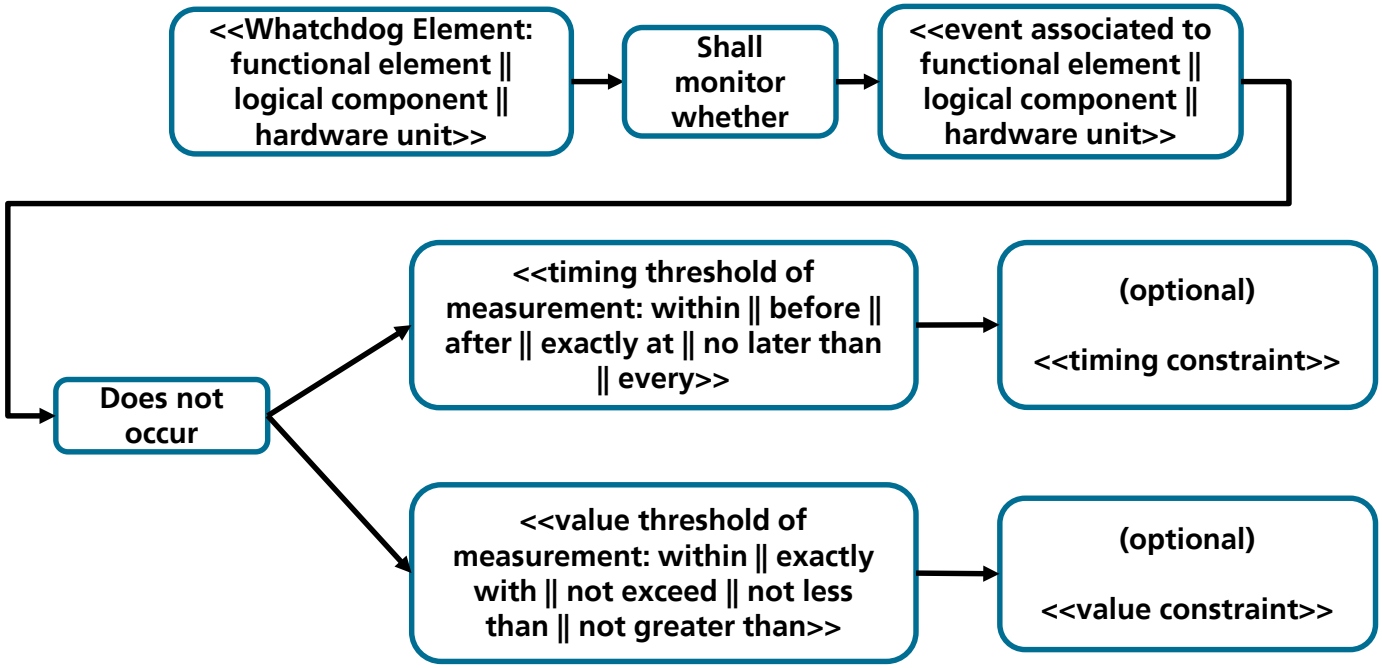
## Detection Safety Patterns

- Sanity Check
- Watchdog
- Comparison

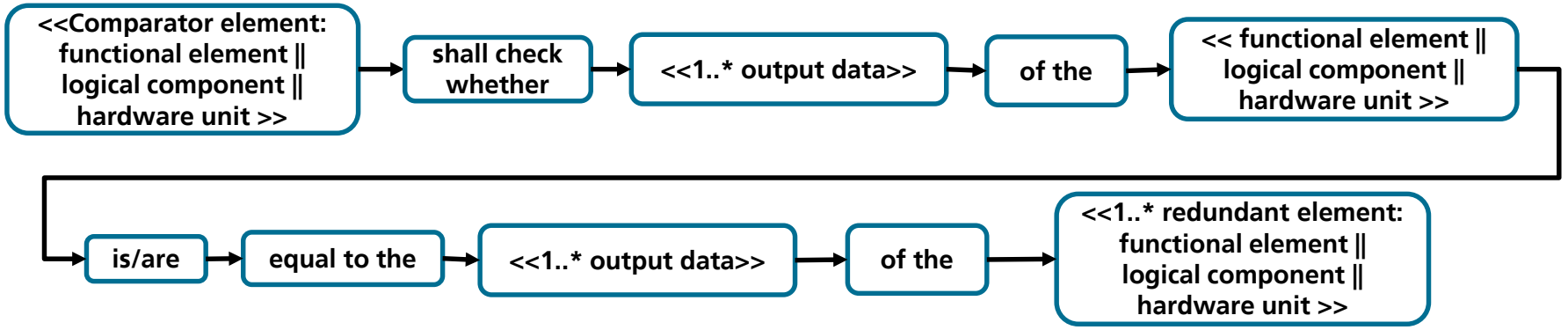
## Containment Safety Patterns

- Redundancy
- Reconfiguration
- Degradation
- Firewall
- Interlock
- Voting

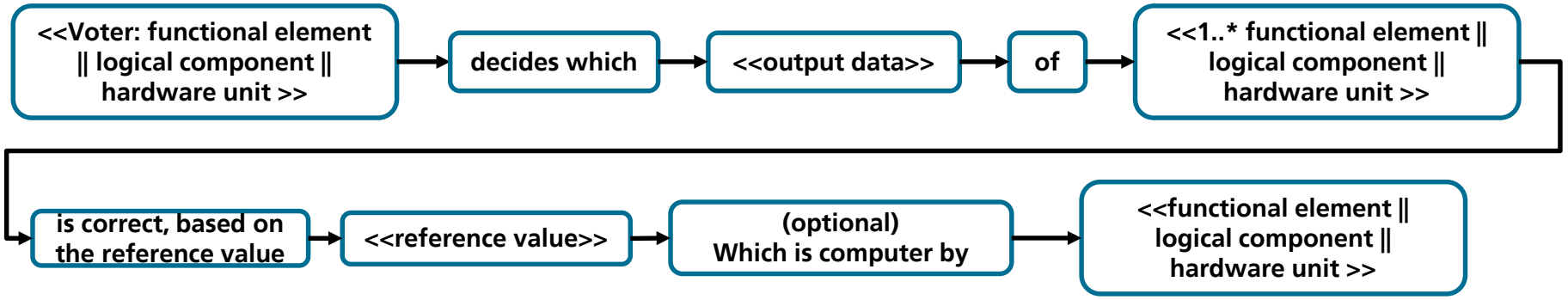
# Watchdog



# Comparator




# Voting









# Relevant Publications


- *Pablo Oliveira Antonino*, Mario Trapp. **Improving Consistency Checks between Safety Concepts and View Based Architecture Design.** In Proceedings of the 12 Probabilistic Safety Assessment and Management Conference (PSAM12), Honolulu, Hawaii, USA, 2014.
- *Pablo Oliveira Antonino*, Mario Trapp, Ashwin Venugopal. **Automatic Detection of Incomplete and Inconsistent Safety Requirements**. SAE 2015 World Congress and Exhibition, Detroit, Michigan USA, 2015.
- *Pablo Oliveira Antonino*, Mario Trapp, Paulo Barbosa, Luana Sousa. **The Parameterized Safety Requirements Templates**. 8th IEEE International Symposium on Software and Systems Traceability – an ICSE 2015 Symposium. Florence, Italy, 2015.
- *Pablo Oliveira Antonino,* David Santiago Velasco Moncada, Daniel Schneider, Mario Trapp, Jan Reich. **I-SafE: An integrated Safety Engineering Tool-Framework**. The 5th International Workshop on Dependable Control and Discrete Systems. Mexico, 2015.
- *Pablo Oliveira Antonino*, Mario Trapp, Paulo Barbosa, Edmar C. Gurjão, Jeferson Rosário. **The Safety Requirements Decomposition Pattern.** SAFECOMP 2015. Delft, The Netherlands, 2015.
- *Pablo Oliveira Antonino*, D. S. Velasco Moncada, T. Kuhn, D. Schneider and M. Trapp, **Integrated Model-based Safety Engineering with I-SafE.** *Embedded Software Engineering Kongress 2015 (ESE 2015)*, Sindelfingen, Germany, 2015.
- P. Barbosa, F. Leite, R. Mendonca, M. Andrade, L. Sousa and *Pablo Oliveira Antonino*. RAwTIM – **Uma Ferramenta para Rastreabilidade da Informação em Análises de Riscos.** in *Brazilian Conference on Software: Theory and Practice – Tools Section*, Belo Horizonte, Brazil, 2015.
- Thomas Kuhn, *Pablo Oliveira Antonino*. **Model-Driven Development of Embedded Systems**. Embedded Software Engineering Congress 2014. Sindelfingen, Germany, December 2014.

**Pablo Oliveira Antonino**
Phone: +49 631 6800-2213
Mail: pablo.antonino@iese.fraunhofer.de

Fraunhofer IESE: http://www.iese.fraunhofer.de