SECURITY & SAFETY

AUTOMOTIVE SPICE ®

PROCESS IMPROVEMENT

PF

PROCESS

FELLOWS

PROCESS FELLOWS

Harmonization of standards in automotive development

Increased efficiency through an integrated lifecycle approach for ASPICE 4.0, ISO 26262, and ISO 21434

Timo Karasch/ 12 January 2026

Agenda

- Introduction
- Extended Life Cycle Approach
  - Integrating the requirements of ASPICE, ISO 26262 and ISO 21434 into one standard process
- Practical example how this could be implemented
  - Software Architectural Design (SWE.2)
- Mapping strategy
  - ISO 26262 and ISO 21434 objectives towards ASPICE 4.0
- Reporting
  - for ASPICE Assessment, Safety and Cybersecurity Audit – plus interface for Safety/Security Assessment
- Summary

Introduction

What is the idea?

In this webinar we will combine the content of

Automotive SPICE

Functional Safety

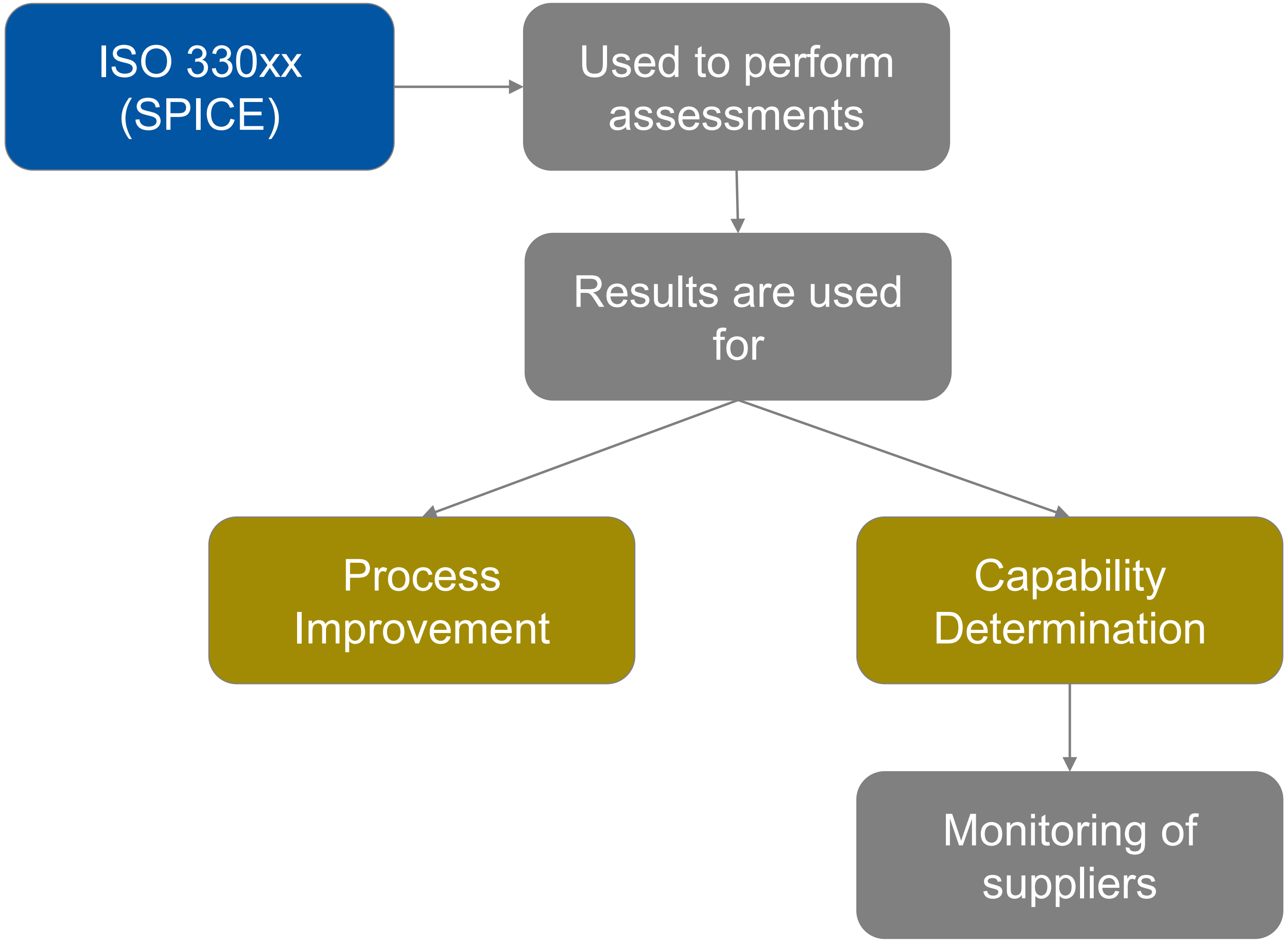
Cyber Security

We will:

- learn about the contents
- see the combinations
- identify the differences

What is the purpose of SPICE?

Automotive SPICE



ISO 330xx (SPICE)

Used to perform assessments

Results are used for

Process Improvement

Capability Determination

Monitoring of suppliers

What is the meaning of “Safety”?

Functional Safety

Safety is the absence of an unreasonable risks (that can cause harm).

We are interested in **Functional Safety**, that means…

… the absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems.

Safety shall be achieved during development through a strategy:

- **Plan** all necessary activities
- **Do** perform the activities
- **Check** the results against plan
- **Act** in case of any deviations

➔ This is **Safety Management**

What is the meaning of “Security”?

Cyber Security

Security is prevention against threats caused by deliberately induced events.

We are interested in **Cyber Security**, that means…

… protection/resilience against (intentional) attacks (including accidents/hazards) on the **confidentiality**, **integrity** and **availability** of assets related to an E/E system.

Security shall be achieved **during and after** the development through a strategy:

- Continuous and prompt adaptation of systems to defend against new threats.

➔ This is **Cybersecurity Management** based on **organizational polices, rules and processes.**

Safety vs. Security – what is the difference?

Functional Safety

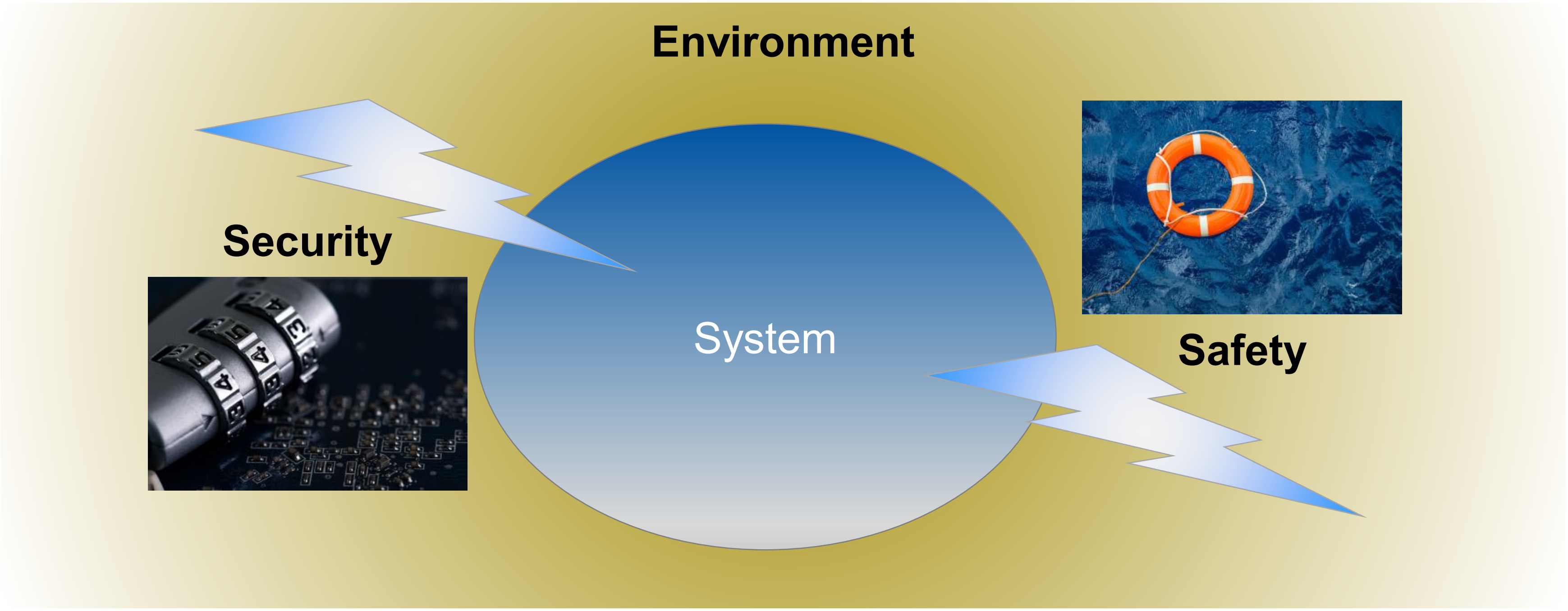
Cyber Security

Security means…

… we are considering potential hazards to our system caused by someone from outside!

Safety means…

… we are taking care of potential hazards caused by our system!



Security can influence **Safety**, but shall not disturb Safety ➔ „better safe than sorry“

Why are Safety and Security topics for us?

Functional Safety

Cyber Security

Trend:

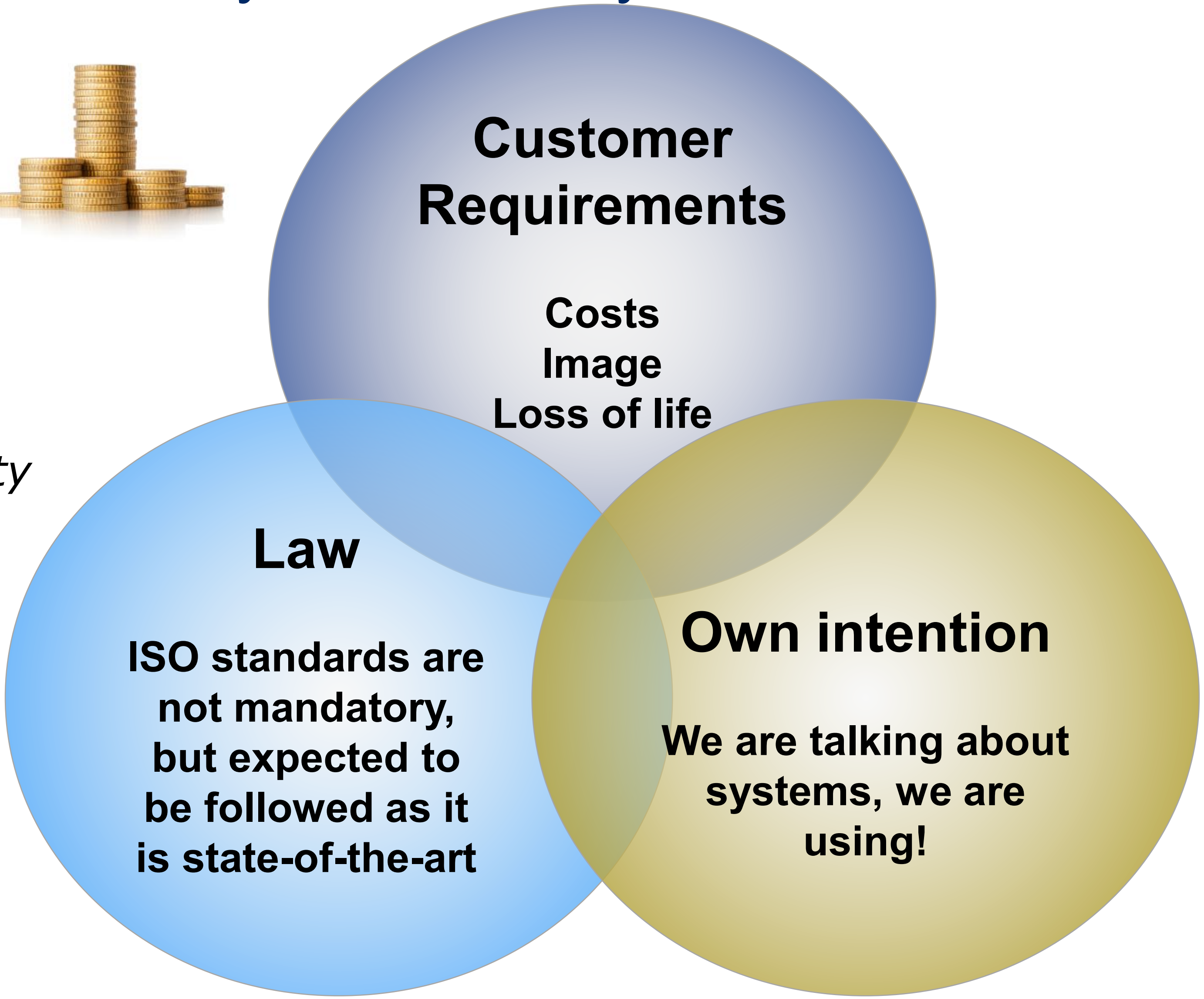
- More and more responsibility is given to E/E systems in automotive
- Growing complexity and growing intelligence of systems
- Distributed development = Higher need for communication
- Autonomous driving – Human driver is not in full control of the vehicle anymore
- Random and systematic failures are still part of E/E system development → Safety
- We must always expect targeted attacks on E/E systems → Security

Why do we need Safety and Security?

Functional Safety

Cyber Security

Product liability



Customer Requirements

Costs
Image
Loss of life

Law

ISO standards are not mandatory, but expected to be followed as it is state-of-the-art

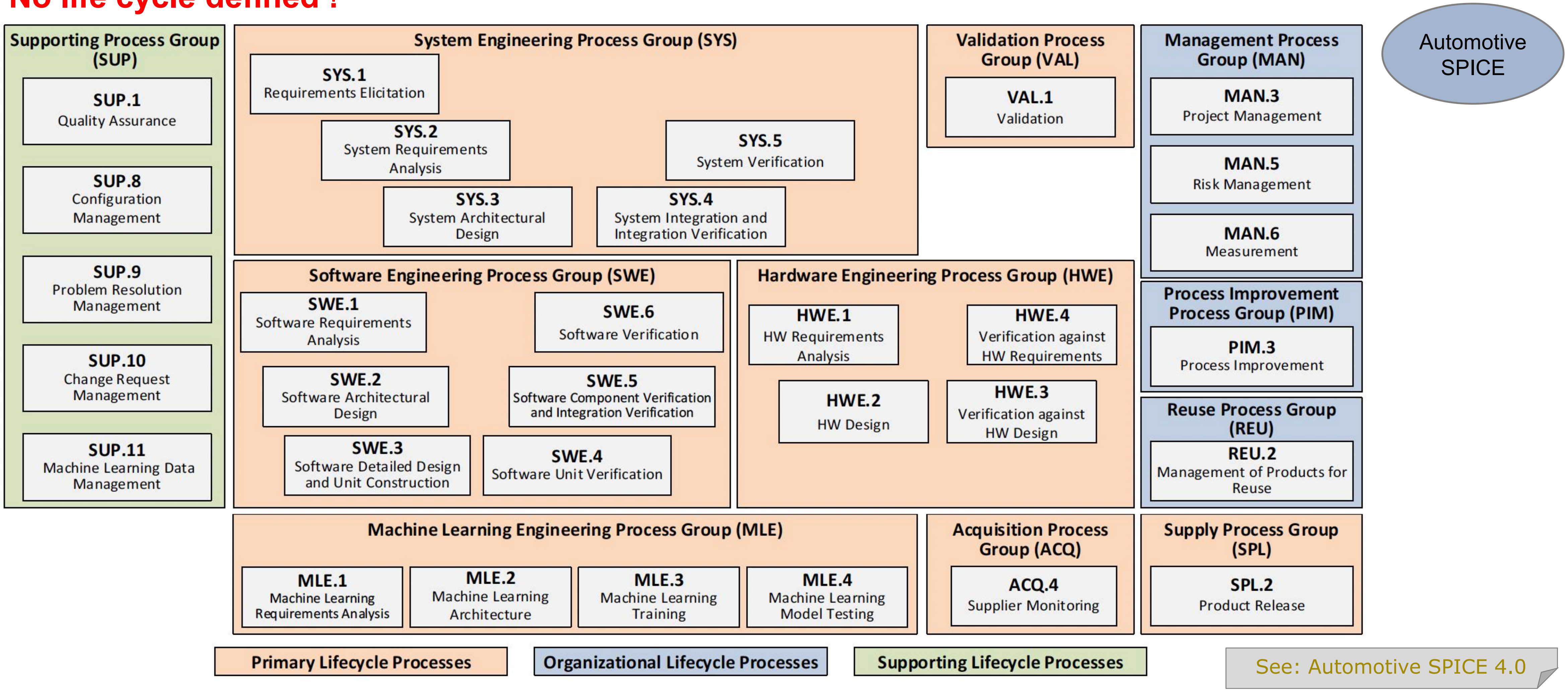
Own intention

We are talking about systems, we are using!

Extended Life Cycle Approach –
Integrating the requirements of ASPICE, ISO 26262 and ISO 21434 into one standard process

The Automotive SPICE® Process Reference Model 4.0

No life cycle defined !



Automotive SPICE

Supporting Process Group (SUP)
- **SUP.1** Quality Assurance
- **SUP.8** Configuration Management
- **SUP.9** Problem Resolution Management
- **SUP.10** Change Request Management
- **SUP.11** Machine Learning Data Management

System Engineering Process Group (SYS)
- **SYS.1** Requirements Elicitation
- **SYS.2** System Requirements Analysis
- **SYS.3** System Architectural Design
- **SYS.4** System Integration and Integration Verification
- **SYS.5** System Verification

Validation Process Group (VAL)
- **VAL.1** Validation

Management Process Group (MAN)
- **MAN.3** Project Management
- **MAN.5** Risk Management
- **MAN.6** Measurement

Software Engineering Process Group (SWE)
- **SWE.1** Software Requirements Analysis
- **SWE.2** Software Architectural Design
- **SWE.3** Software Detailed Design and Unit Construction
- **SWE.4** Software Unit Verification
- **SWE.5** Software Component Verification and Integration Verification
- **SWE.6** Software Verification

Hardware Engineering Process Group (HWE)
- **HWE.1** HW Requirements Analysis
- **HWE.2** HW Design
- **HWE.3** Verification against HW Design
- **HWE.4** Verification against HW Requirements

Process Improvement Process Group (PIM)
- **PIM.3** Process Improvement

Reuse Process Group (REU)
- **REU.2** Management of Products for Reuse

Machine Learning Engineering Process Group (MLE)
- **MLE.1** Machine Learning Requirements Analysis
- **MLE.2** Machine Learning Architecture
- **MLE.3** Machine Learning Training
- **MLE.4** Machine Learning Model Testing

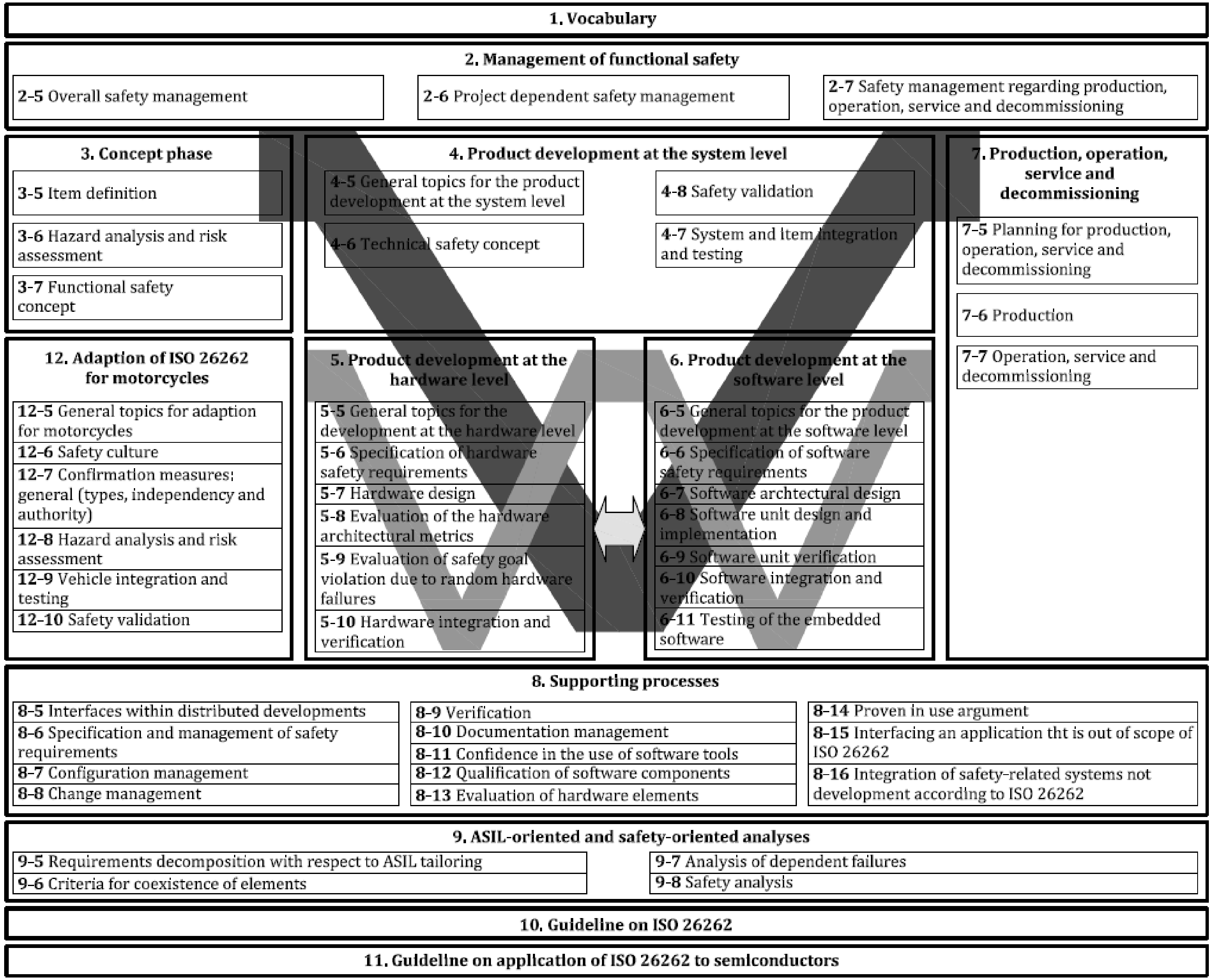
Acquisition Process Group (ACQ)
- **ACQ.4** Supplier Monitoring

Supply Process Group (SPL)
- **SPL.2** Product Release

Primary Lifecycle Processes | **Organizational Lifecycle Processes** | **Supporting Lifecycle Processes**

See: Automotive SPICE 4.0

The structure of ISO 26262



1. Vocabulary

2. Management of functional safety

- 2-5 Overall safety management
- 2-6 Project dependent safety management
- 2-7 Safety management regarding production, operation, service and decommissioning

3. Concept phase

- 3-5 Item definition
- 3-6 Hazard analysis and risk assessment
- 3-7 Functional safety concept

4. Product development at the system level

- 4-5 General topics for the product development at the system level
- 4-6 Technical safety concept
- 4-7 System and item integration and testing
- 4-8 Safety validation

7. Production, operation, service and decommissioning

- 7-5 Planning for production, operation, service and decommissioning
- 7-6 Production
- 7-7 Operation, service and decommissioning

12. Adaption of ISO 26262 for motorcycles

- 12-5 General topics for adaption for motorcycles
- 12-6 Safety culture
- 12-7 Confirmation measures: general (types, independency and authority)
- 12-8 Hazard analysis and risk assessment
- 12-9 Vehicle integration and testing
- 12-10 Safety validation

5. Product development at the hardware level

- 5-5 General topics for the development at the hardware level
- 5-6 Specification of hardware safety requirements
- 5-7 Hardware design
- 5-8 Evaluation of the hardware architectural metrics
- 5-9 Evaluation of safety goal violation due to random hardware failures
- 5-10 Hardware integration and verification

6. Product development at the software level

- 6-5 General topics for the product development at the software level
- 6-6 Specification of software safety requirements
- 6-7 Software archtectural design
- 6-8 Software unit design and implementation
- 6-9 Software unit verification
- 6-10 Software integration and verification
- 6-11 Testing of the embedded software

8. Supporting processes

- 8-5 Interfaces within distributed developments
- 8-6 Specification and management of safety requirements
- 8-7 Configuration management
- 8-8 Change management
- 8-9 Verification
- 8-10 Documentation management
- 8-11 Confidence in the use of software tools
- 8-12 Qualification of software components
- 8-13 Evaluation of hardware elements
- 8-14 Proven in use argument
- 8-15 Interfacing an application tht is out of scope of ISO 26262
- 8-16 Integration of safety-related systems not development according to ISO 26262

9. ASIL-oriented and safety-oriented analyses

- 9-5 Requirements decomposition with respect to ASIL tailoring
- 9-6 Criteria for coexistence of elements
- 9-7 Analysis of dependent failures
- 9-8 Safety analysis

10. Guideline on ISO 26262

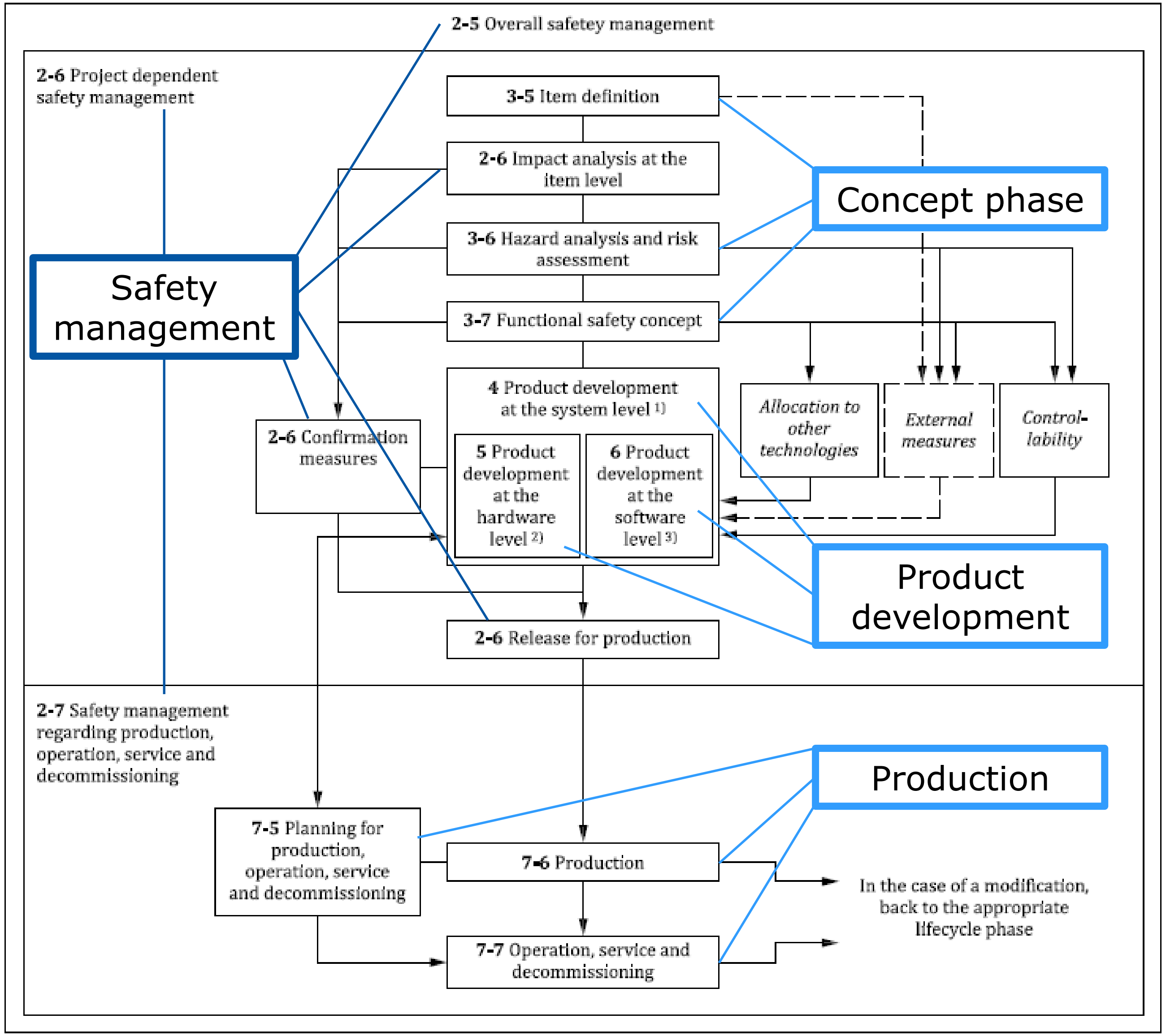
11. Guideline on application of ISO 26262 to semiconductors

Functional Safety

See: ISO 26262 – Part 2

The Safety lifecycle

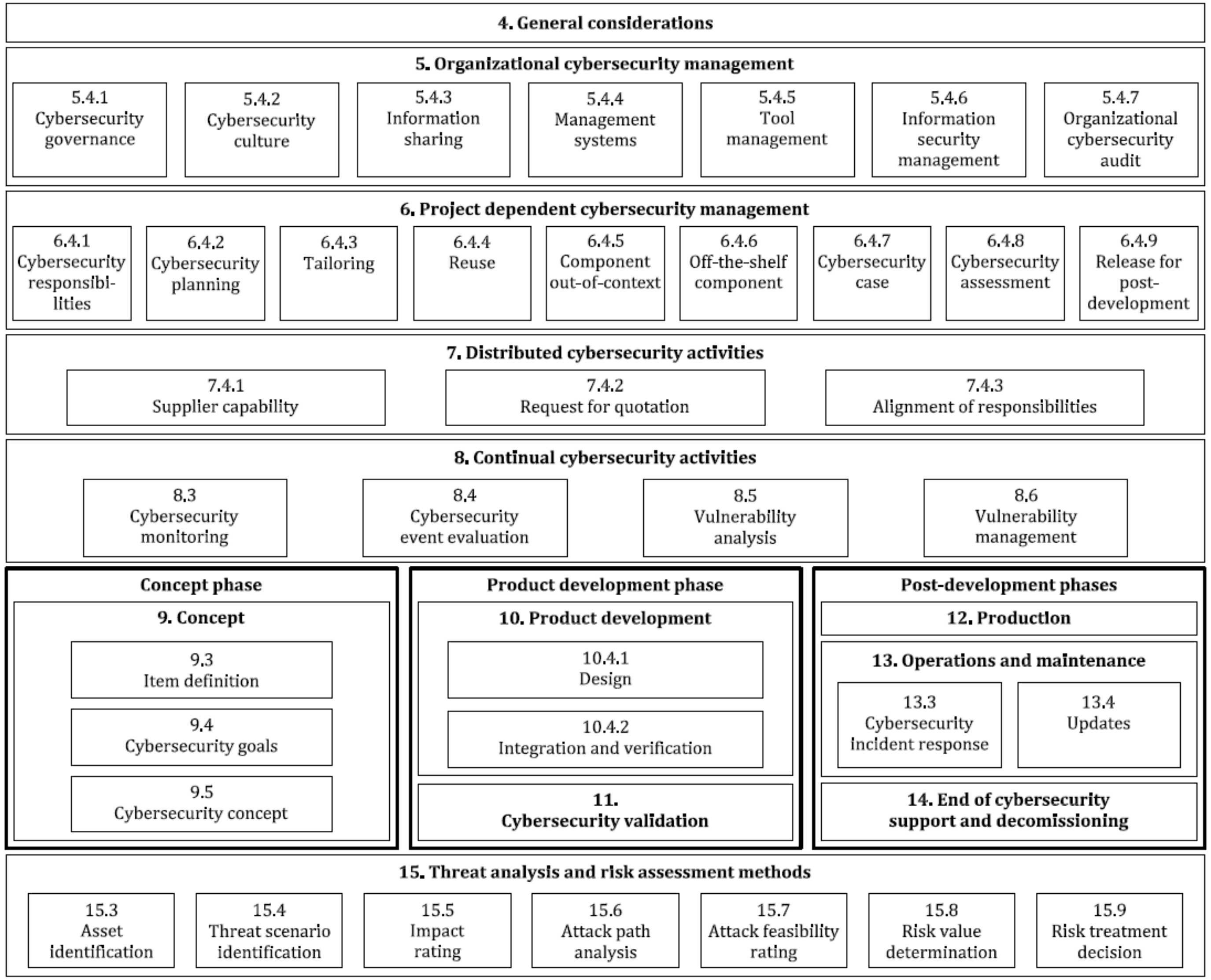
Functional Safety


2-5 Overall safetey management
2-6 Project dependent safety management
3-5 Item definition
2-6 Impact analysis at the item level
3-6 Hazard analysis and risk assessment
3-7 Functional safety concept
4 Product development at the system level 1)
5 Product development at the hardware level 2)
6 Product development at the software level 3)
2-6 Confirmation measures
Allocation to other technologies
External measures
Control-lability
2-6 Release for production
2-7 Safety management regarding production, operation, service and decommissioning
7-5 Planning for production, operation, service and decommissioning
7-6 Production
7-7 Operation, service and decommissioning
In the case of a modification, back to the appropriate lifecycle phase
Safety management
Concept phase
Product development
Production

See: ISO 26262 – Part 2

The ISO/SAE 21434 Structure

Cyber Security



4. General considerations

5. Organizational cybersecurity management

- 5.4.1 Cybersecurity governance
- 5.4.2 Cybersecurity culture
- 5.4.3 Information sharing
- 5.4.4 Management systems
- 5.4.5 Tool management
- 5.4.6 Information security management
- 5.4.7 Organizational cybersecurity audit

6. Project dependent cybersecurity management

- 6.4.1 Cybersecurity responsibilities
- 6.4.2 Cybersecurity planning
- 6.4.3 Tailoring
- 6.4.4 Reuse
- 6.4.5 Component out-of-context
- 6.4.6 Off-the-shelf component
- 6.4.7 Cybersecurity case
- 6.4.8 Cybersecurity assessment
- 6.4.9 Release for post-development

7. Distributed cybersecurity activities

- 7.4.1 Supplier capability
- 7.4.2 Request for quotation
- 7.4.3 Alignment of responsibilities

8. Continual cybersecurity activities

- 8.3 Cybersecurity monitoring
- 8.4 Cybersecurity event evaluation
- 8.5 Vulnerability analysis
- 8.6 Vulnerability management

Concept phase

9. Concept

- 9.3 Item definition
- 9.4 Cybersecurity goals
- 9.5 Cybersecurity concept

Product development phase

10. Product development

- 10.4.1 Design
- 10.4.2 Integration and verification

11. Cybersecurity validation

Post-development phases

12. Production

13. Operations and maintenance

- 13.3 Cybersecurity incident response
- 13.4 Updates

14. End of cybersecurity support and decomissioning

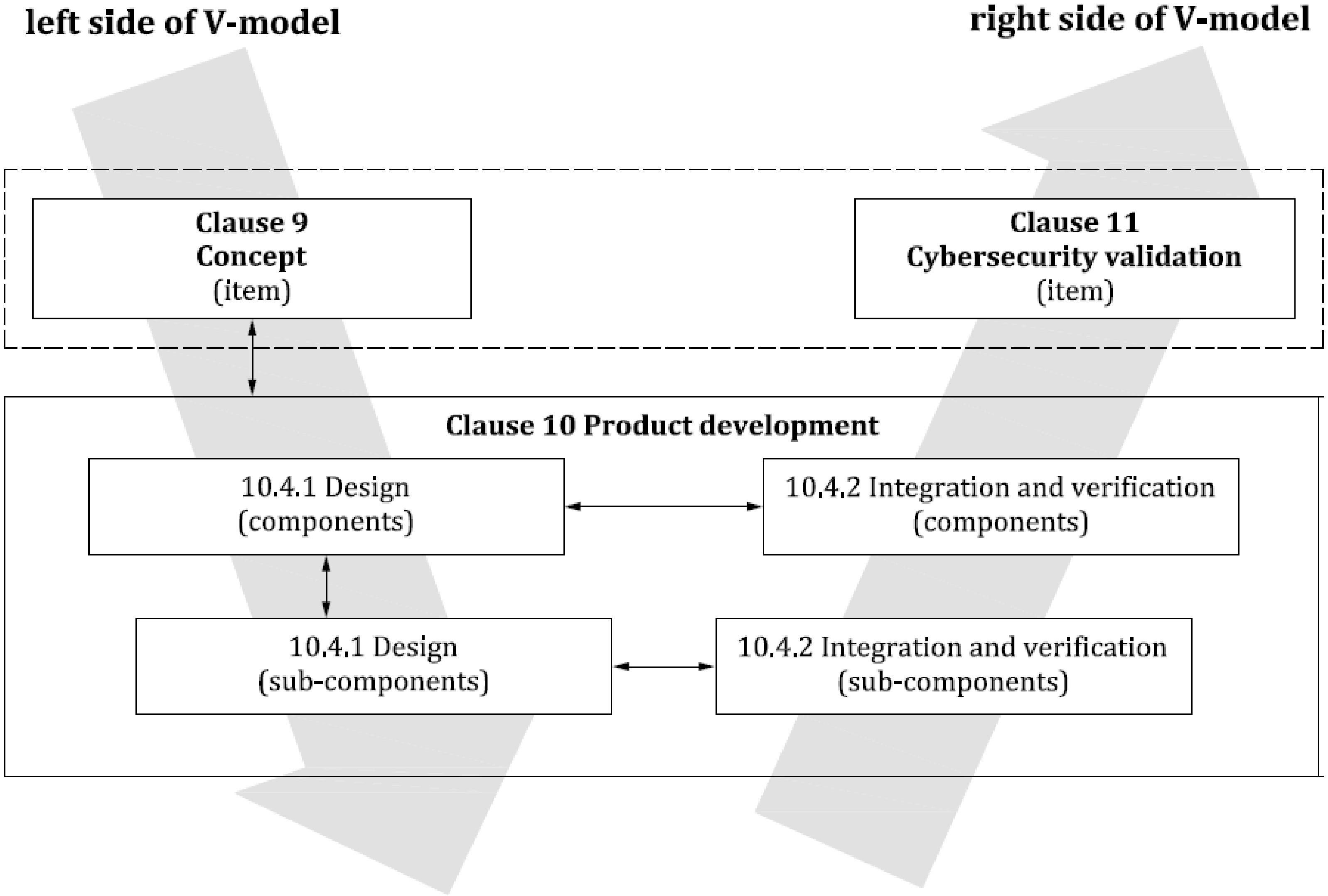
15. Threat analysis and risk assessment methods

- 15.3 Asset identification
- 15.4 Threat scenario identification
- 15.5 Impact rating
- 15.6 Attack path analysis
- 15.7 Attack feasibility rating
- 15.8 Risk value determination
- 15.9 Risk treatment decision

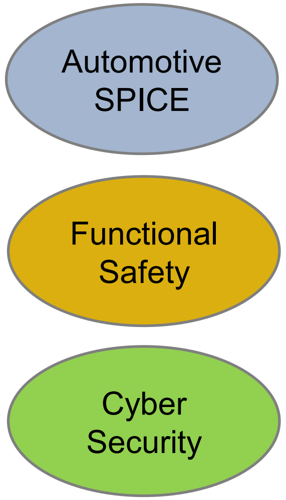
See: ISO/SAE 21434

Cybersecurity development

Cyber Security



What are the commonalities and differences?


Automotive SPICE

Functional Safety

Cyber Security

Commonalities:

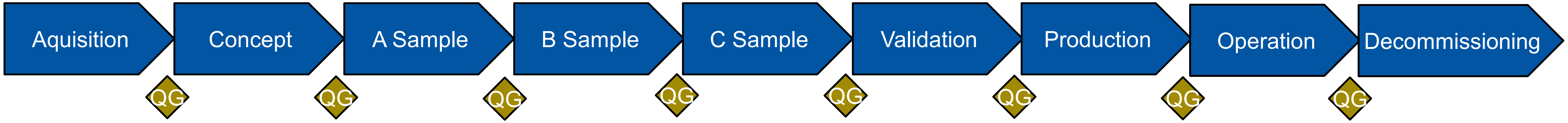
- **Complete product development** including system, software and hardware
- Management of **project-specific** development
- Application of **supporting** processes
- **Supplier** involvement

Differences:

- SPICE is only defining the **WHAT**, ISO 26262 and ISO/SAE 21434 add specific methods (**HOW**)
- SPICE is only focusing on **development** phases, ISO 26262 and ISO/SAE 21434 also on **post-development**, especially for new threats and vulnerabilities
- ISO 26262 and ISO/SAE 21434 start with some **risk and analysis activities** (HARA and TARA) to derive additional requirements for the product

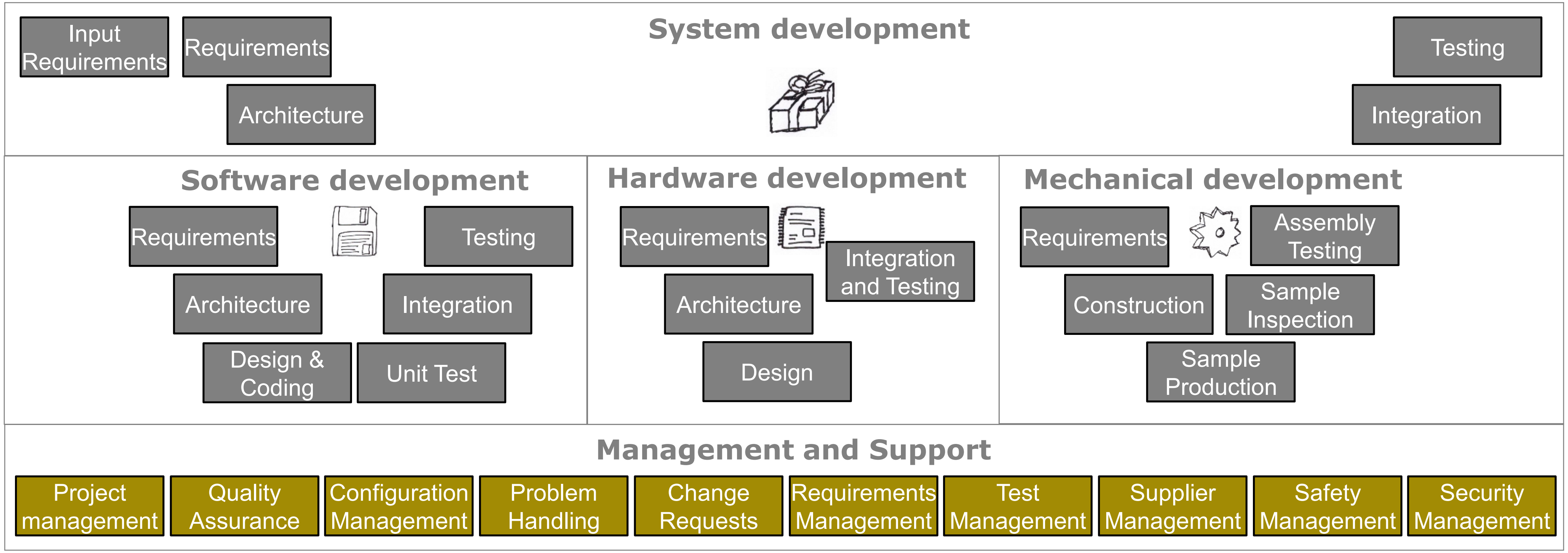
Exemplary Product Lifecycle incl. Processes

Product Lifecycle



Aquisition
QG
Concept
QG
A Sample
QG
B Sample
QG
C Sample
QG
Validation
QG
Production
QG
Operation
QG
Decommissioning

Process Overview



System development

Input Requirements
Requirements
Architecture
Testing
Integration

Software development

Requirements
Architecture
Design & Coding
Unit Test
Integration
Testing

Hardware development

Requirements
Architecture
Design
Integration and Testing

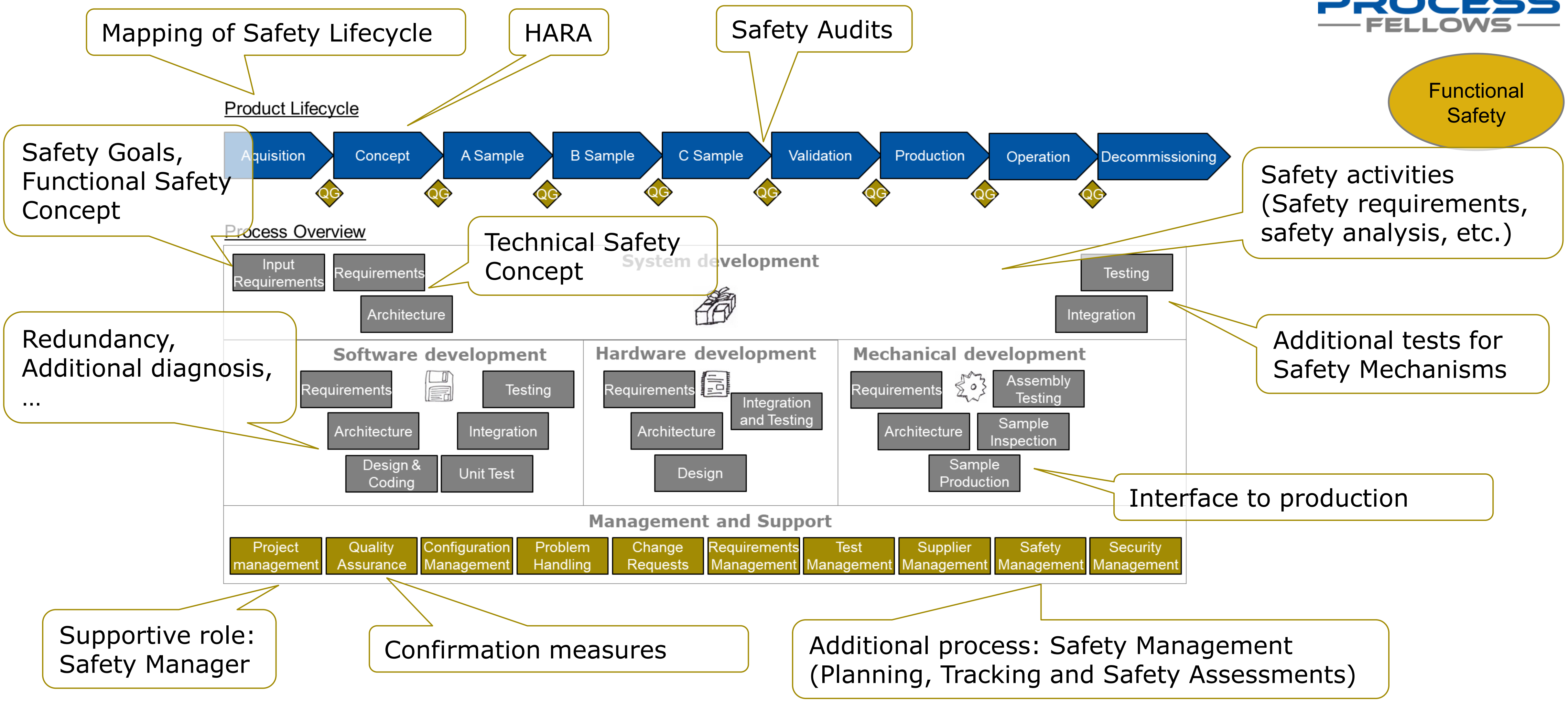
Mechanical development

Requirements
Construction
Sample Production
Sample Inspection
Assembly Testing

Management and Support

Project management
Quality Assurance
Configuration Management
Problem Handling
Change Requests
Requirements Management
Test Management
Supplier Management
Safety Management
Security Management

Adding Functional Safety



Mapping of Safety Lifecycle

HARA

Safety Audits

Safety Goals, Functional Safety Concept

Technical Safety Concept

Safety activities (Safety requirements, safety analysis, etc.)

Redundancy, Additional diagnosis, …

Additional tests for Safety Mechanisms

Interface to production

Supportive role: Safety Manager

Confirmation measures

Additional process: Safety Management (Planning, Tracking and Safety Assessments)

Adding Cybersecurity

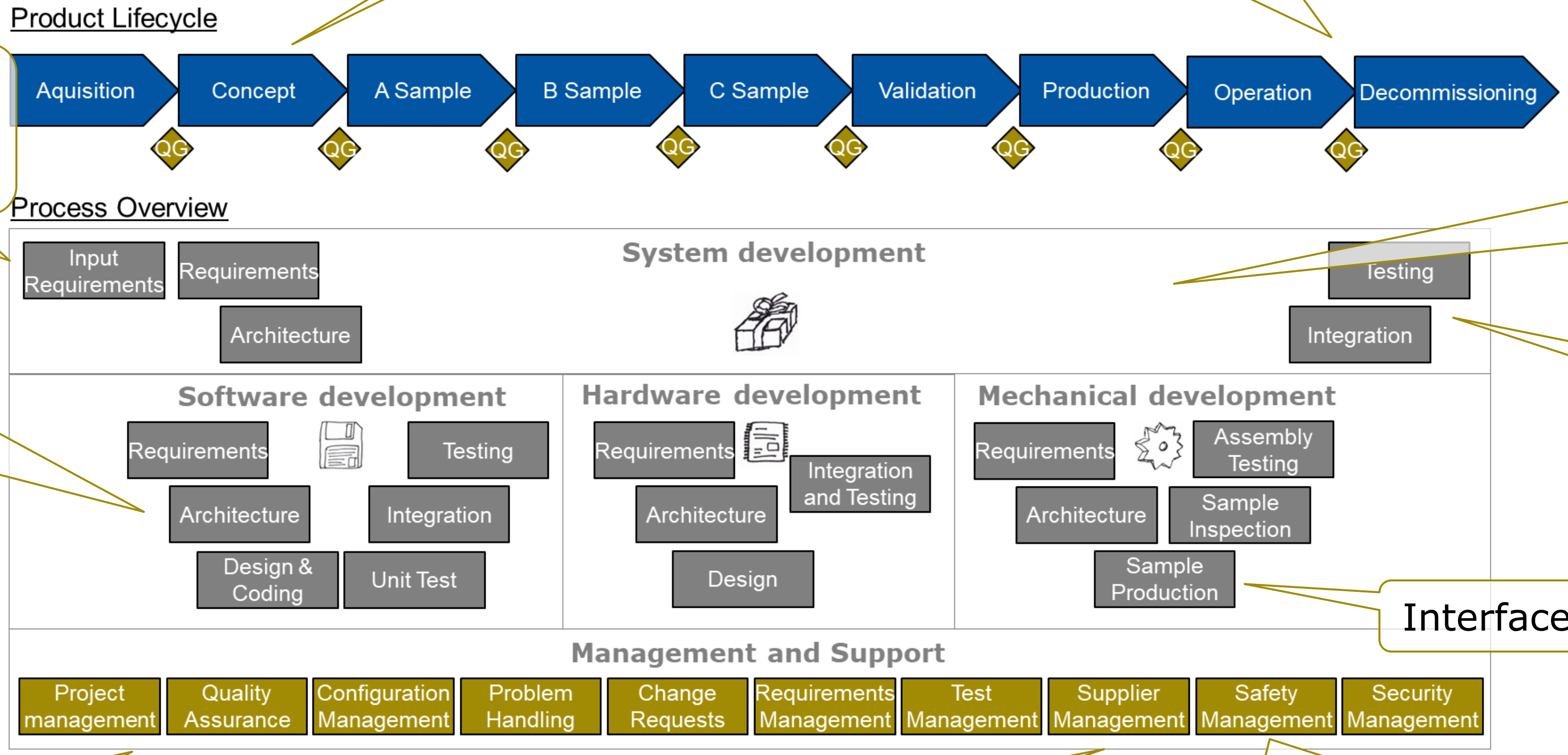
Mapping of Security Lifecycle

TARA

Post-development support

Cyber Security

Security Goals, Cybersecurity Concept



Security activities (Security requirements, security analysis, etc.)

Additional Cybersecurity controls

Penetration testing

Interface to production

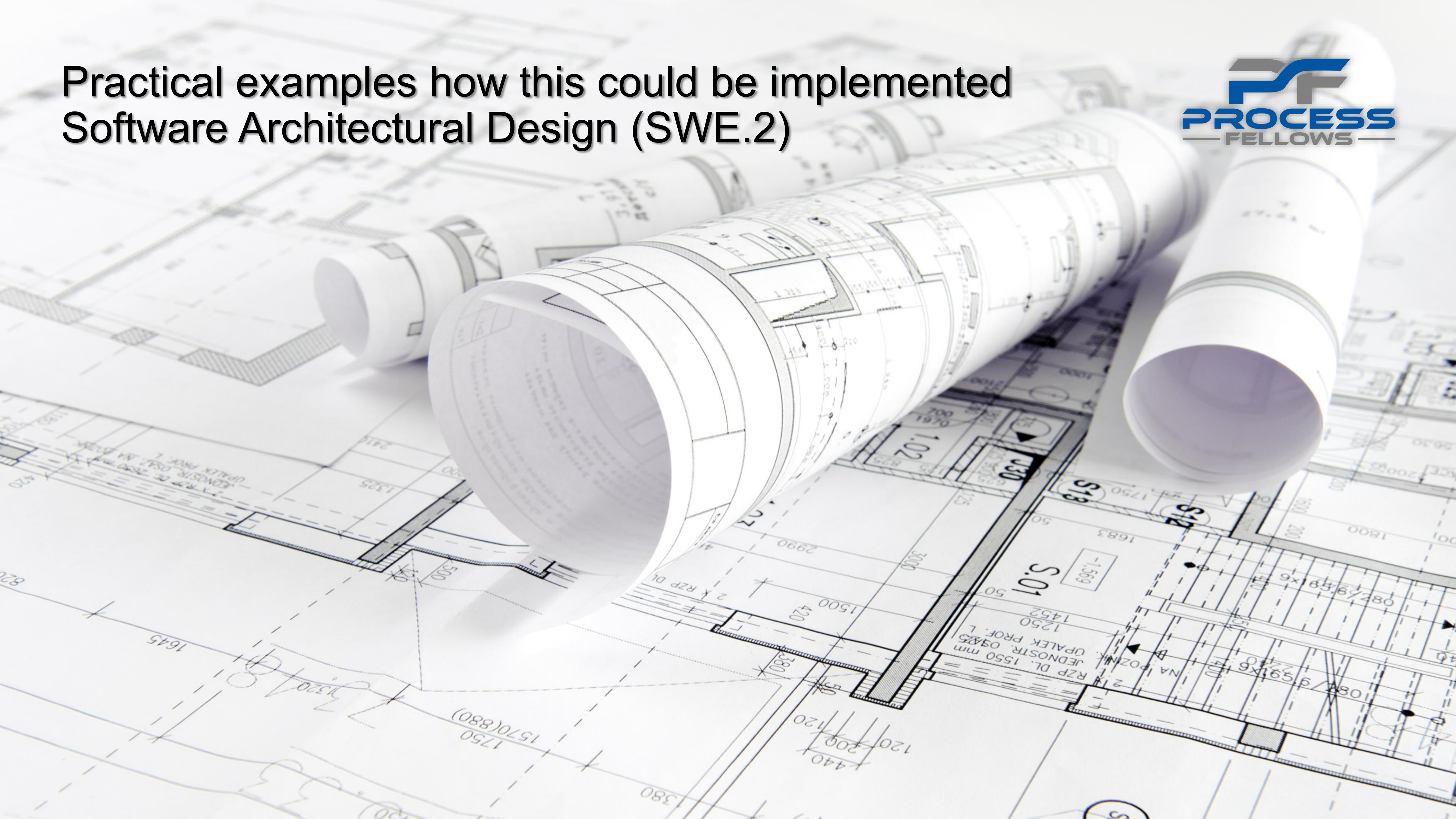
Supportive role: Security Manager

Supplier evaluation regarding CS capabilities

Additional process: Security Management (Planning, Tracking and Security Assessments)

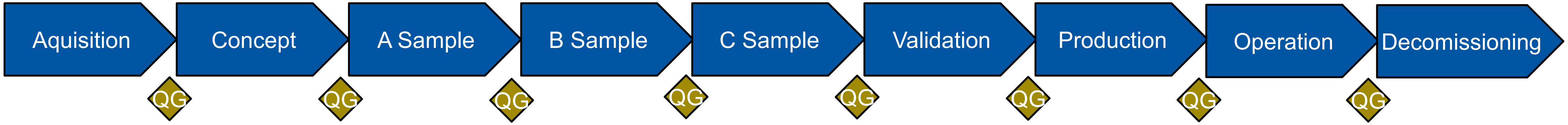
Practical examples how this could be implemented Software Architectural Design (SWE.2)

PROCESS FELLOWS



Software Architectural Design (SWE.2)

Product Lifecycle



Aquisition

QG

Concept

QG

A Sample

QG

B Sample

QG

C Sample

QG

Validation

QG

Production

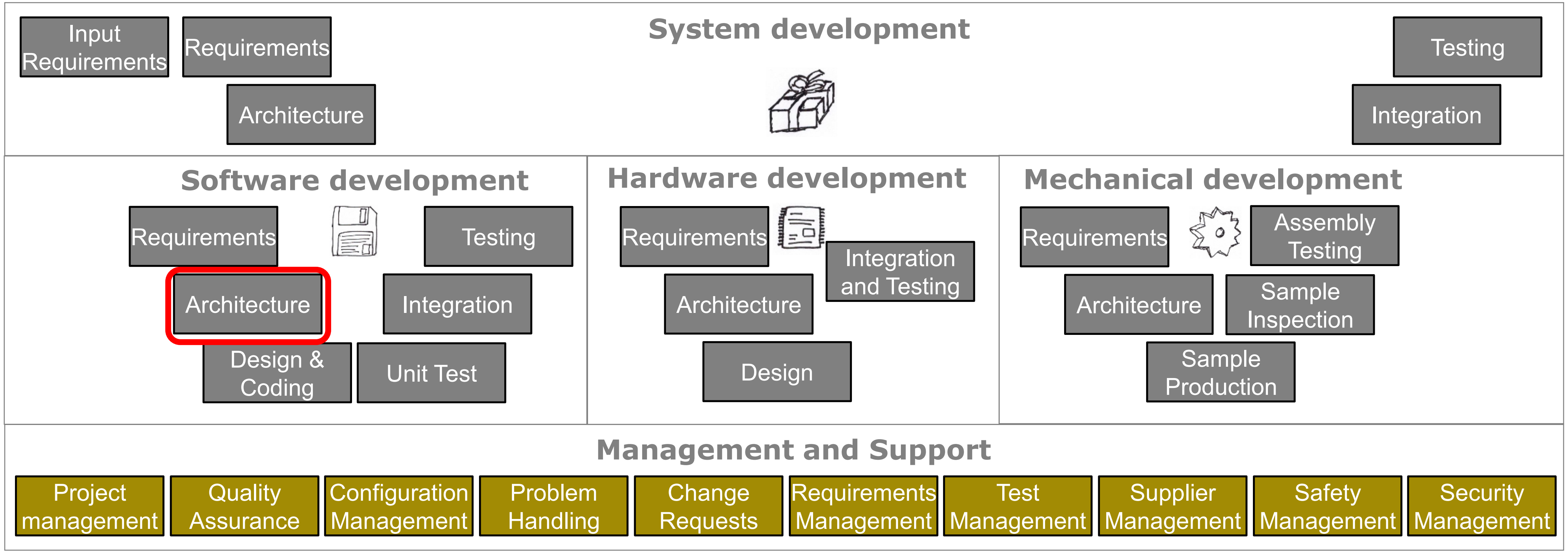
QG

Operation

QG

Decomissioning

Process Overview



System development

Input Requirements

Requirements

Architecture

Testing

Integration

Software development

Requirements

Testing

Architecture

Integration

Design & Coding

Unit Test

Hardware development

Requirements

Integration and Testing

Architecture

Design

Mechanical development

Requirements

Assembly Testing

Architecture

Sample Inspection

Sample Production

Management and Support

Project management

Quality Assurance

Configuration Management

Problem Handling

Change Requests

Requirements Management

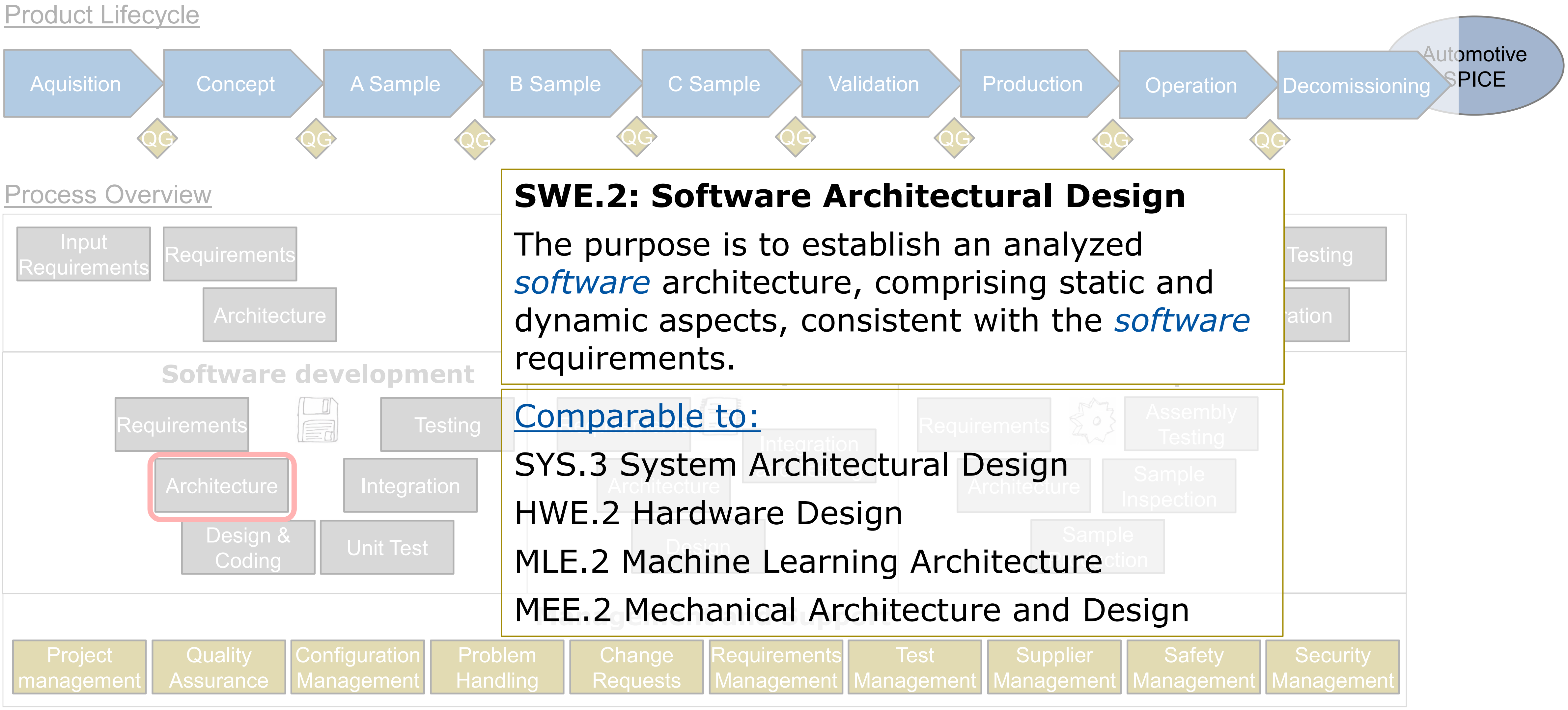
Test Management

Supplier Management

Safety Management

Security Management

Software Architectural Design (SWE.2)



Product Lifecycle

Aquisition
Concept
A Sample
B Sample
C Sample
Validation
Production
Operation
Decomissioning

QG QG QG QG QG QG QG QG

Automotive SPICE

Process Overview

Input Requirements
Requirements
Architecture
Testing
ation

Software development

Requirements
Testing
Architecture
Integration
Design & Coding
Unit Test

Requirements
Assembly Testing
Architecture
Sample Inspection
Sample

Project management
Quality Assurance
Configuration Management
Problem Handling
Change Requests
Requirements Management
Test Management
Supplier Management
Safety Management
Security Management

SWE.2: Software Architectural Design

The purpose is to establish an analyzed *software* architecture, comprising static and dynamic aspects, consistent with the *software* requirements.

Comparable to:

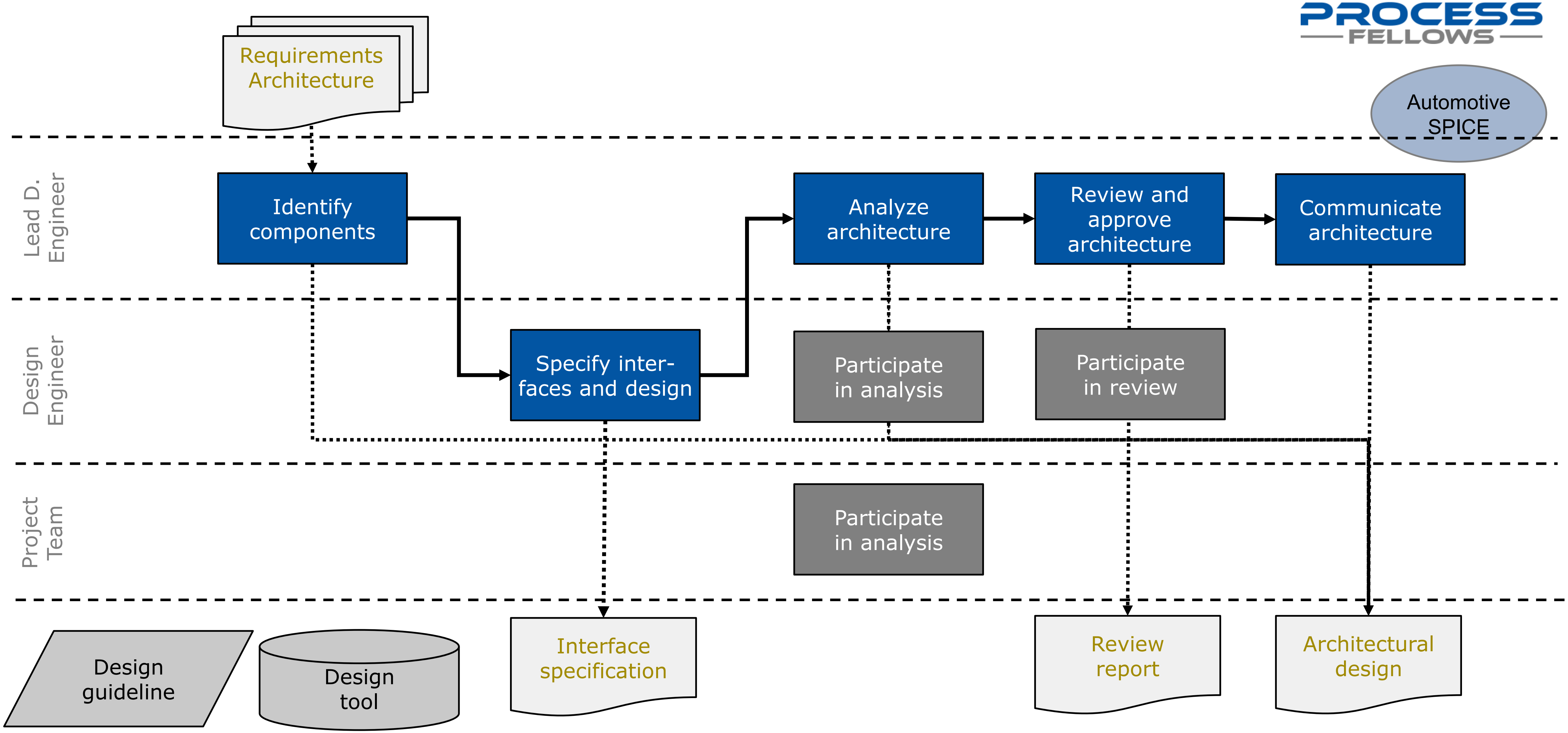
SYS.3 System Architectural Design

HWE.2 Hardware Design

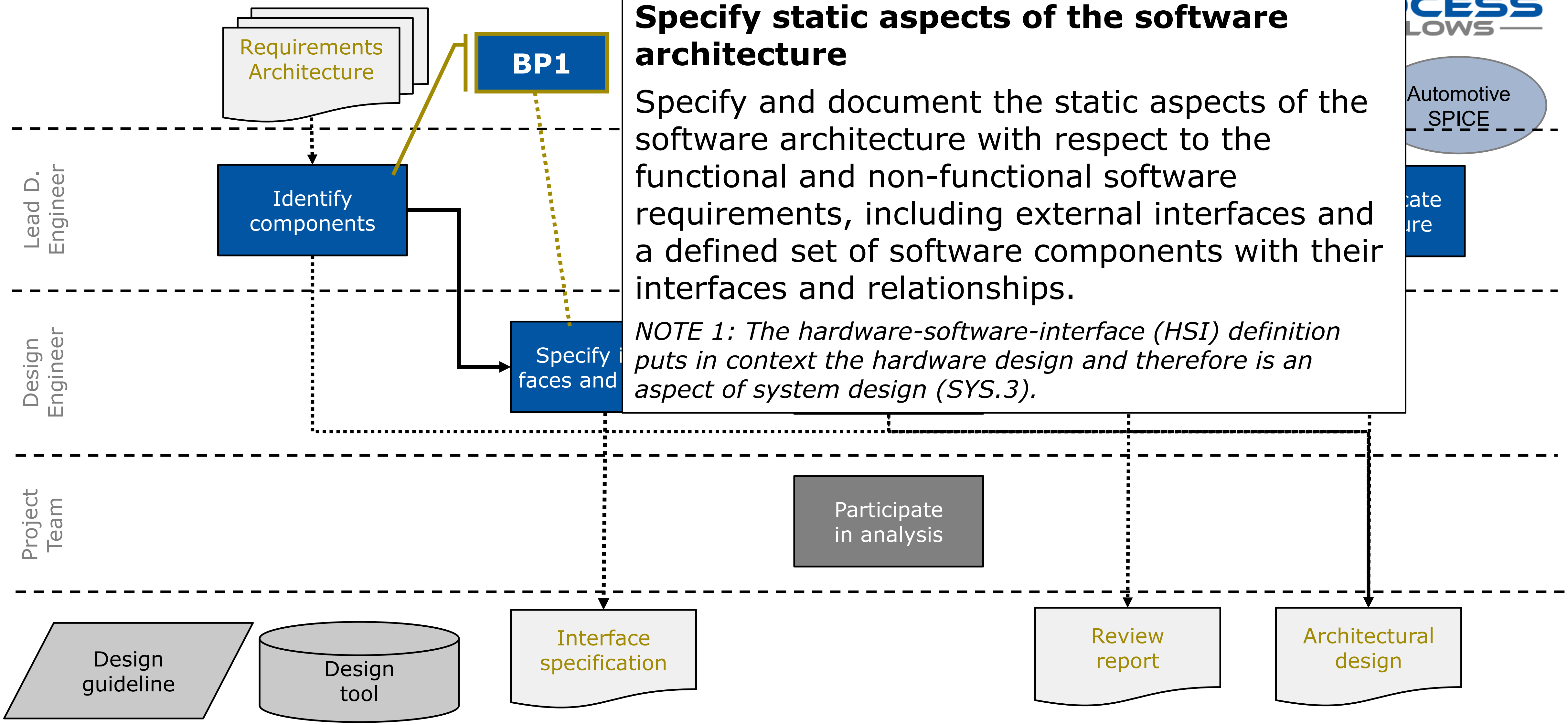
MLE.2 Machine Learning Architecture

MEE.2 Mechanical Architecture and Design

Software Architectural Design (SWE.2)



Software Architectural Design (SWE.2)



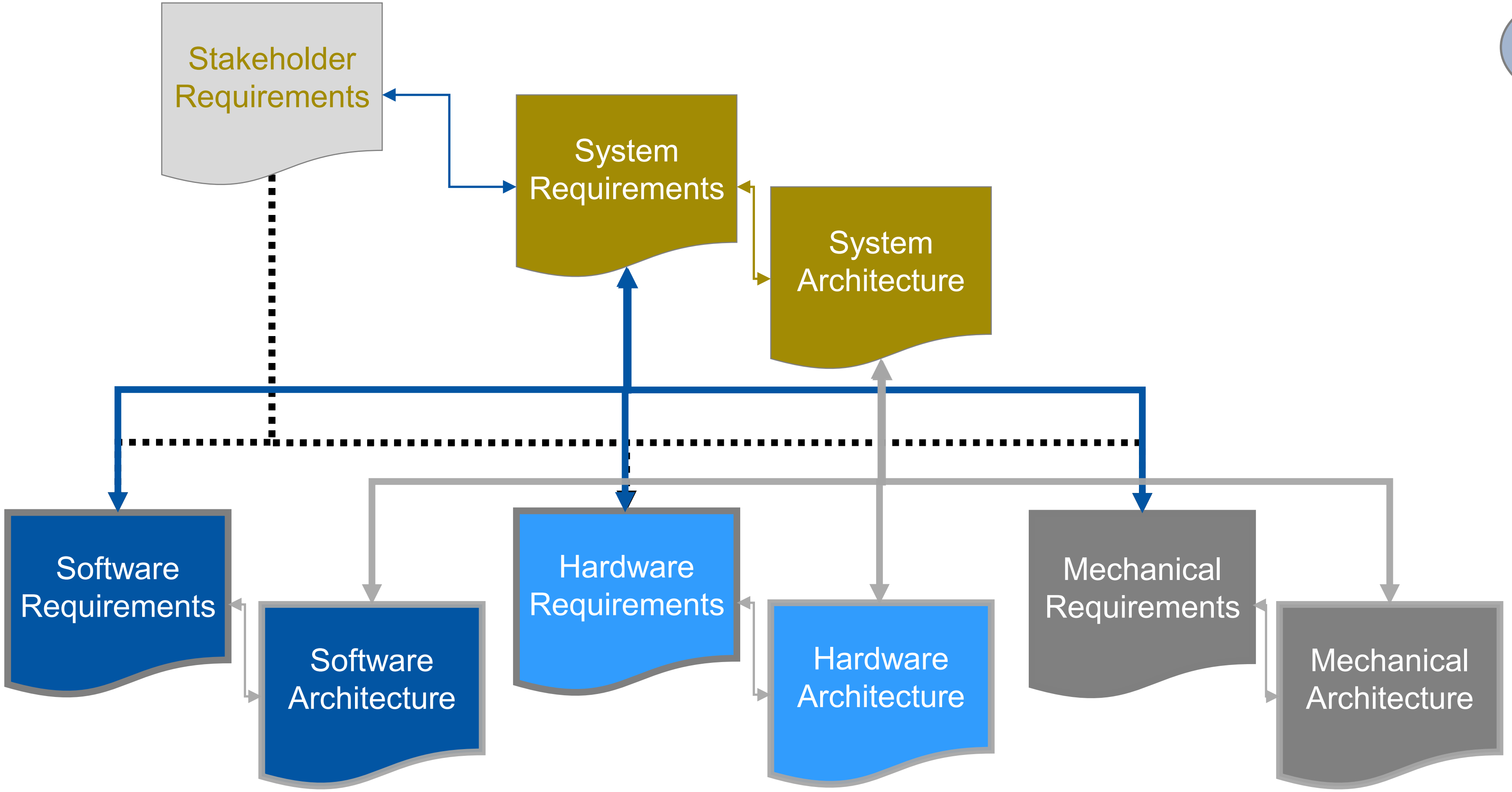
Specify static aspects of the software architecture

Specify and document the static aspects of the software architecture with respect to the functional and non-functional software requirements, including external interfaces and a defined set of software components with their interfaces and relationships.

NOTE 1: The hardware-software-interface (HSI) definition puts in context the hardware design and therefore is an aspect of system design (SYS.3).

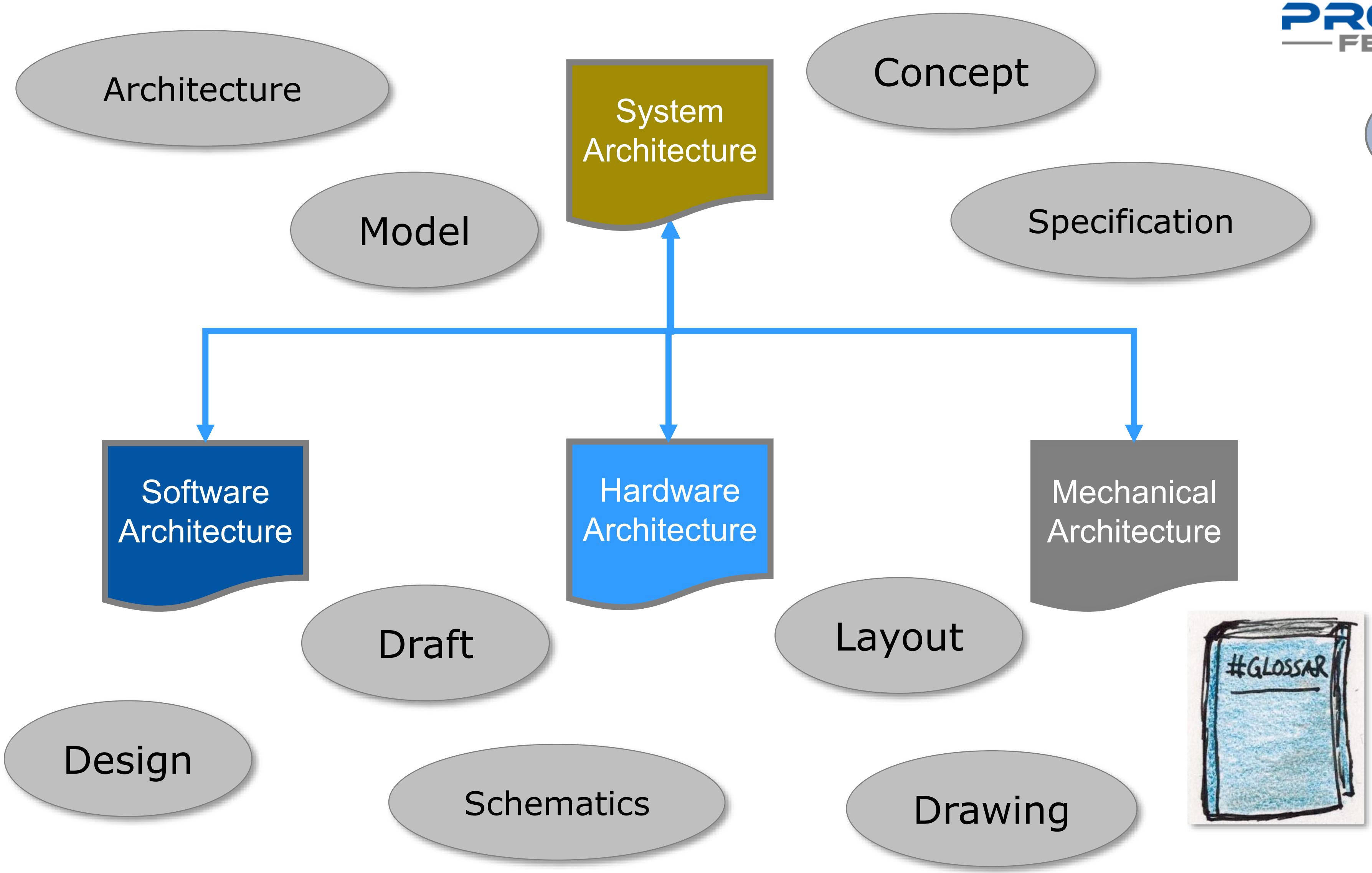
What's an architecture in Automotive SPICE®?

Automotive SPICE

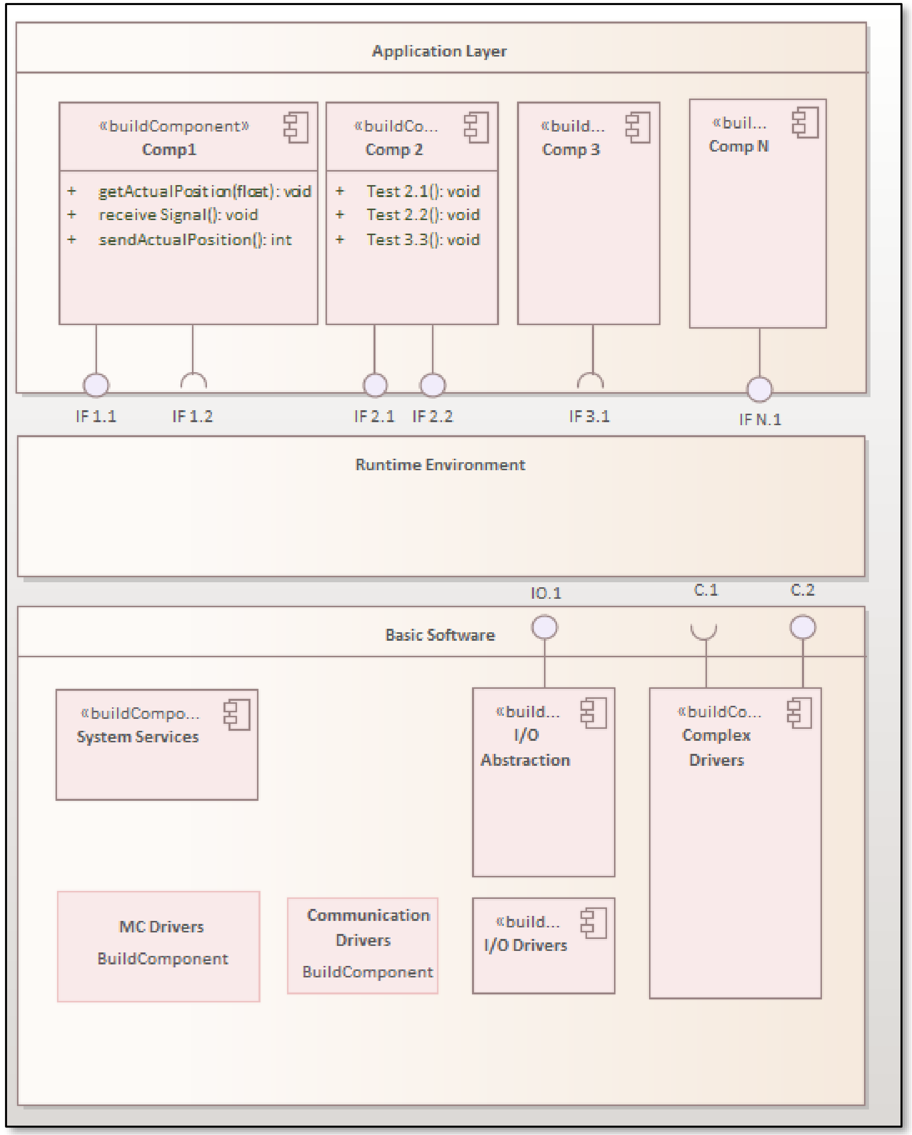


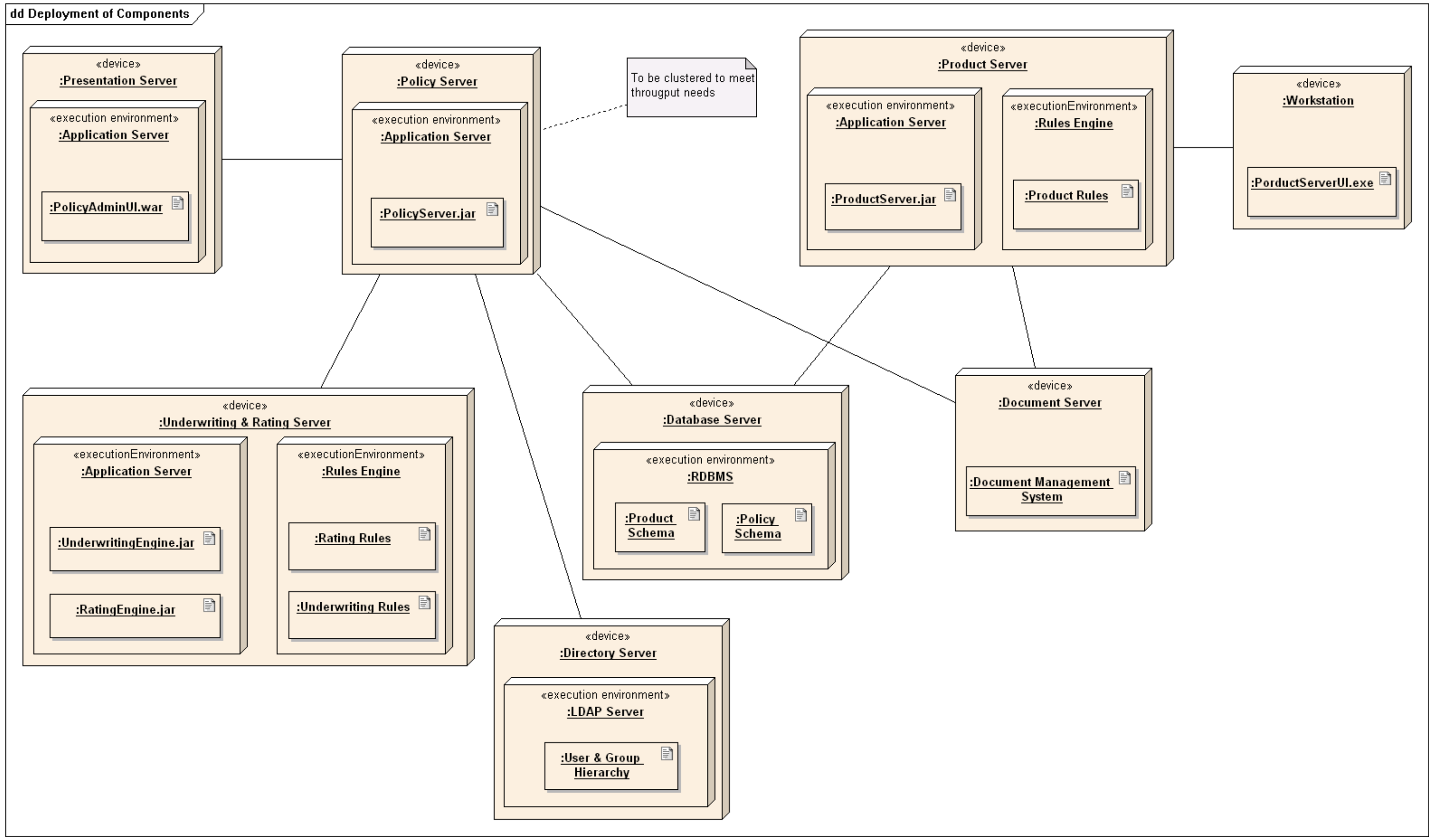
…and in reality?

Automotive SPICE



Software Architecture with UML



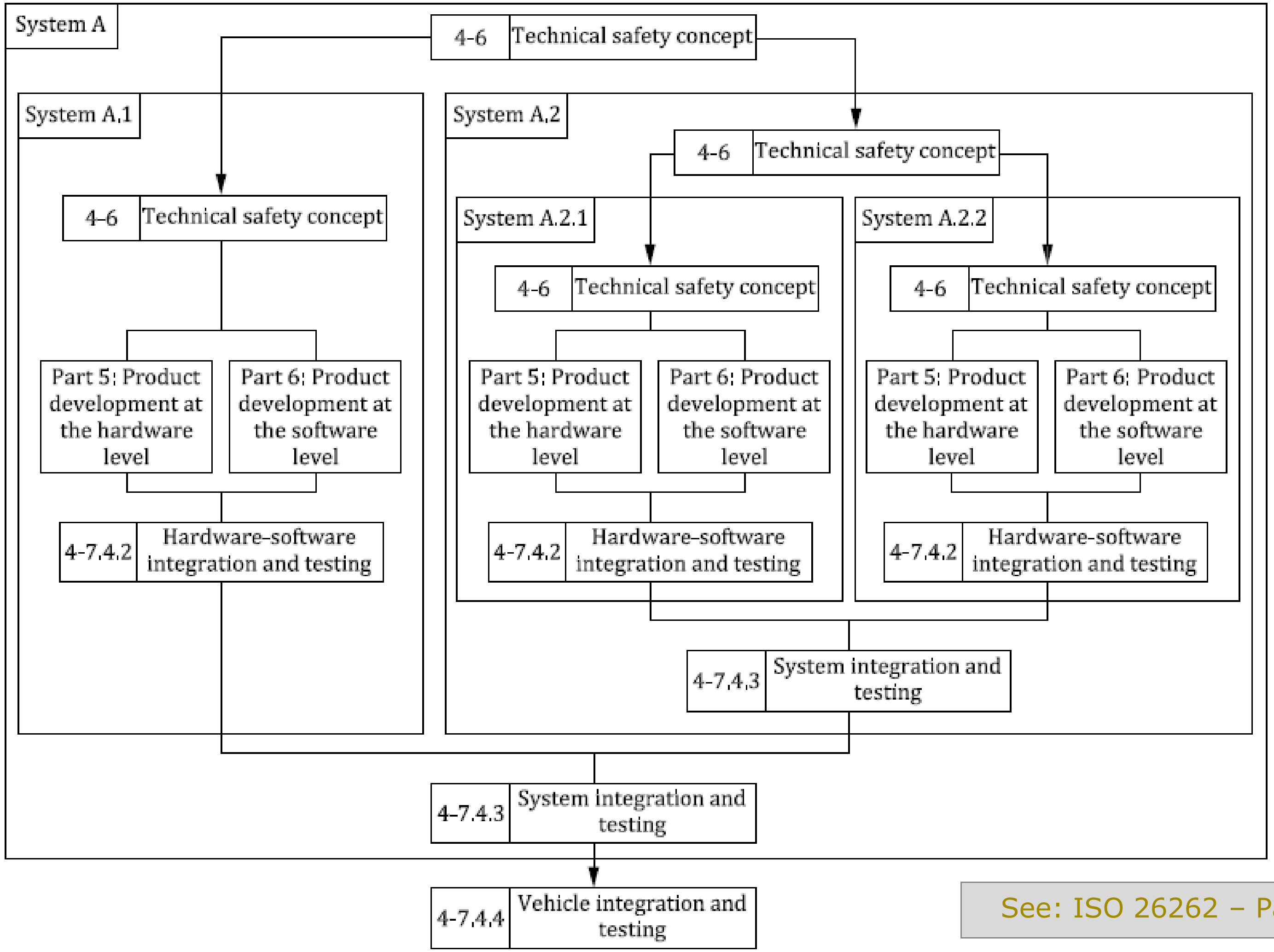


Reference: Deployment diagram,
Wikipedia: https://en.wikipedia.org/wiki/Component_diagram

What is a Technical Safety Concept?

Specification of the **technical safety requirements** and their allocation to **system elements** with associated information providing a rationale for functional safety at the system level:

- Technical safety requirements (incl. safety mechanisms)
- System architectural design specification
- Allocation to hardware and software
- Development of requirements and architecture on hardware and software level



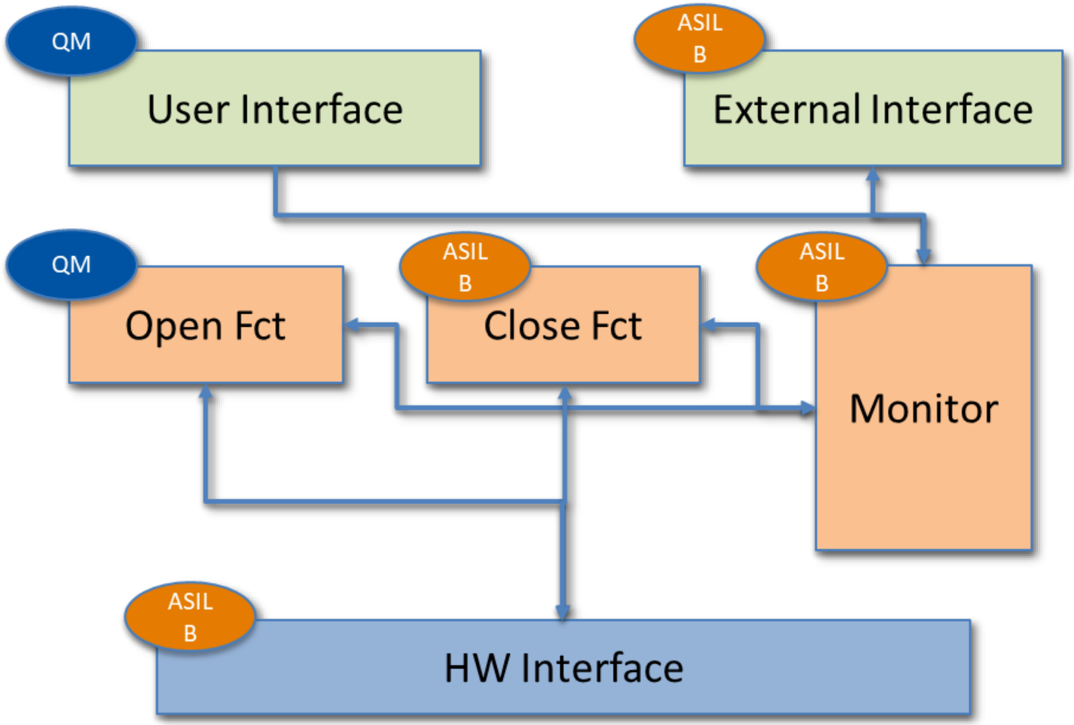
System A
4-6 Technical safety concept
System A.1
4-6 Technical safety concept
Part 5: Product development at the hardware level
Part 6: Product development at the software level
4-7.4.2 Hardware-software integration and testing
System A.2
4-6 Technical safety concept
System A.2.1
4-6 Technical safety concept
Part 5: Product development at the hardware level
Part 6: Product development at the software level
4-7.4.2 Hardware-software integration and testing
System A.2.2
4-6 Technical safety concept
Part 5: Product development at the hardware level
Part 6: Product development at the software level
4-7.4.2 Hardware-software integration and testing
4-7.4.3 System integration and testing
4-7.4.3 System integration and testing
4-7.4.4 Vehicle integration and testing

See: ISO 26262 – Part 4

Safety classifications of architectural elements

Functional Safety

Elements of the architecture can be assigned with criticality levels
→ ASIL (Automotive Safety Integrity Level)



What is the HSI (Hardware-Software Interface)?

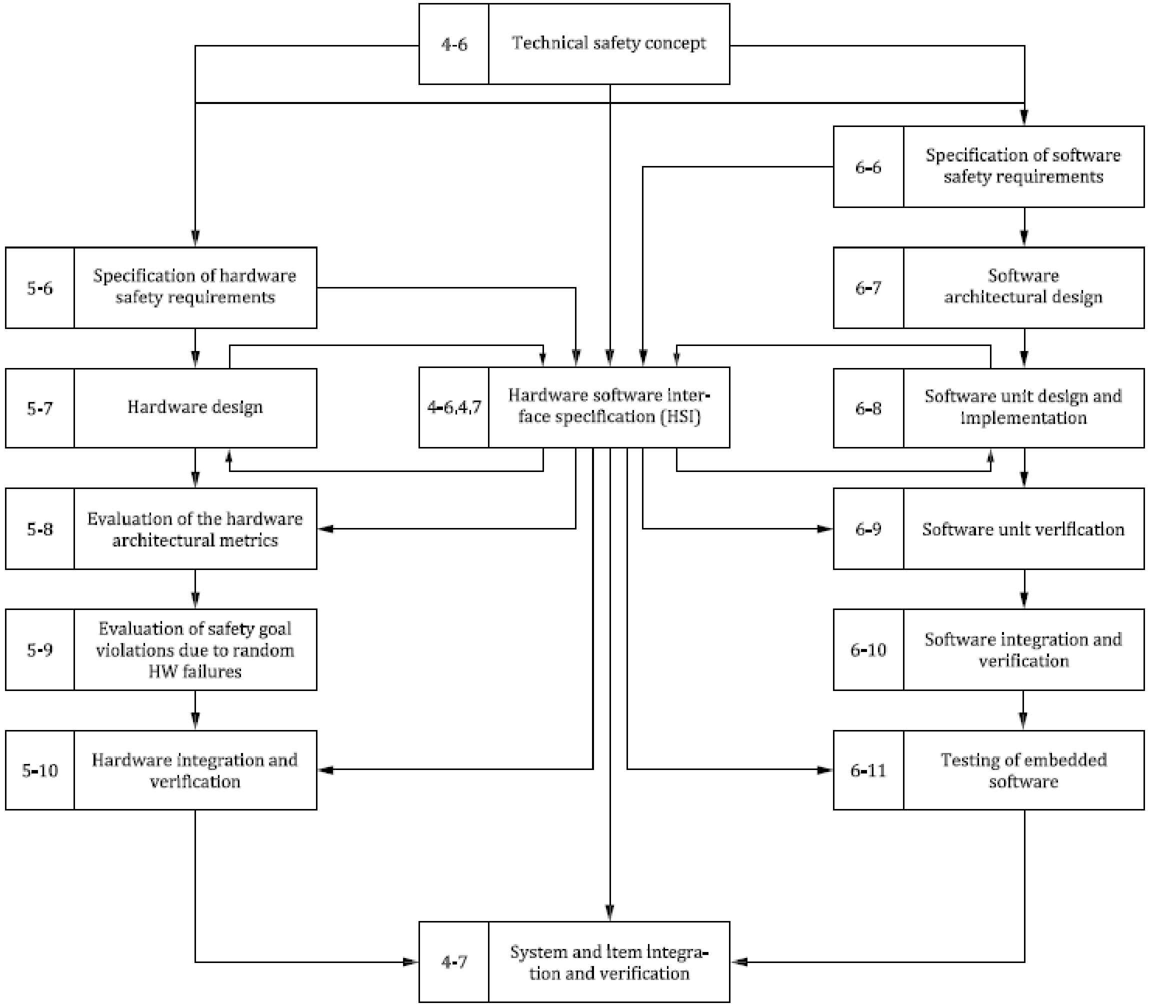
Functional Safety

HSI elements

- Memory
- Bus interfaces
- Converter
- Multiplexer
- Electrical I/O
- Watchdog

HSI characteristics

- Interrupts
- Timing consistency
- Data integrity
- Initialization
- Message transfer
- Network modes
- Memory management
- Real-time counter


4-6 Technical safety concept
6-6 Specification of software safety requirements
5-6 Specification of hardware safety requirements
6-7 Software architectural design
5-7 Hardware design
4-6,4,7 Hardware software interface specification (HSI)
6-8 Software unit design and implementation
5-8 Evaluation of the hardware architectural metrics
6-9 Software unit verification
5-9 Evaluation of safety goal violations due to random HW failures
6-10 Software integration and verification
5-10 Hardware integration and verification
6-11 Testing of embedded software
4-7 System and item integration and verification

See: ISO 26262 – Part 4

ASIL-dependent methods

Functional Safety

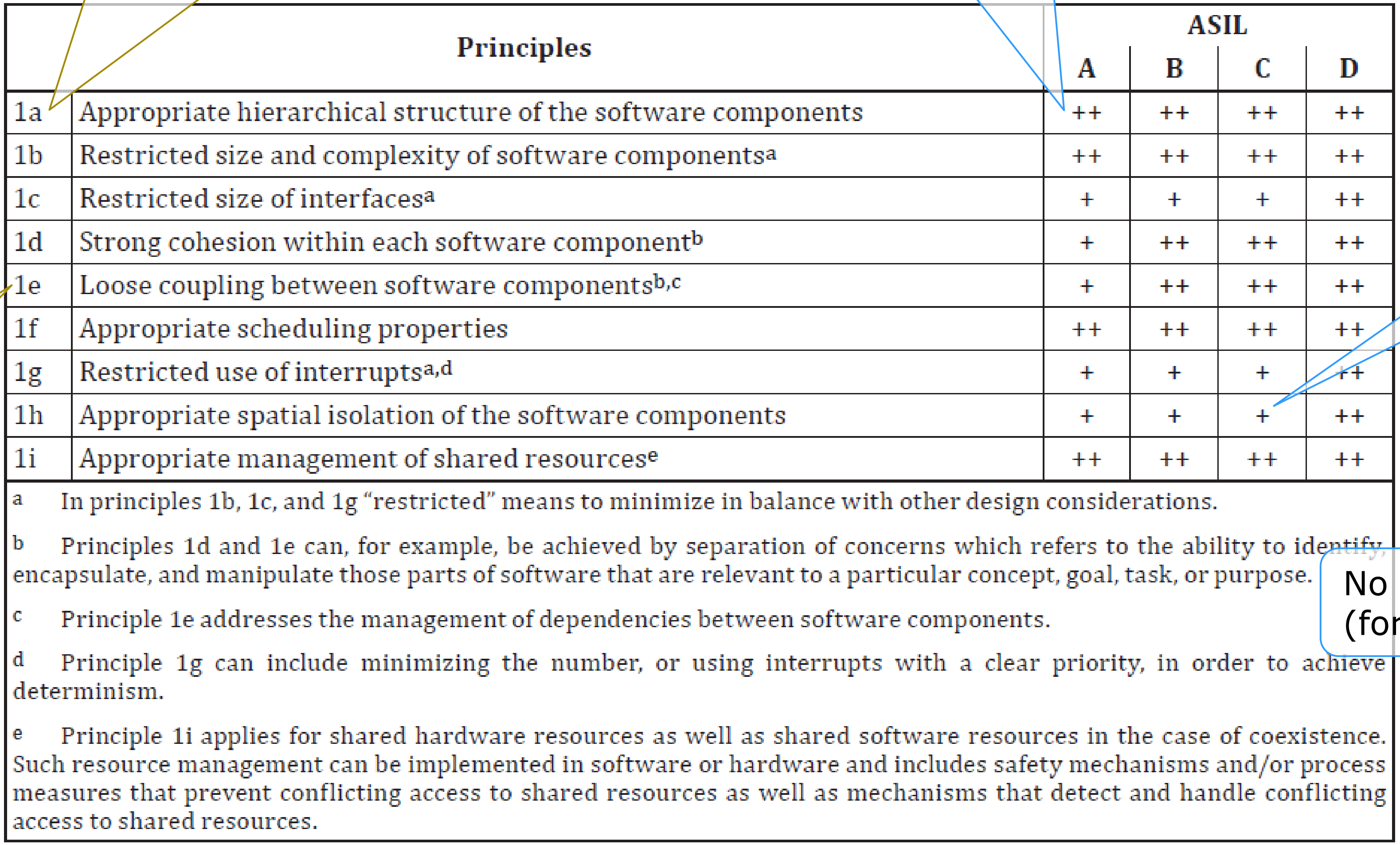
alternative entries (a,b,c,…)

Highly recommended (++)

Recommended (+)

No recommendation (for or against) (o)

consecutive entries (1,2,3,…)



	Principles	ASIL			
		A	B	C	D
1a	Appropriate hierarchical structure of the software components	++	++	++	++
1b	Restricted size and complexity of software components[a]	++	++	++	++
1c	Restricted size of interfaces[a]	+	+	+	++
1d	Strong cohesion within each software component[b]	+	++	++	++
1e	Loose coupling between software components[b,c]	+	++	++	++
1f	Appropriate scheduling properties	++	++	++	++
1g	Restricted use of interrupts[a,d]	+	+	+	++
1h	Appropriate spatial isolation of the software components	+	+	+	++
1i	Appropriate management of shared resources[e]	++	++	++	++

[a] In principles 1b, 1c, and 1g "restricted" means to minimize in balance with other design considerations.

[b] Principles 1d and 1e can, for example, be achieved by separation of concerns which refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concept, goal, task, or purpose.

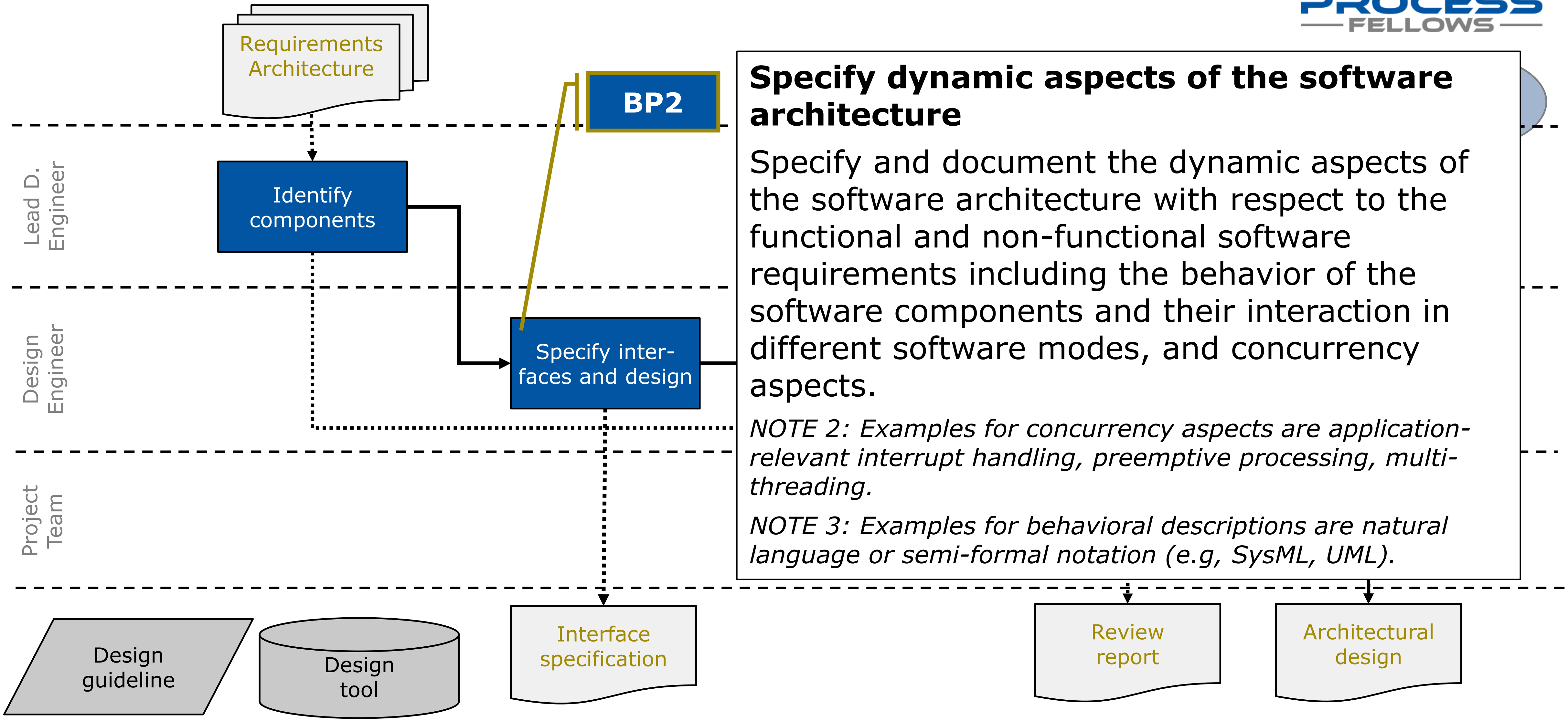
[c] Principle 1e addresses the management of dependencies between software components.

[d] Principle 1g can include minimizing the number, or using interrupts with a clear priority, in order to achieve determinism.

[e] Principle 1i applies for shared hardware resources as well as shared software resources in the case of coexistence. Such resource management can be implemented in software or hardware and includes safety mechanisms and/or process measures that prevent conflicting access to shared resources as well as mechanisms that detect and handle conflicting access to shared resources.

See: ISO 26262 – Part 2

Software Architectural Design (SWE.2)



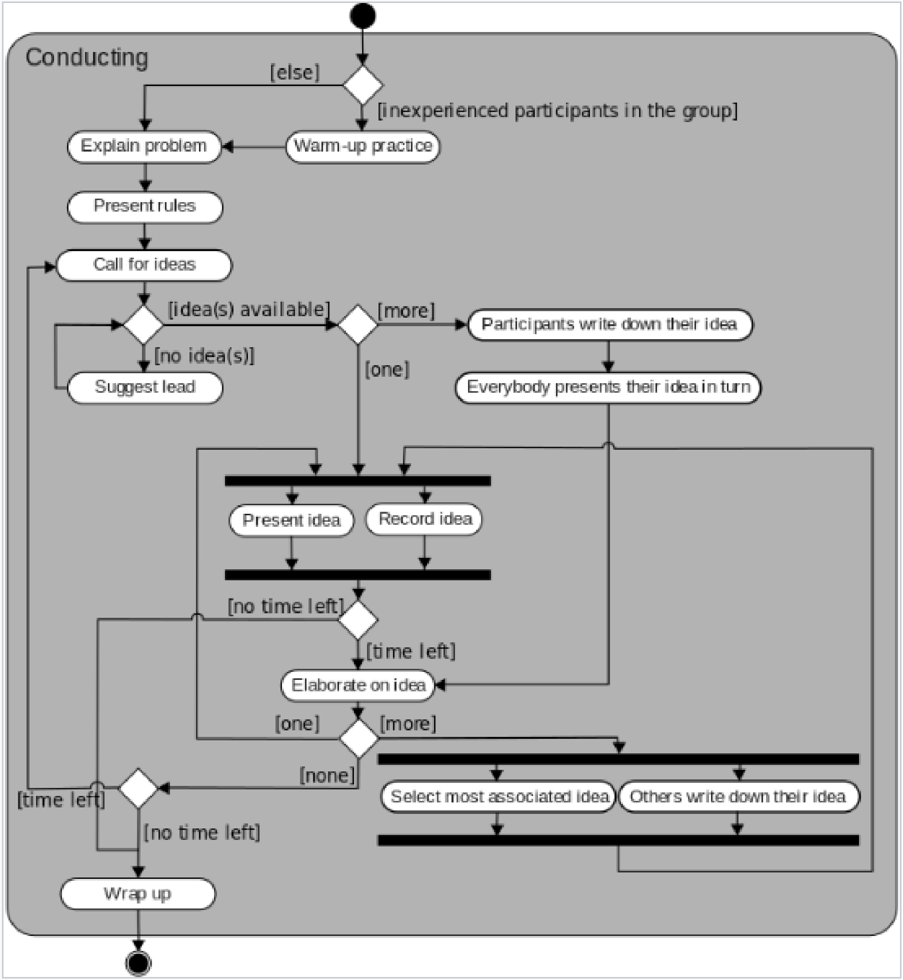
Specify dynamic aspects of the software architecture

Specify and document the dynamic aspects of the software architecture with respect to the functional and non-functional software requirements including the behavior of the software components and their interaction in different software modes, and concurrency aspects.

NOTE 2: Examples for concurrency aspects are application-relevant interrupt handling, preemptive processing, multi-threading.

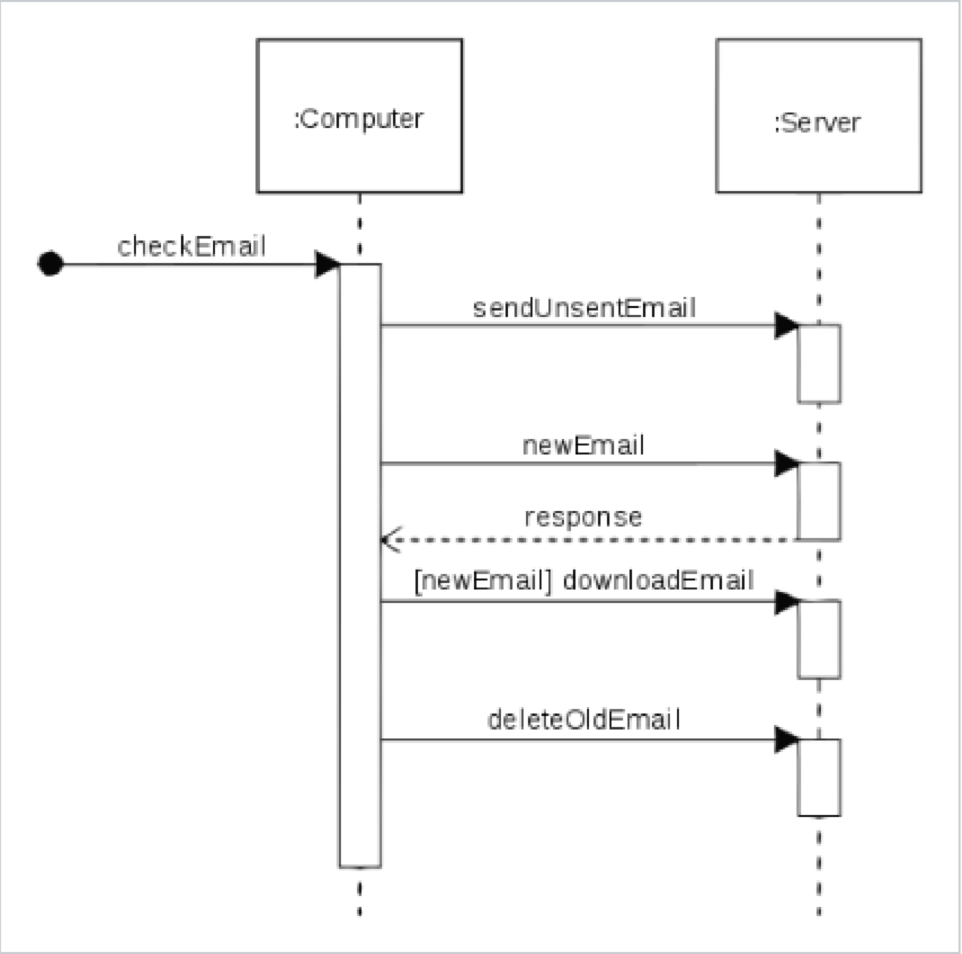
NOTE 3: Examples for behavioral descriptions are natural language or semi-formal notation (e.g, SysML, UML).

Example: Dynamic View – Activity / Sequence Diagram



UML 1.x Activity diagram for a guided brainstorming process.

Source: Wikipedia



The Sequence diagram of UML

Source: Wikipedia

Example: Dynamic View – State Machine Diagram

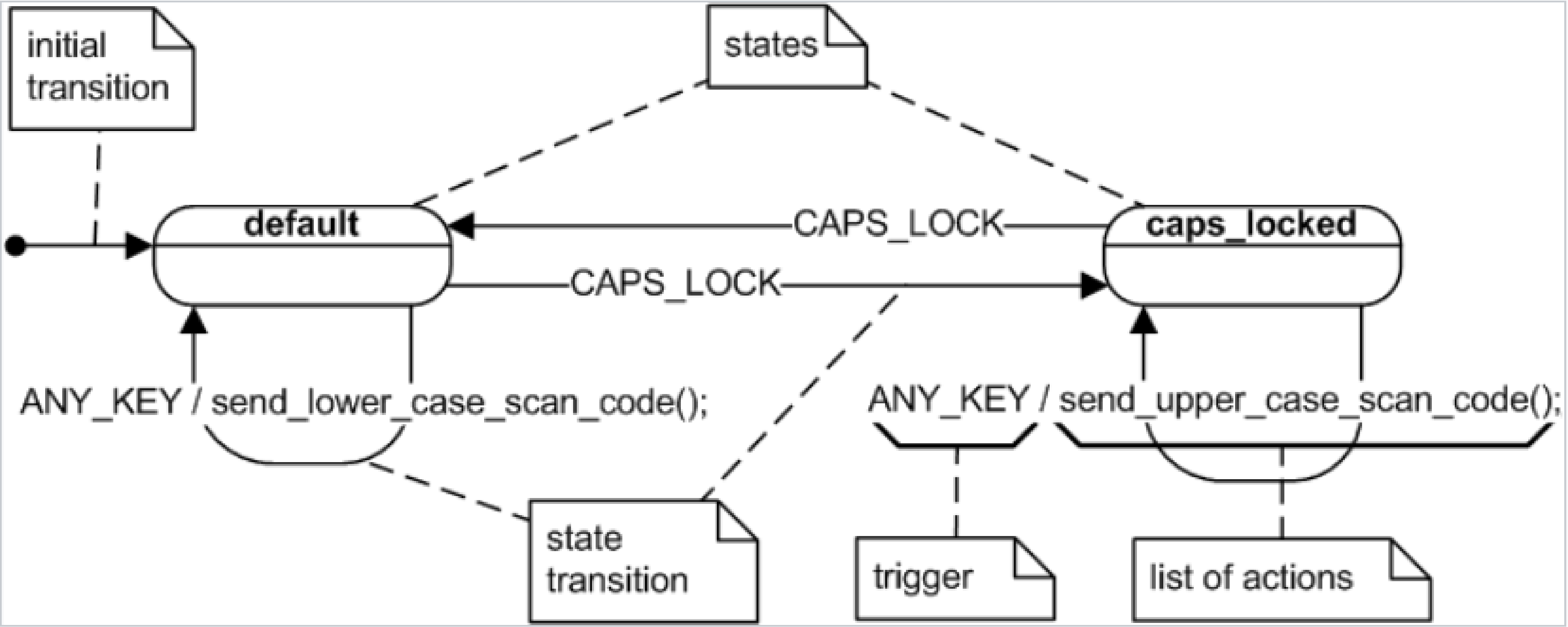
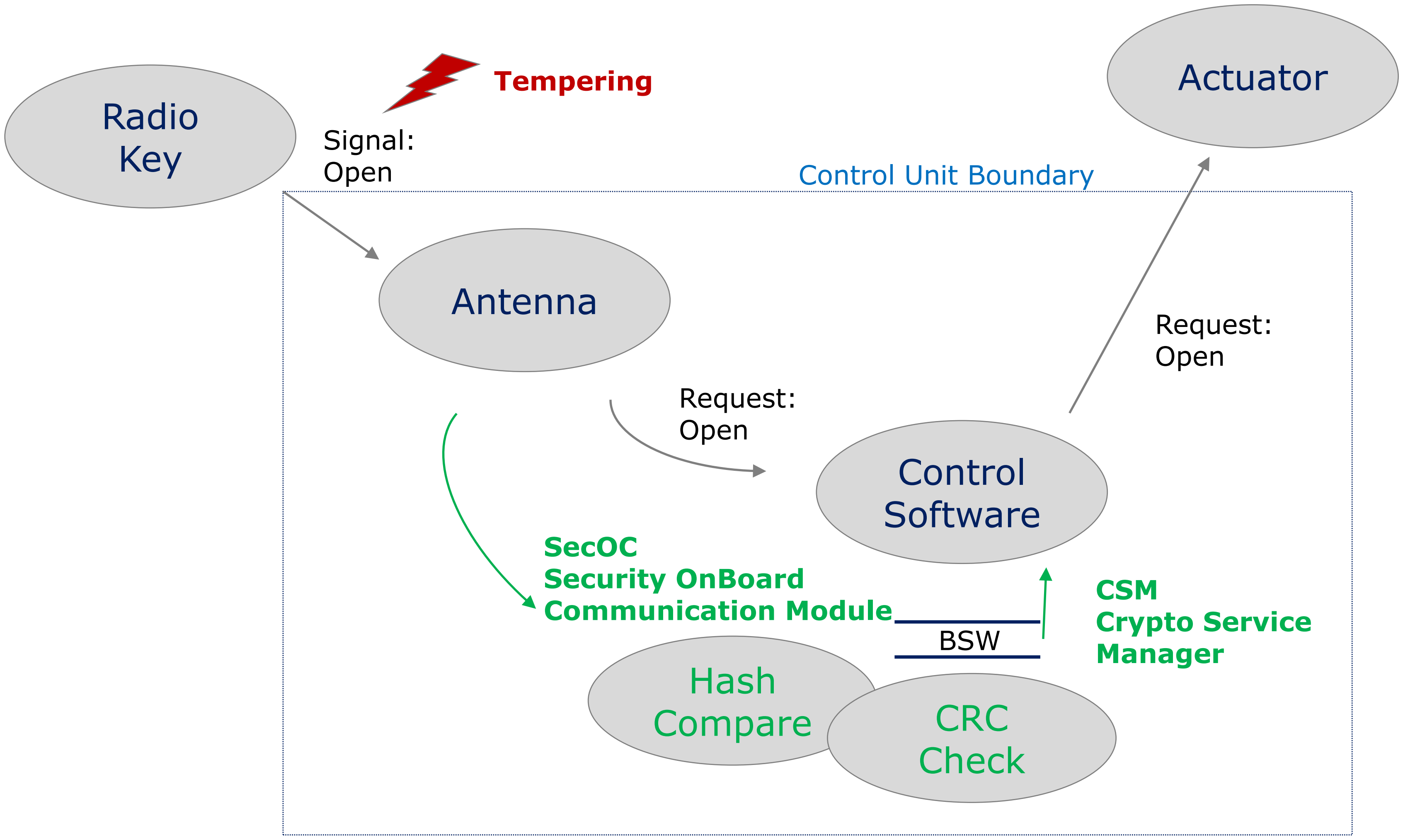


Figure 1: UML state diagram representing the computer keyboard state machine

Source: Wikipedia

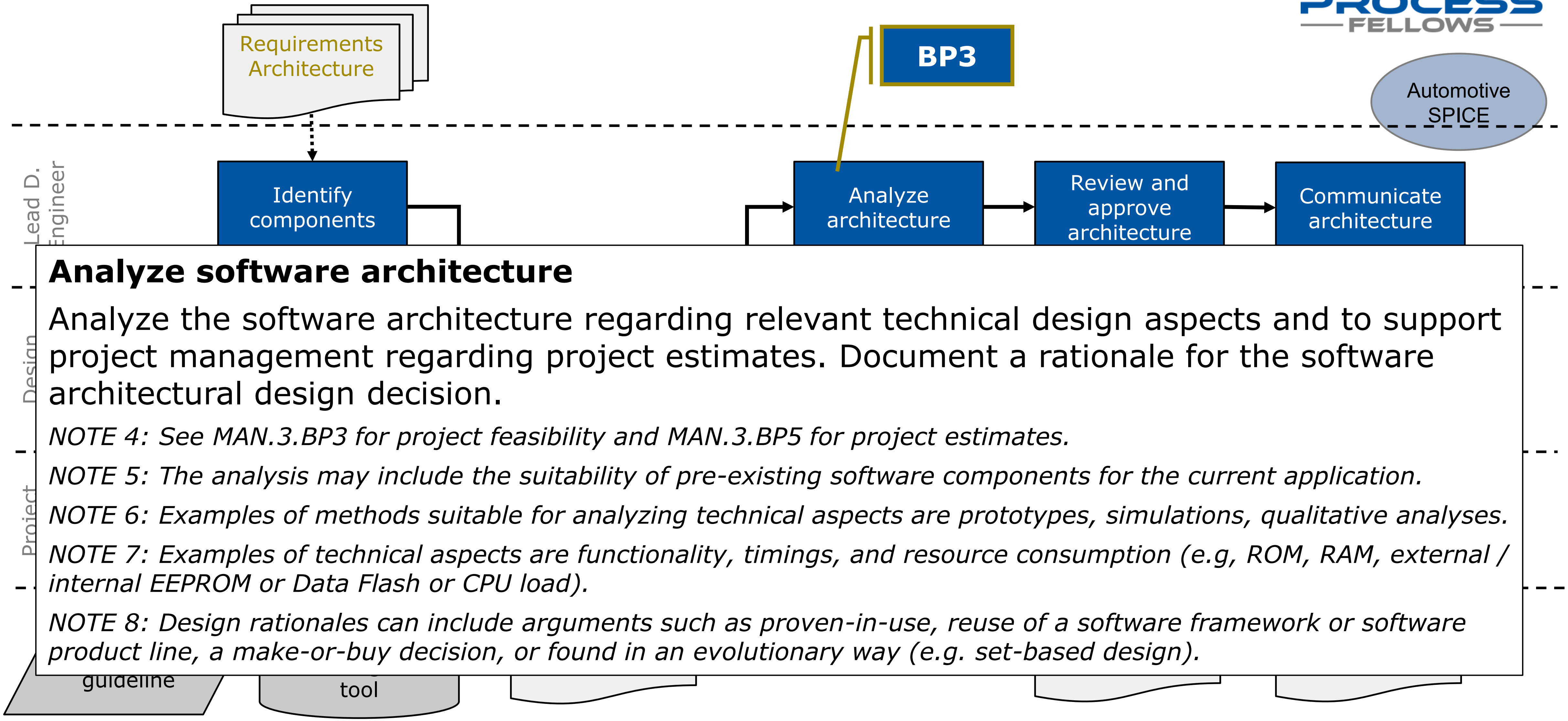
Threat Modelling

Cyber Security



- Cybersecurity will add security-specific mechanisms such as zones of trust, interface protection, secure data storage, firewalls, sand-boxes, etc.
- Interfaces might be allocated to threats, used as input for verification.
- Threat models might be used as an additional architectural view.

Software Architectural Design (SWE.2)



Analyze software architecture

Analyze the software architecture regarding relevant technical design aspects and to support project management regarding project estimates. Document a rationale for the software architectural design decision.

NOTE 4: See MAN.3.BP3 for project feasibility and MAN.3.BP5 for project estimates.

NOTE 5: The analysis may include the suitability of pre-existing software components for the current application.

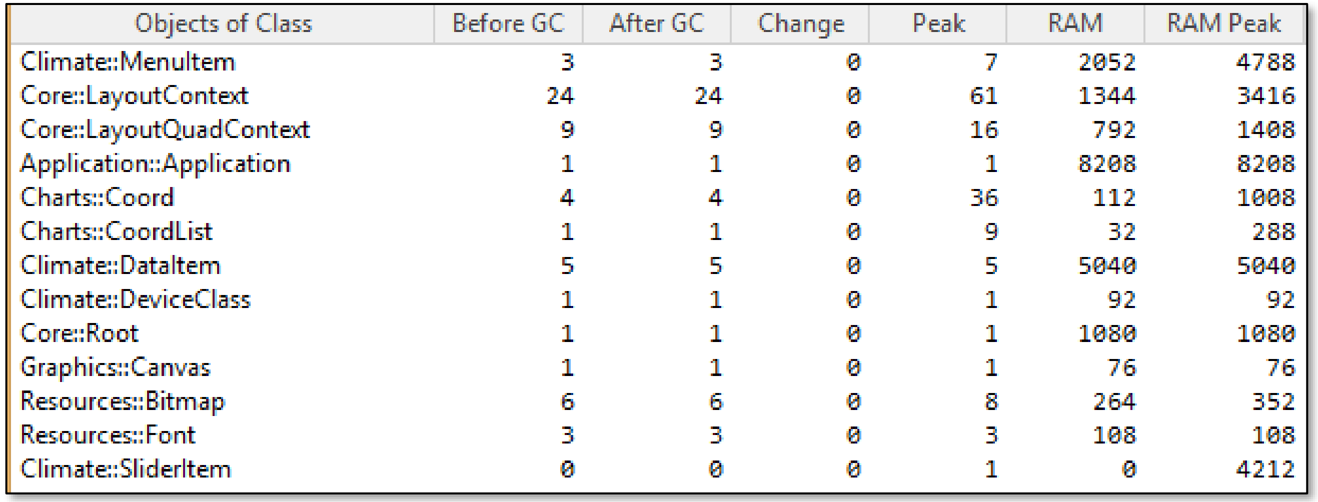
NOTE 6: Examples of methods suitable for analyzing technical aspects are prototypes, simulations, qualitative analyses.

NOTE 7: Examples of technical aspects are functionality, timings, and resource consumption (e.g, ROM, RAM, external / internal EEPROM or Data Flash or CPU load).

NOTE 8: Design rationales can include arguments such as proven-in-use, reuse of a software framework or software product line, a make-or-buy decision, or found in an evolutionary way (e.g. set-based design).

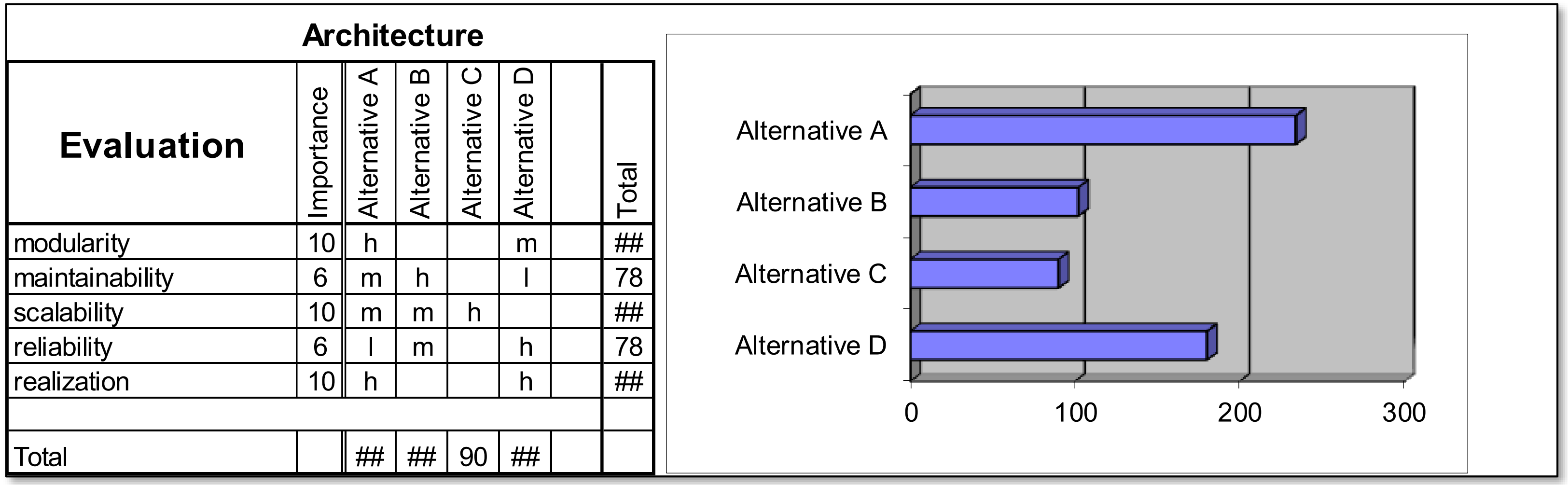
Some practical examples

Resource objectives



Objects of Class	Before GC	After GC	Change	Peak	RAM	RAM Peak
Climate::MenuItem	3	3	0	7	2052	4788
Core::LayoutContext	24	24	0	61	1344	3416
Core::LayoutQuadContext	9	9	0	16	792	1408
Application::Application	1	1	0	1	8208	8208
Charts::Coord	4	4	0	36	112	1008
Charts::CoordList	1	1	0	9	32	288
Climate::DataItem	5	5	0	5	5040	5040
Climate::DeviceClass	1	1	0	1	92	92
Core::Root	1	1	0	1	1080	1080
Graphics::Canvas	1	1	0	1	76	76
Resources::Bitmap	6	6	0	8	264	352
Resources::Font	3	3	0	3	108	108
Climate::SliderItem	0	0	0	1	0	4212

Design decision

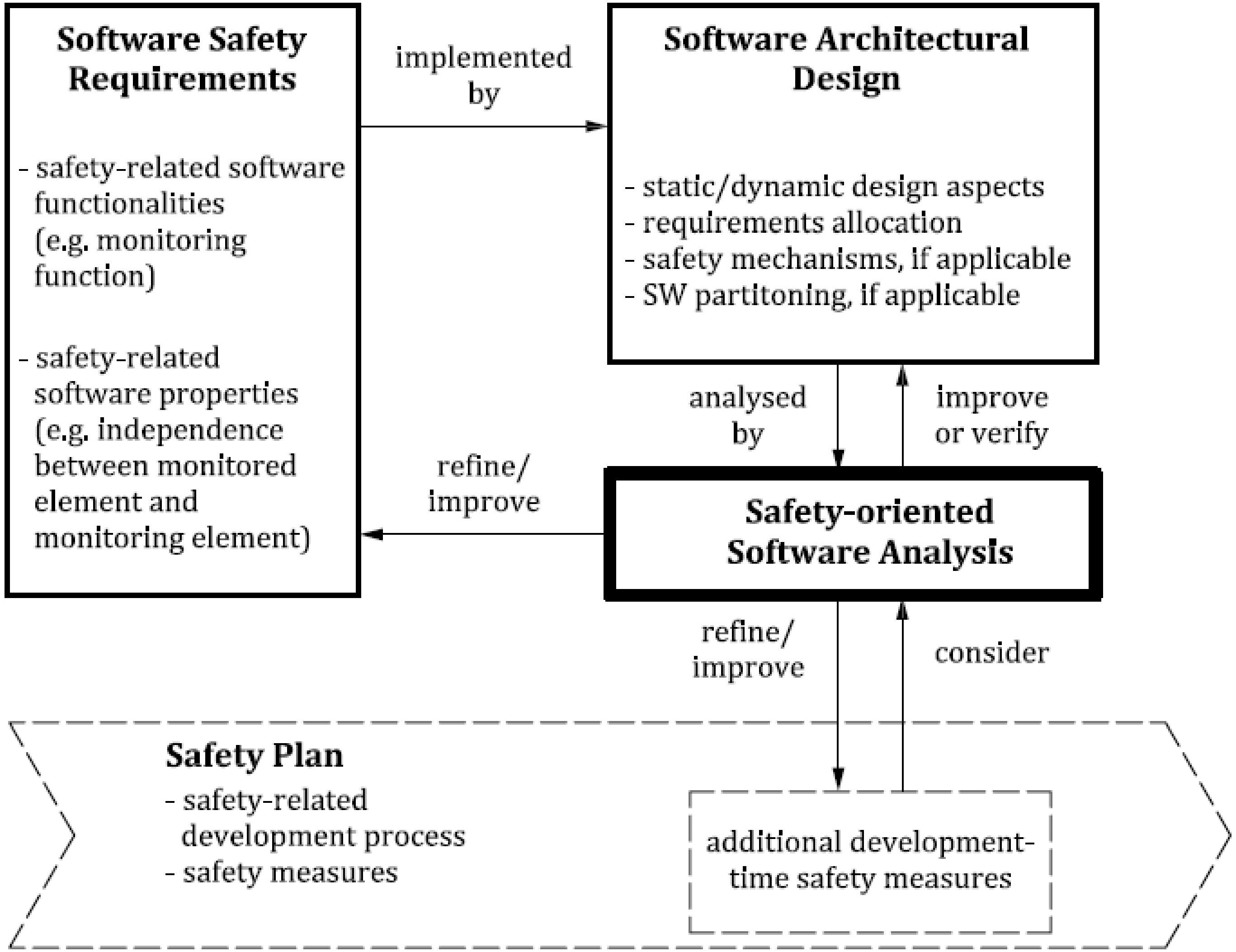


Architecture

Evaluation	Importance	Alternative A	Alternative B	Alternative C	Alternative D		Total
modularity	10	h			m		##
maintainability	6	m	h		l		78
scalability	10	m	m	h			##
reliability	6	l	m		h		78
realization	10	h			h		##
Total		##	##	90	##		

Safety-oriented Software Analysis

Functional Safety



Software Safety Requirements

- safety-related software functionalities (e.g. monitoring function)
- safety-related software properties (e.g. independence between monitored element and monitoring element)

implemented by

Software Architectural Design

- static/dynamic design aspects
- requirements allocation
- safety mechanisms, if applicable
- SW partitoning, if applicable

analysed by

improve or verify

refine/ improve

Safety-oriented Software Analysis

refine/ improve

consider

Safety Plan

- safety-related development process
- safety measures

additional development-time safety measures

See: ISO 26262 – Part 6

What is the focus of Safety Analysis?

Functional Safety

Systematic failures (internal & external causes, mitigation)

- Failure related in a deterministic way to a certain cause, e.g., by wrong design or implementation bugs.
- For example: Failures in requirements specification or architectural decision.
- **➔ All SW failures are systematic!**

Random failures (detection, control and mitigation)

- Failure that can occur unpredictably during the lifetime of a hardware element by aging and that follows a probability distribution.
- For example: Failure of a hardware component after a corresponding period of operation.
- **➔ Only in HW and ME!**

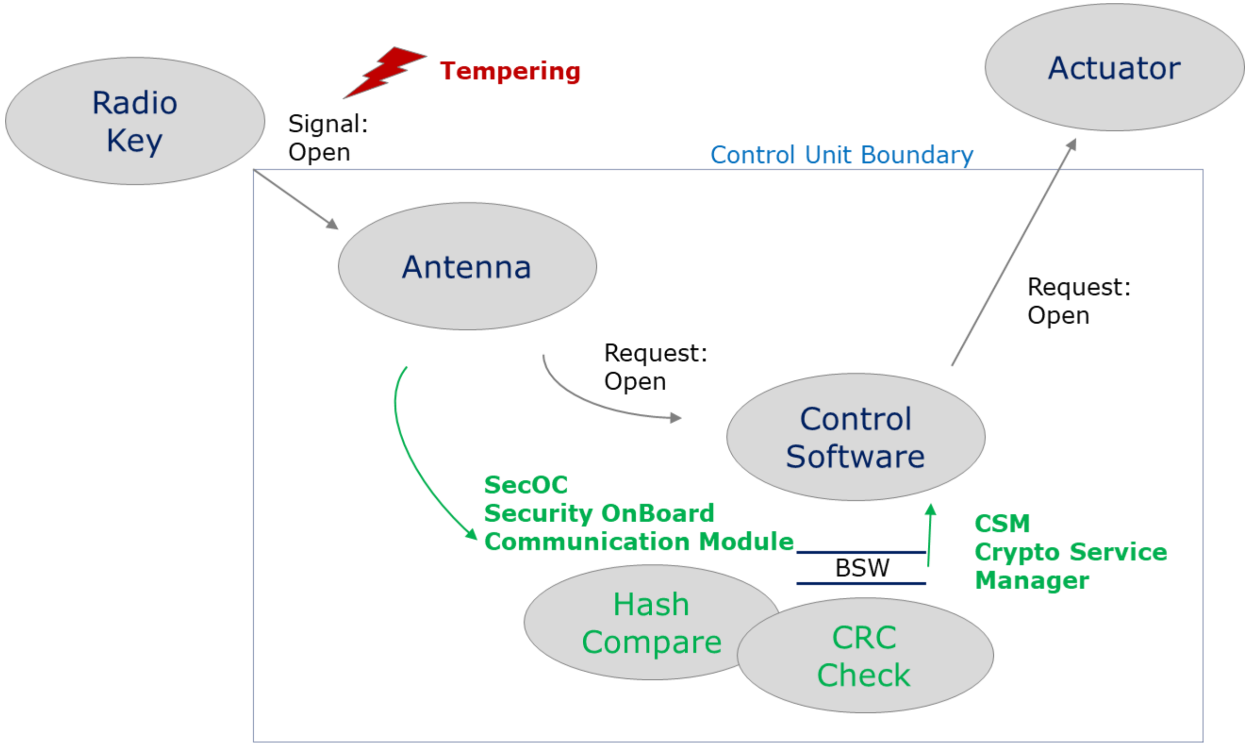
Weakness or Vulnerability analysis

Cyber Security

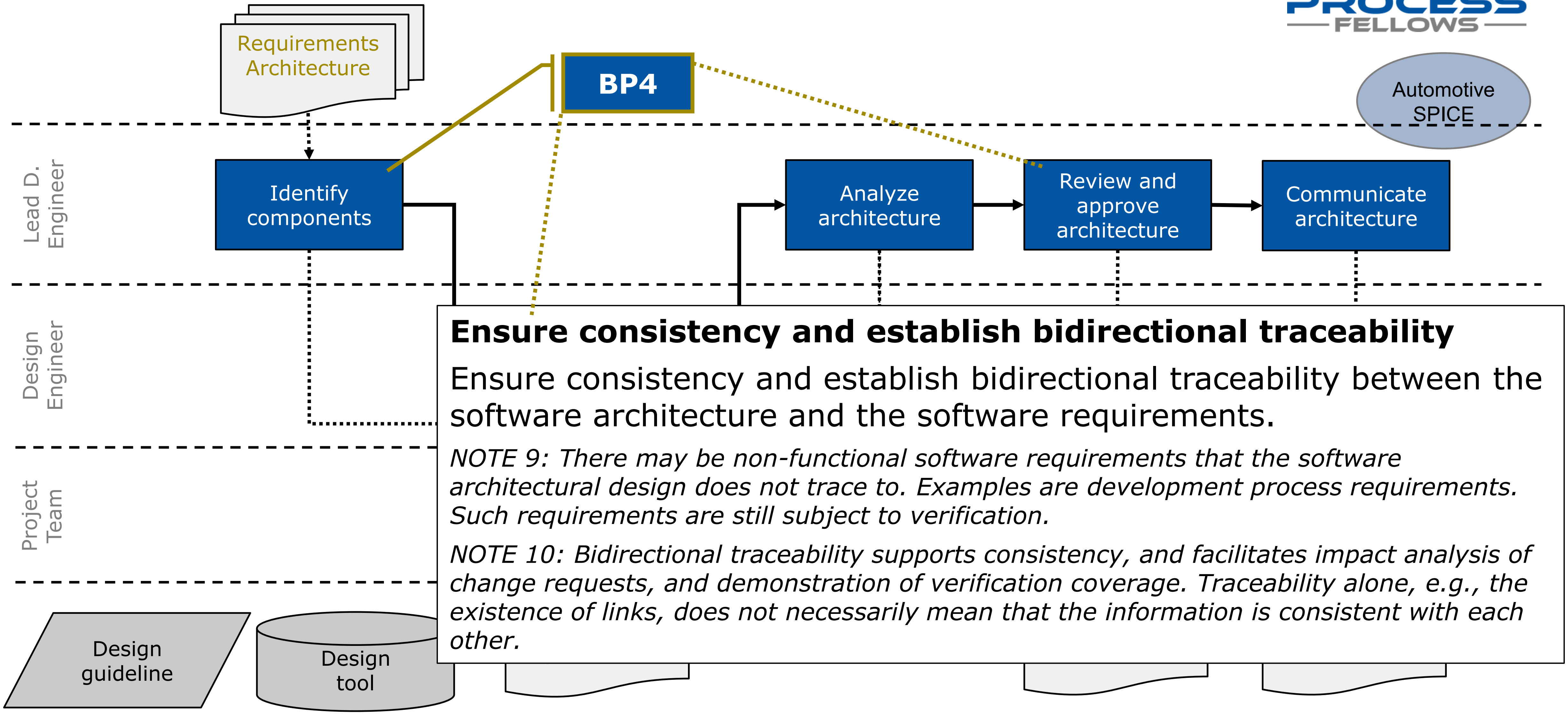
- Cybersecurity can provide additional cases of architectural alternatives (e.g., choosing CS libraries).
- A weakness analysis of the architecture shall be performed.

Important aspects:

- Security controls
- Allocation to requirements
- Architecture and design guidelines
- Review



Software Architectural Design (SWE.2)



Ensure consistency and establish bidirectional traceability

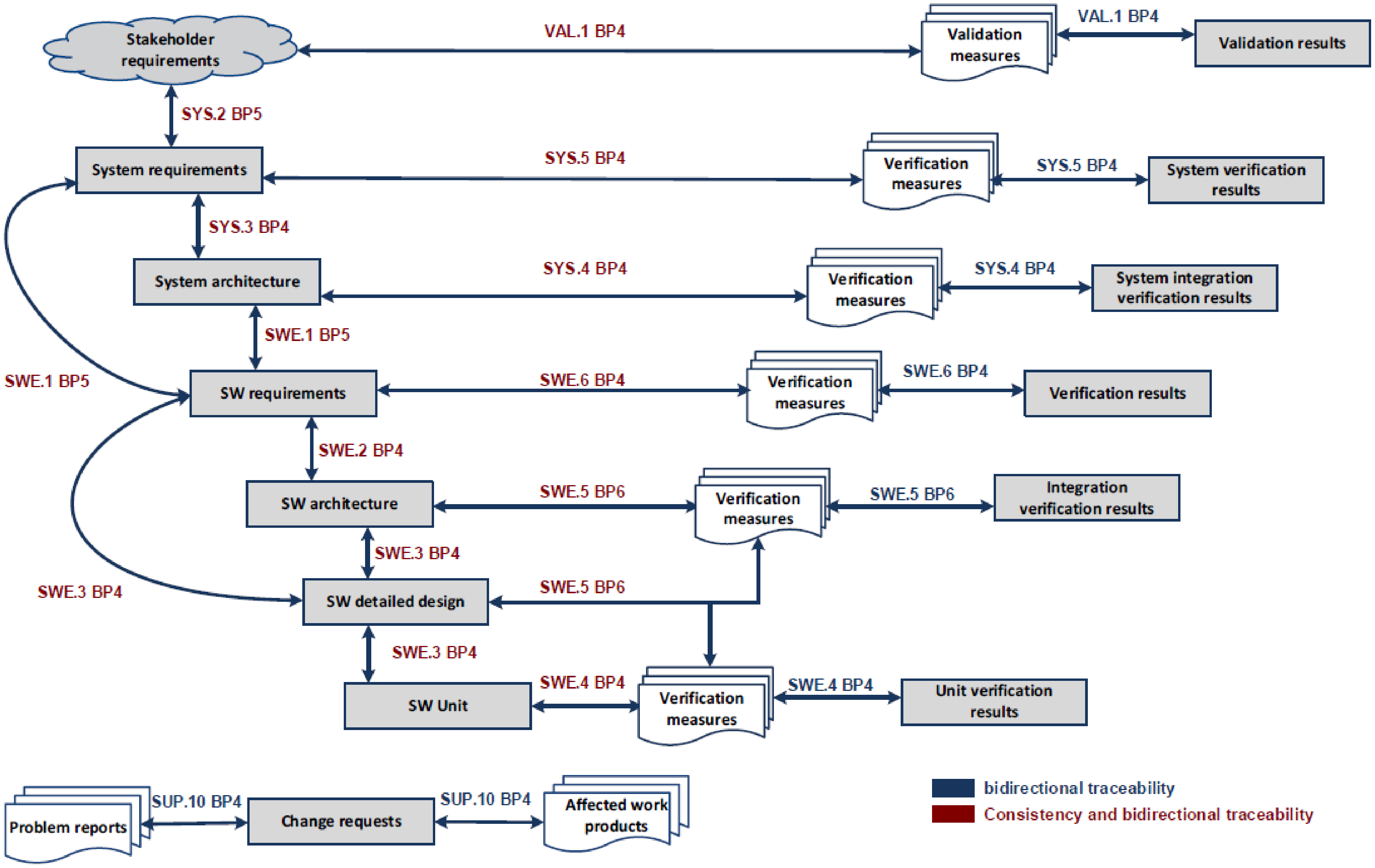
Ensure consistency and establish bidirectional traceability between the software architecture and the software requirements.

NOTE 9: There may be non-functional software requirements that the software architectural design does not trace to. Examples are development process requirements. Such requirements are still subject to verification.

NOTE 10: Bidirectional traceability supports consistency, and facilitates impact analysis of change requests, and demonstration of verification coverage. Traceability alone, e.g., the existence of links, does not necessarily mean that the information is consistent with each other.

Consistency and bidirectional Traceability

Automotive SPICE



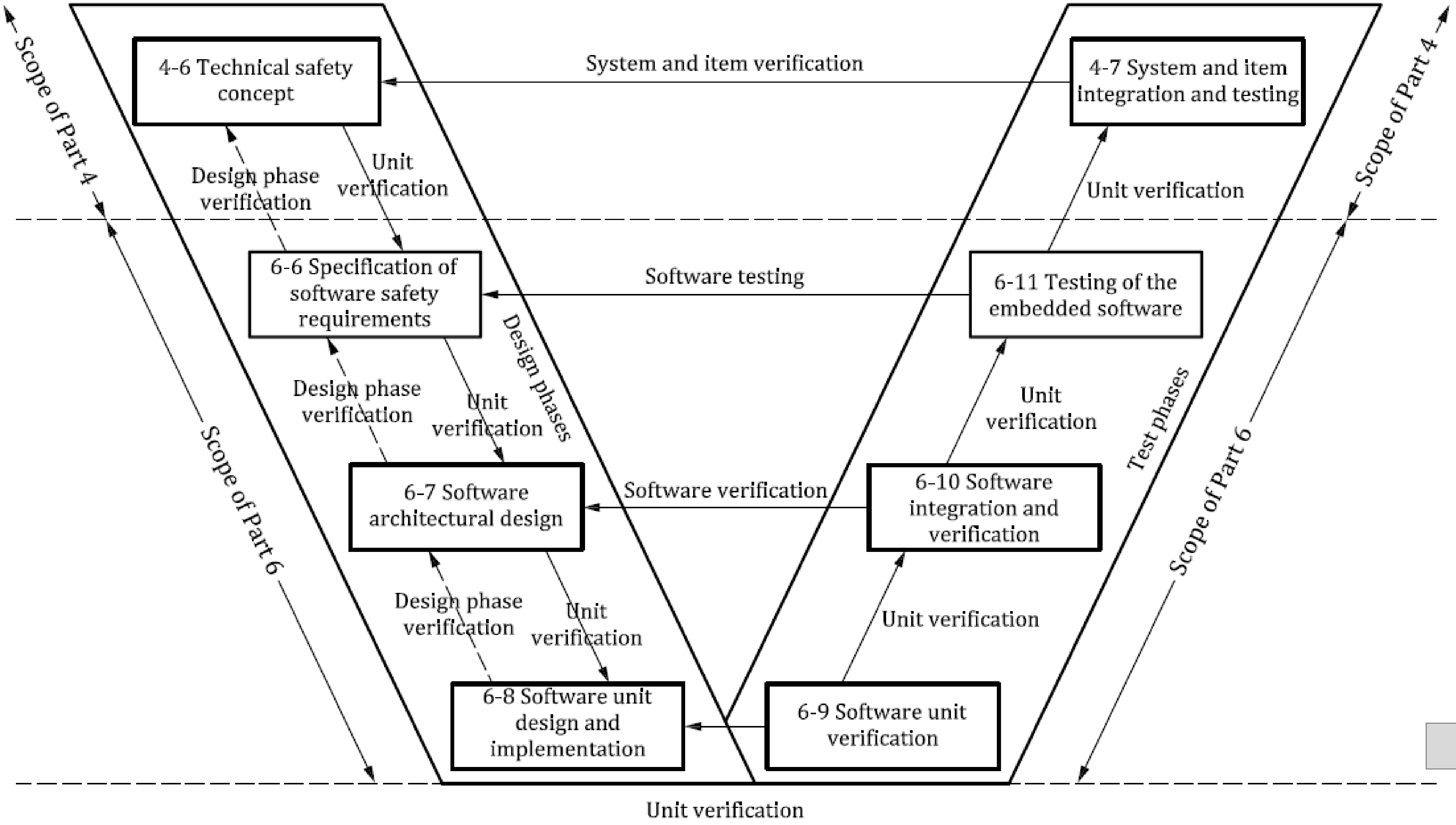
🟦 bidirectional traceability

🟥 Consistency and bidirectional traceability

See: Automotive SPICE 4.0

Consistency and bidirectional Traceability

Functional Safety

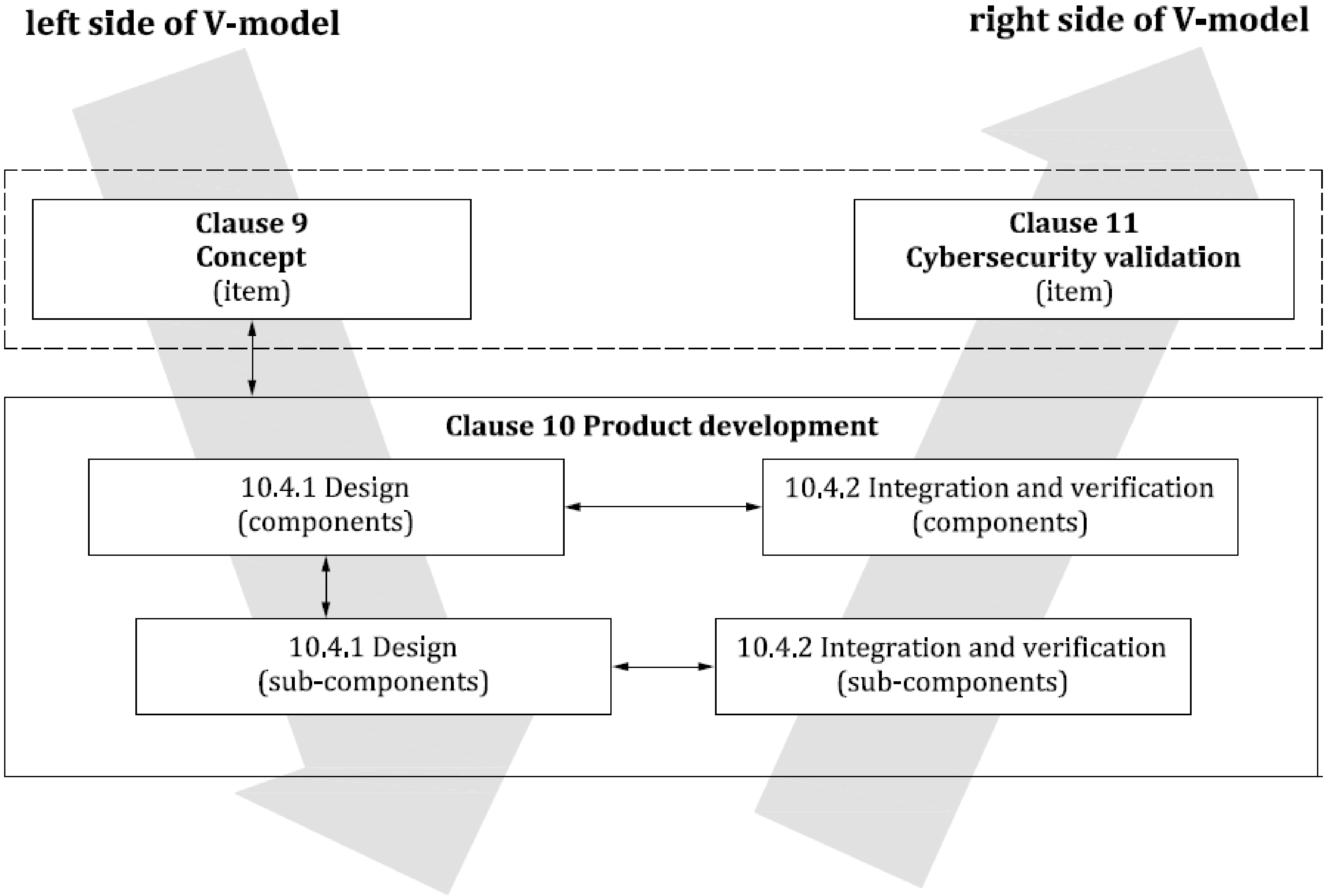


4-6 Technical safety concept
System and item verification
4-7 System and item integration and testing
Design phase verification
Unit verification
Unit verification
Scope of Part 4
Scope of Part 4
6-6 Specification of software safety requirements
Software testing
6-11 Testing of the embedded software
Design phase verification
Unit verification
Design phases
Unit verification
Test phases
Scope of Part 6
Scope of Part 6
6-7 Software architectural design
Software verification
6-10 Software integration and verification
Design phase verification
Unit verification
Unit verification
6-8 Software unit design and implementation
6-9 Software unit verification
Unit verification

See: ISO 26262 – Part 6

Consistency and bidirectional Traceability

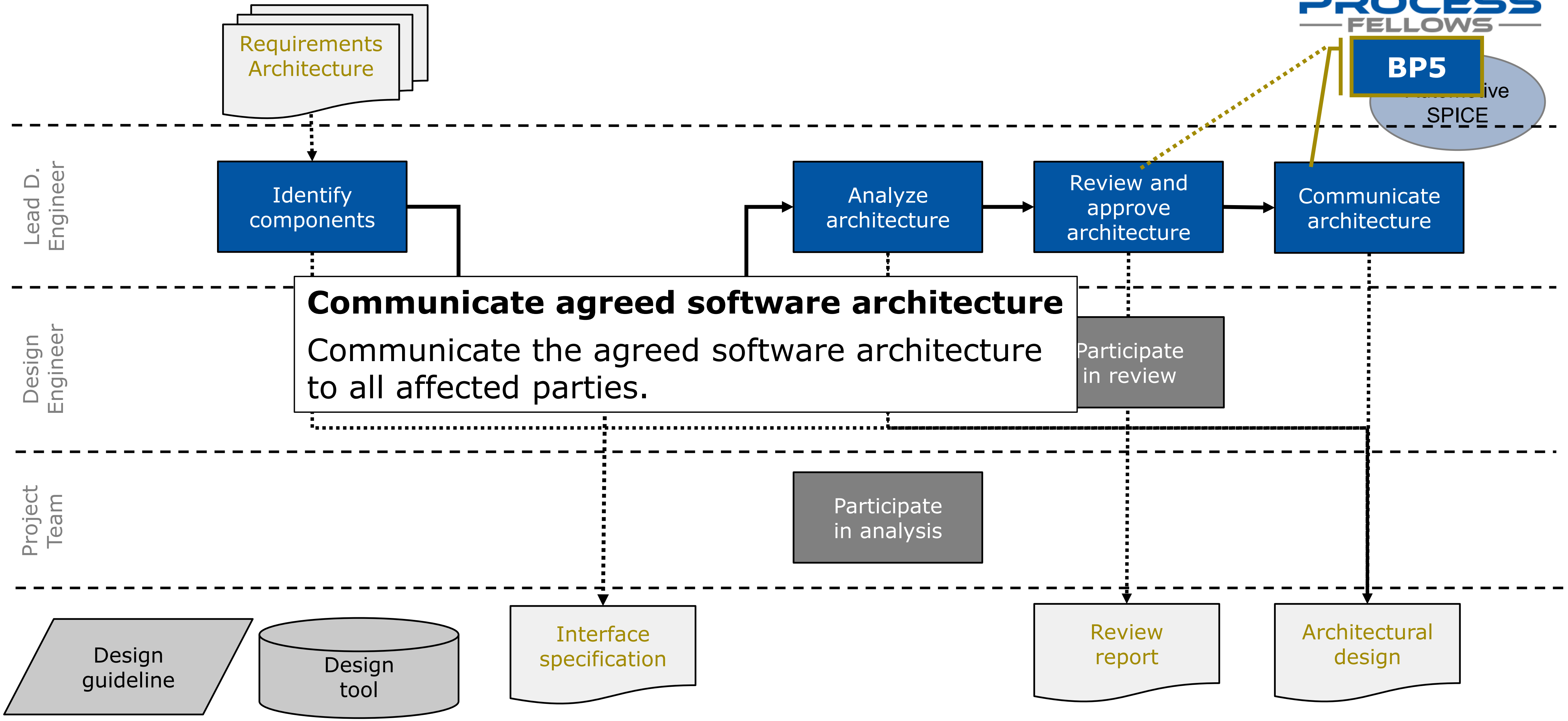
Cyber Security



- Cybersecurity controls need to be allocated to the architecture as well.

See: ISO/SAE 21434

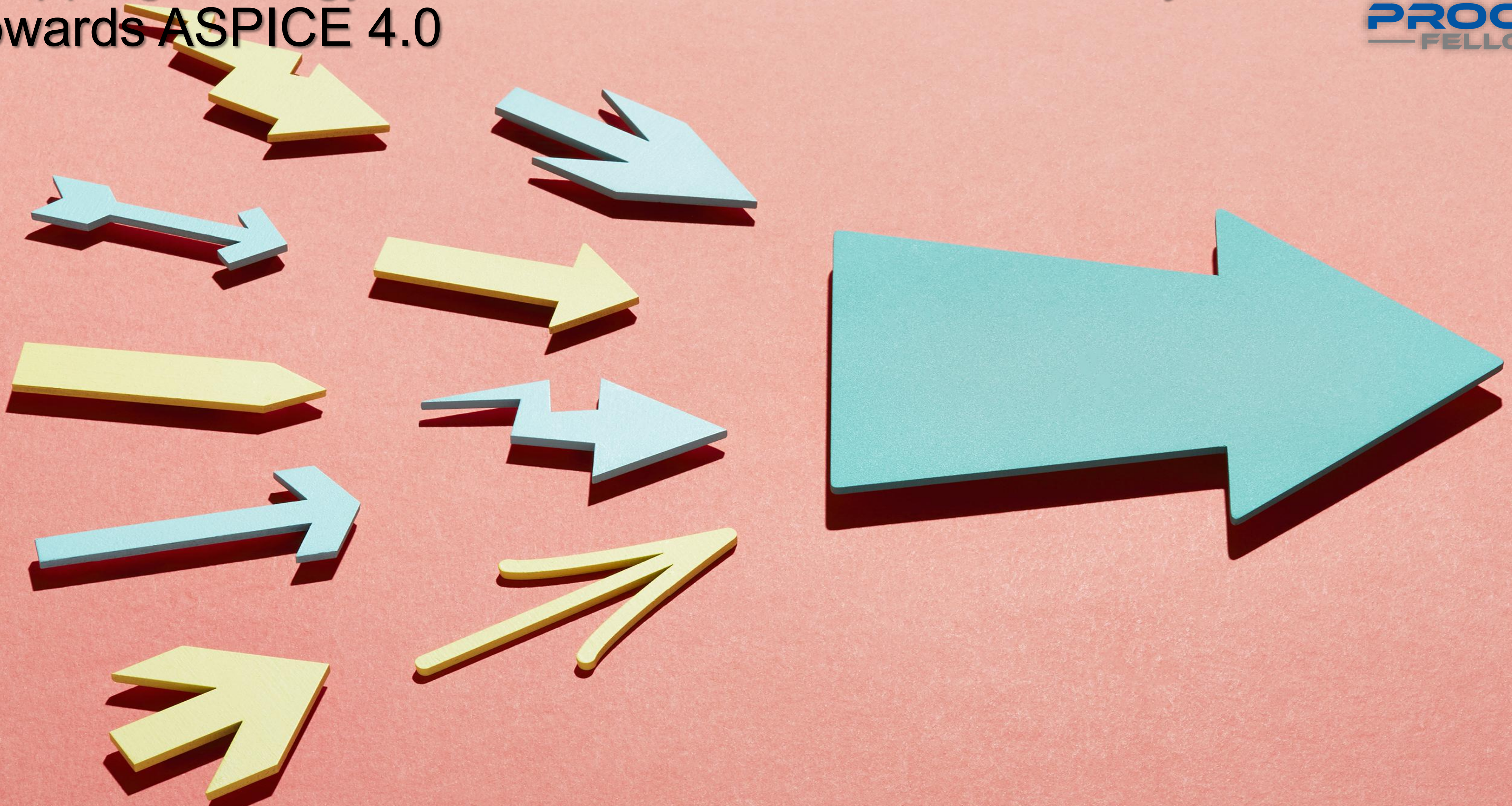
Software Architectural Design (SWE.2)



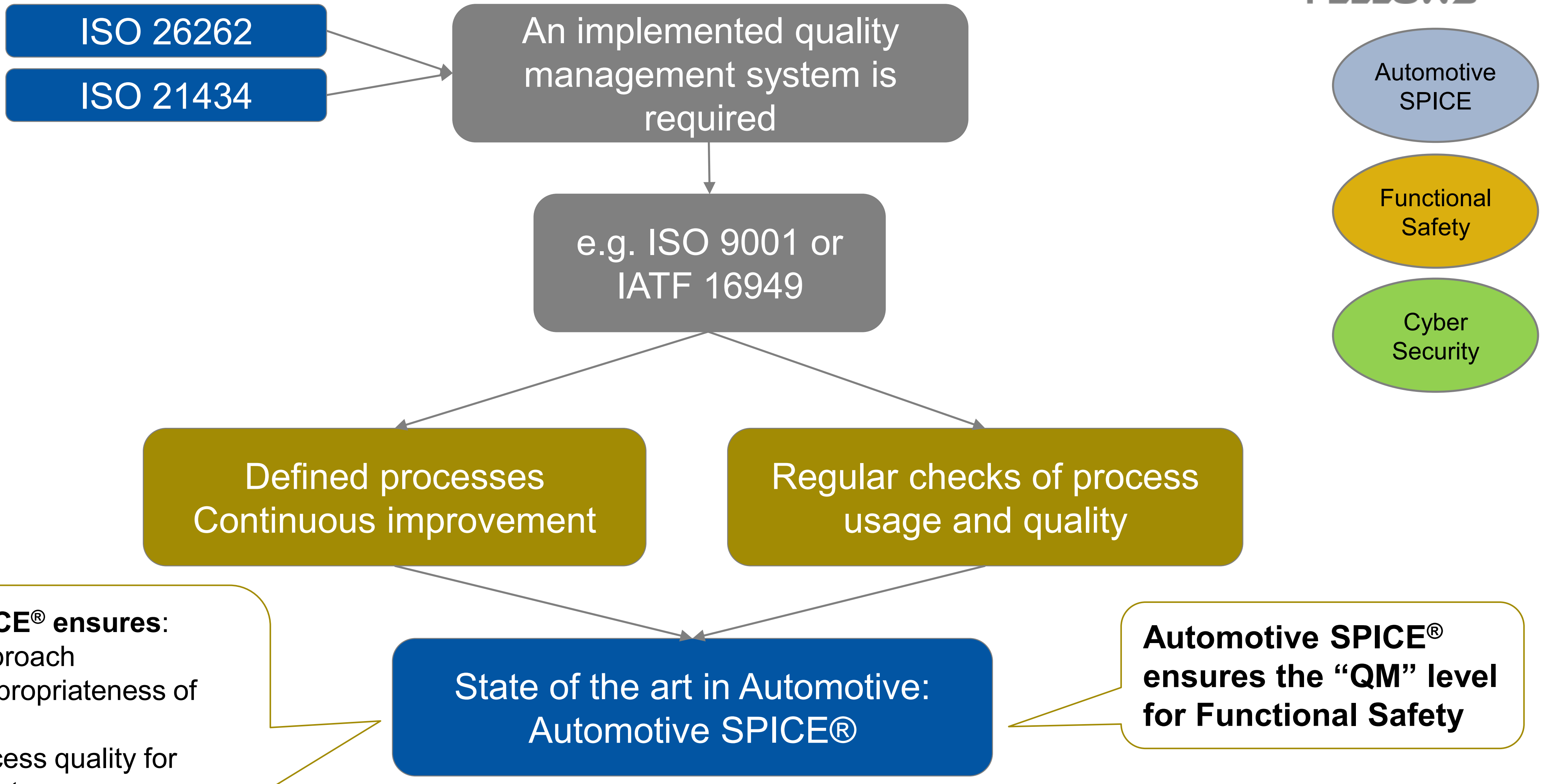
Communicate agreed software architecture

Communicate the agreed software architecture to all affected parties.

Mapping strategy for ISO 26262 and ISO 21434 objectives towards ASPICE 4.0



How to ensure sufficient Process Quality for Safety & Security?


ISO 26262
ISO 21434
An implemented quality management system is required
e.g. ISO 9001 or IATF 16949
Defined processes Continuous improvement
Regular checks of process usage and quality
State of the art in Automotive: Automotive SPICE®
Automotive SPICE
Functional Safety
Cyber Security

Automotive SPICE® ensures:
- Structured approach
- Usage and appropriateness of processes
- Adequate process quality for Functional safety

Automotive SPICE® ensures the "QM" level for Functional Safety

How to combine SPICE Assessment and Safety Audits?

Automotive SPICE

Scope:
- Check for fulfillment of process purpose
- Check for implementation of base practices
- Ensure adequate evidences for performance (work products)

Performance:
- Questions based on base practices

Result:
- Process capability for each process of the project

Consequence:
- Process improvements based on findings

Functional Safety

Scope:
- Check for completeness of safety case (existence of necessary work products)
- Check (spot-check) required content of work products
- Ensure performance of safety activities

Performance:
- Questions based on work products

Result:
- Evidence of completeness of safety activities and safety case

Consequence :
- Recommendation for release

Automotive SPICE

Functional Safety

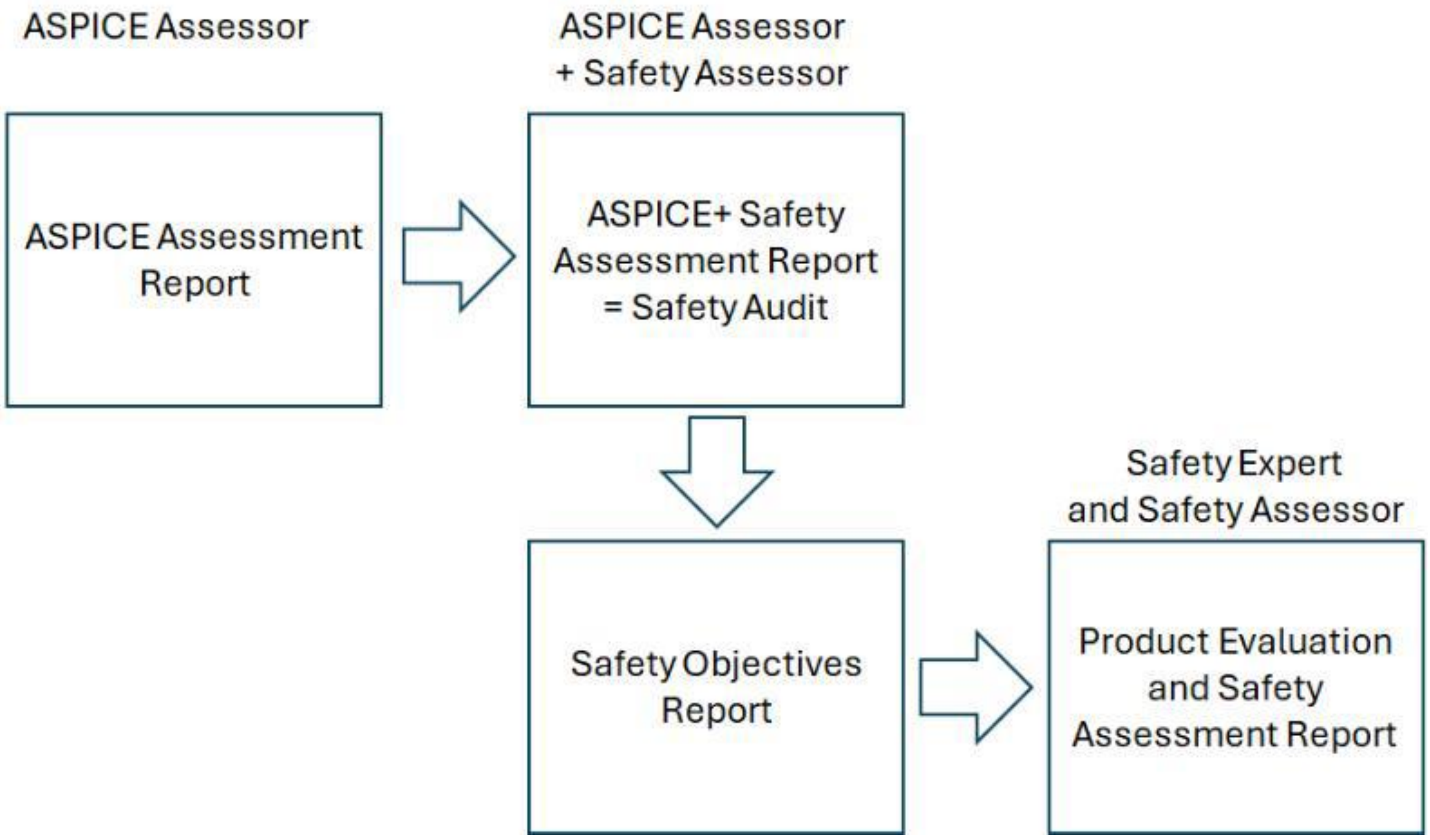
Cyber Security

Check for completeness of security case is recommended as well

Reporting for ASPICE Assessment, Safety Audit and Cybersecurity Audit – plus interface for Safety/Security Assessment

PROCESS FELLOWS

Reports and interfaces between ASPICE and ISO 26262 assessors


ASPICE Assessor
ASPICE Assessment Report
ASPICE Assessor + Safety Assessor
ASPICE+ Safety Assessment Report = Safety Audit
Safety Objectives Report
Safety Expert and Safety Assessor
Product Evaluation and Safety Assessment Report

Automotive SPICE

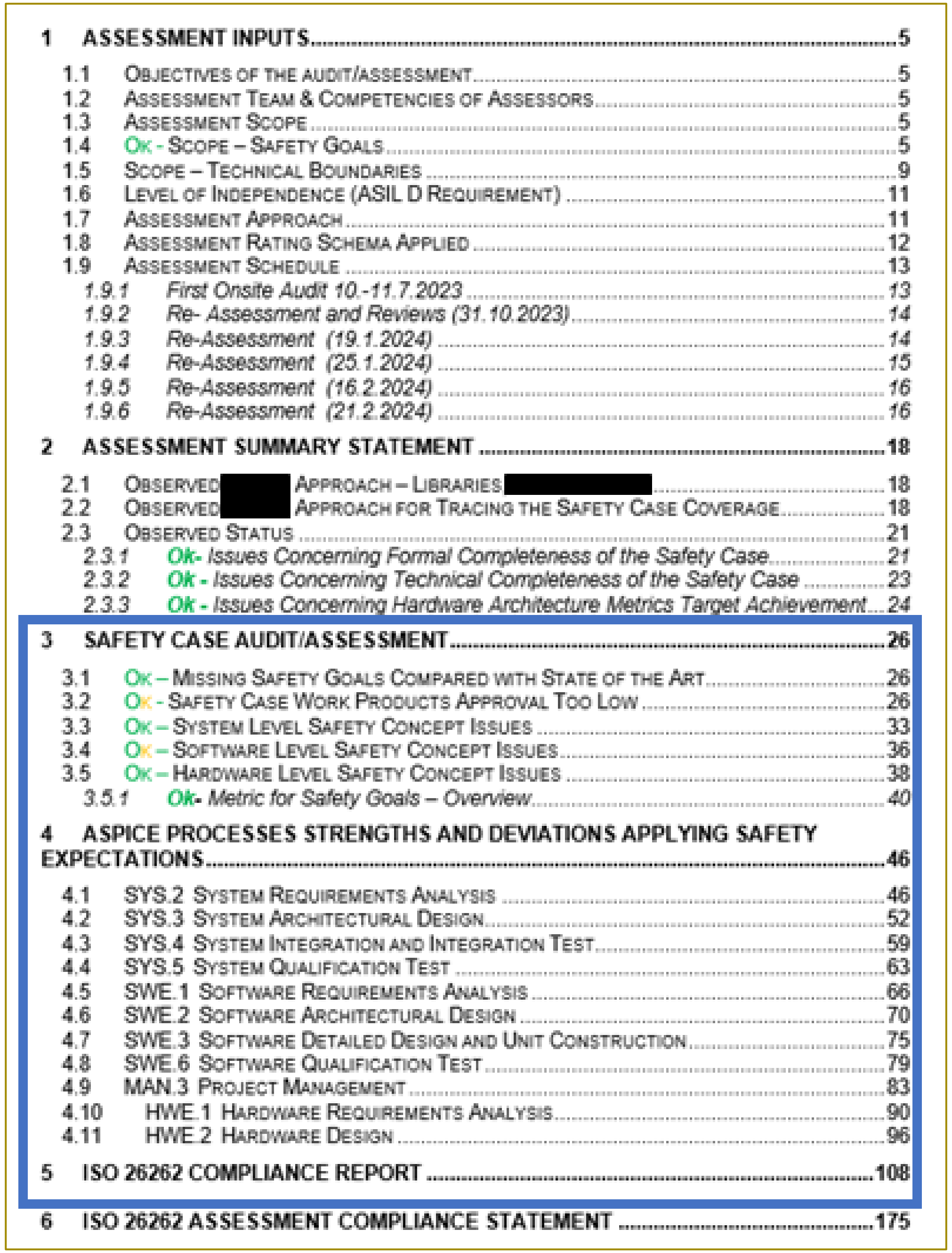
Functional Safety

Cyber Security

Same for security assessment

See paper:
Updated Experiences with Using ASPICE 4.0 for Safety Audits and Interfacing Safety Assessments
Richard Messnarz, Damjan Ekert, Andreas Riel, Georg Macher, Tobias Danmayr, Laura Aschbacher

Example of a fully integrated report



Automotive SPICE

Functional Safety

See paper:
Updated Experiences with Using ASPICE 4.0 for Safety Audits and Interfacing Safety Assessments
Richard Messnarz, Damjan Ekert, Andreas Riel, Georg Macher, Tobias Danmayr, Laura Aschbacher

Thank you for listening !
Any questions ?

PROCESS FELLOWS

Process Fellows GmbH | Schlegelleithe 8 | 91320 Ebermannstadt | GERMANY
Phone: +49 9194 3719 957 | Fax: +49 9194 3719 579
Website: www.processfellows.de | E-Mail: info@processfellows.de