

# ISO 26262 and AUTOSAR. Requirements and Solutions for Safety Related Software.



Simon Fürst.  
5th Vector Congress.  
1<sup>st</sup> and 2<sup>nd</sup> December 2010, Stuttgart.

**BMW Group**



# ISO 26262 and AUTOSAR.

## Preamble.

- ISO 26262 address the development of safety-related systems
  - ⇒ AUTOSAR is “only” the infrastructure part of the software of a system
  
- ISO 26262 contains two different kind of requirements
  - Process related requirements: “how to develop the system”
  - Technical requirements: “what system to develop”
  - ⇒ The AUTOSAR development partnership as well as the implementers of AUTOSAR have to respect both
  
- AUTOSAR only provides specifications
  - ⇒ Only a subset of the requirements of ISO 26262 are applicable
  - ⇒ For the implementers of AUTOSAR an overlapping subset of the requirements of ISO 26262 are applicable

# **ISO 26262 and AUTOSAR.**

## **AUTOSAR.**

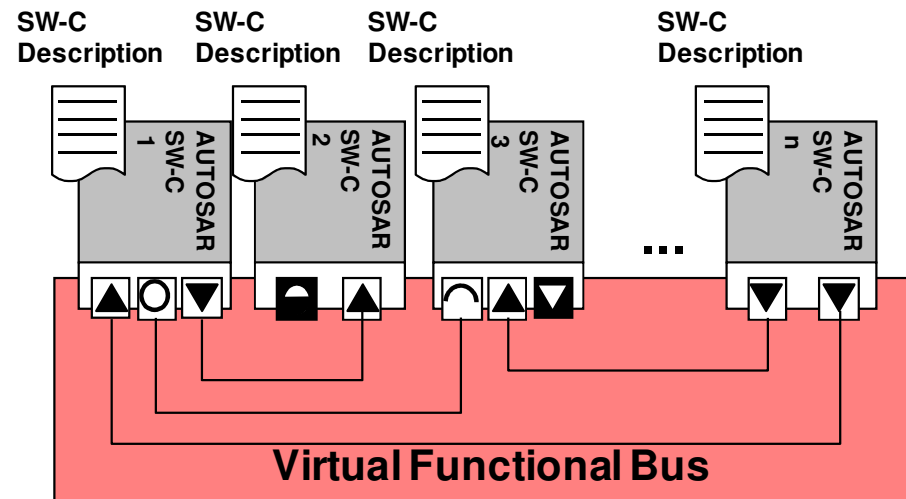
**What has AUTOSAR done so far?**

# ISO 26262 and AUTOSAR.

## AUTOSAR's basic approach.

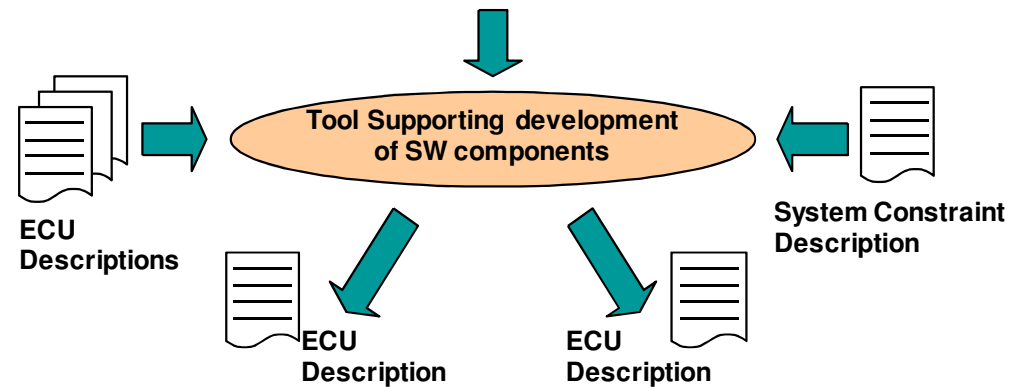
### Virtual Integration

Independent of hardware  
 Virtual Functional Bus



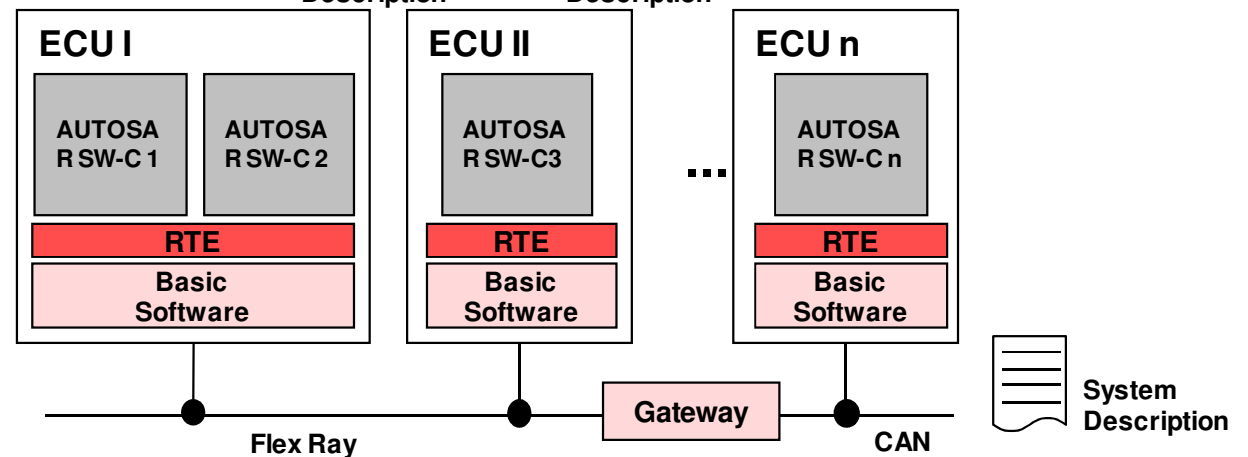
### Introduction of HW Attributes

Holistic view of the entire system,  
 both software and hardware



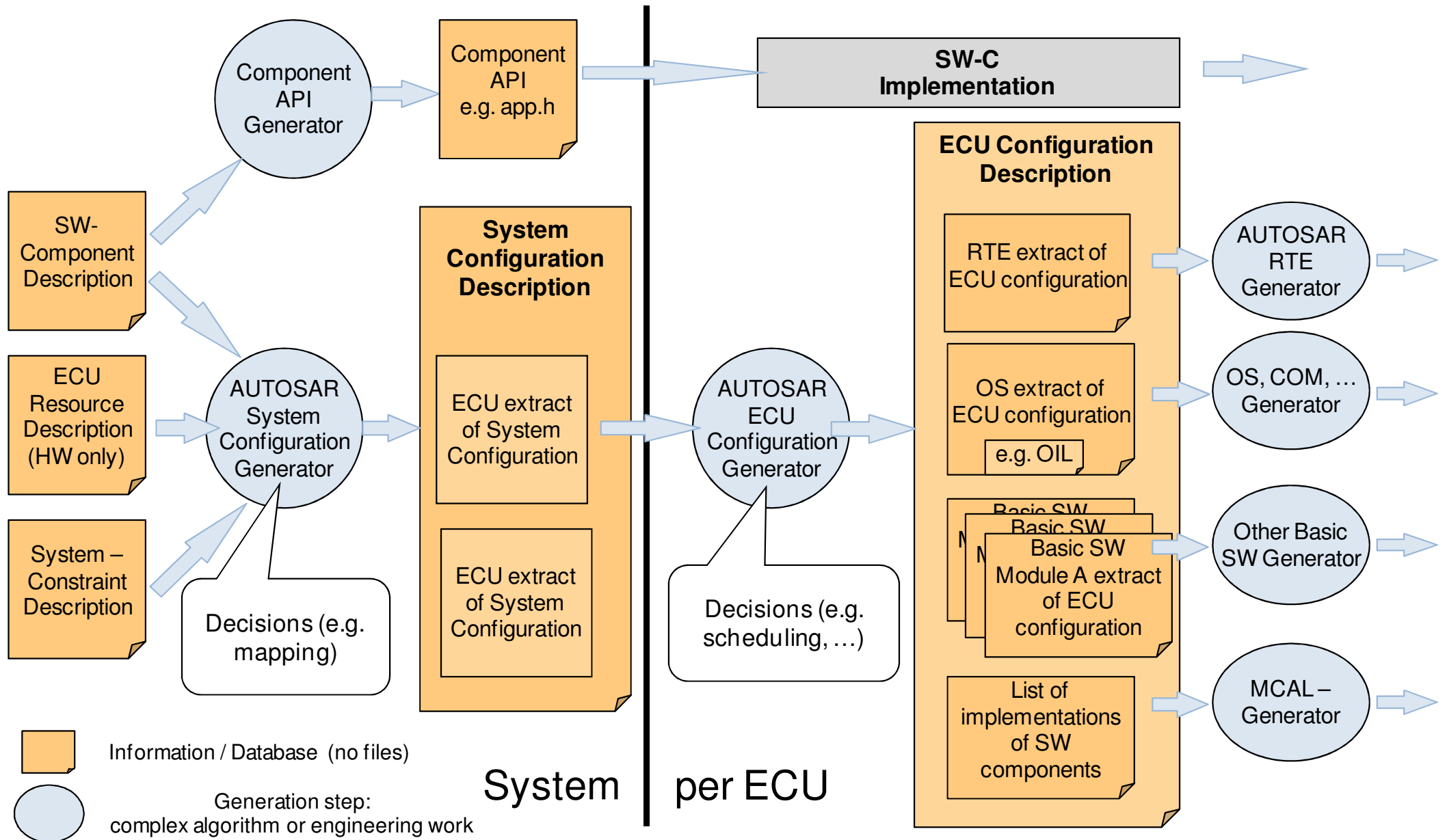
### ECU Configuration

Run-Time Environment  
 Separation of system into its ECU  
 (plus common infrastructure)



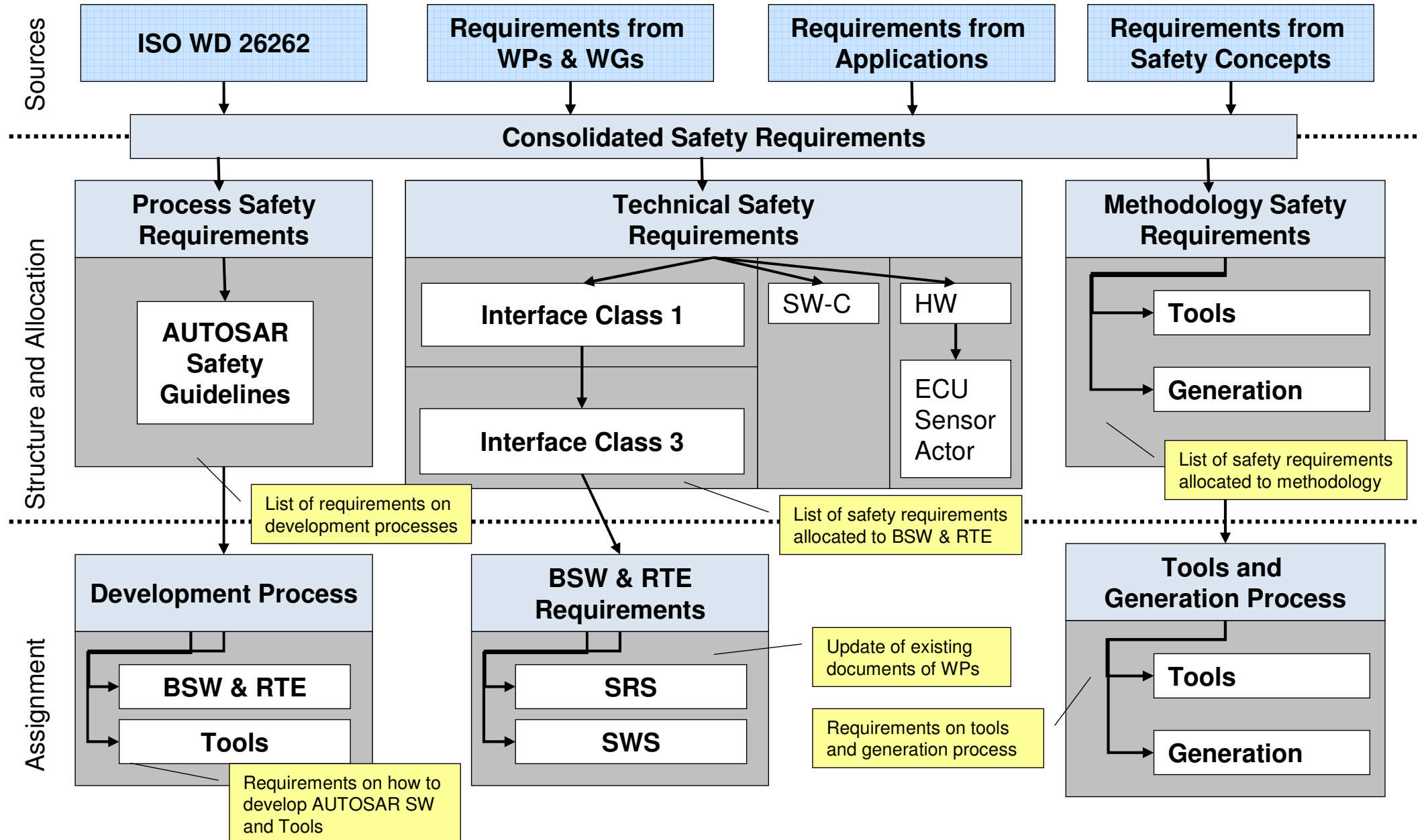
# ISO 26262 and AUTOSAR.

## AUTOSAR Methodology.



# ISO 26262 und AUTOSAR.

## Approach of AUTOSAR w.r.t. Functional Safety.



# **ISO 26262 and AUTOSAR.**

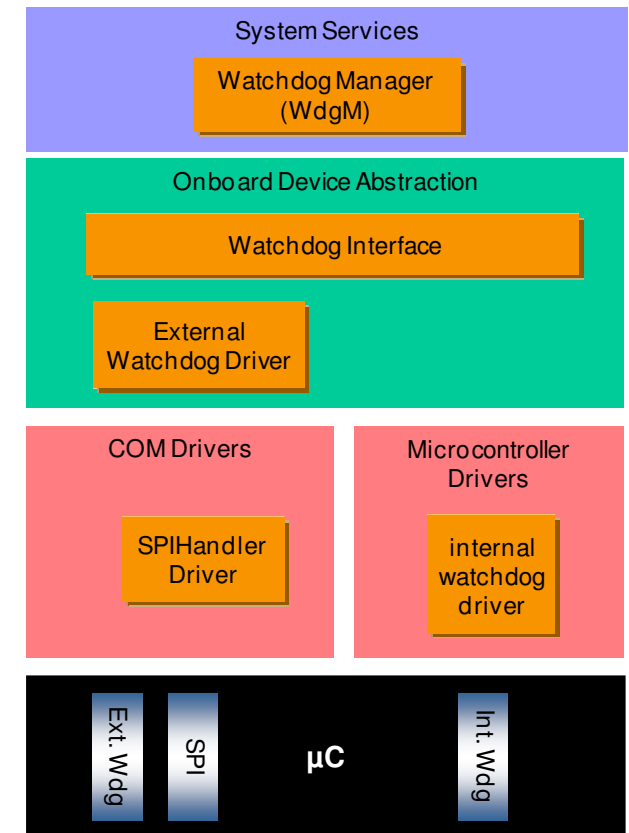
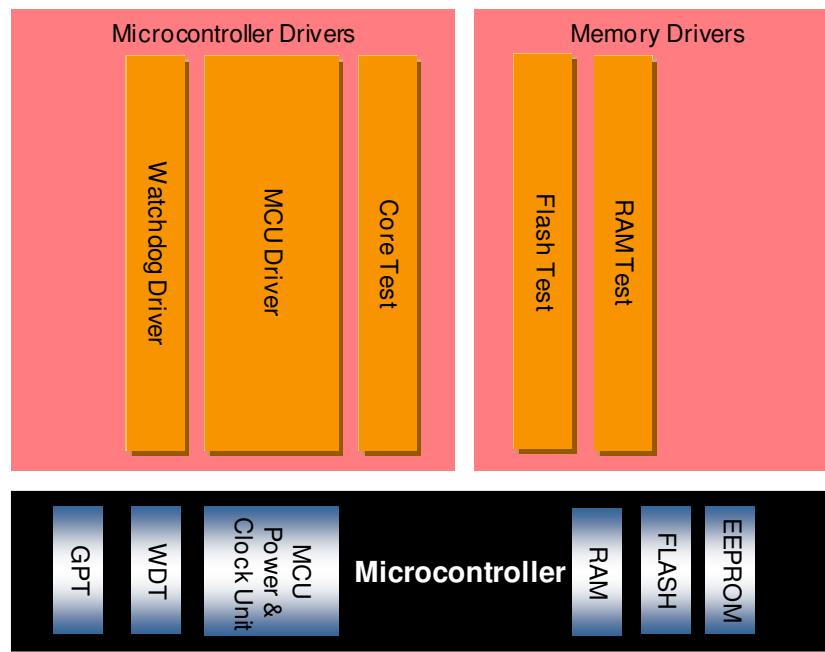
## **Overview of available, built-in AUTOSAR safety mechanisms.**

- Built-in self test mechanisms for detecting hardware faults (testing and monitoring)
- Run-time mechanisms for detecting software faults during the execution of software
  - Program flow monitoring
- Run-time mechanisms for preventing fault interference
  - Memory partitioning for SW-Cs
  - Time partitioning for applications
- Run-time mechanisms for protecting the communication
  - End-to-end (E2E) communication protection for SW-Cs
- Run-time mechanisms for error handling

# ISO 26262 and AUTOSAR.

## Built in safety mechanisms for detecting errors.

- Memory:
  - RAM Test
  - Flash Test
  - Support for ECC memory
- Core:
  - Core Test
- Logical and temporal program flow monitoring



# ISO 26262 and AUTOSAR.

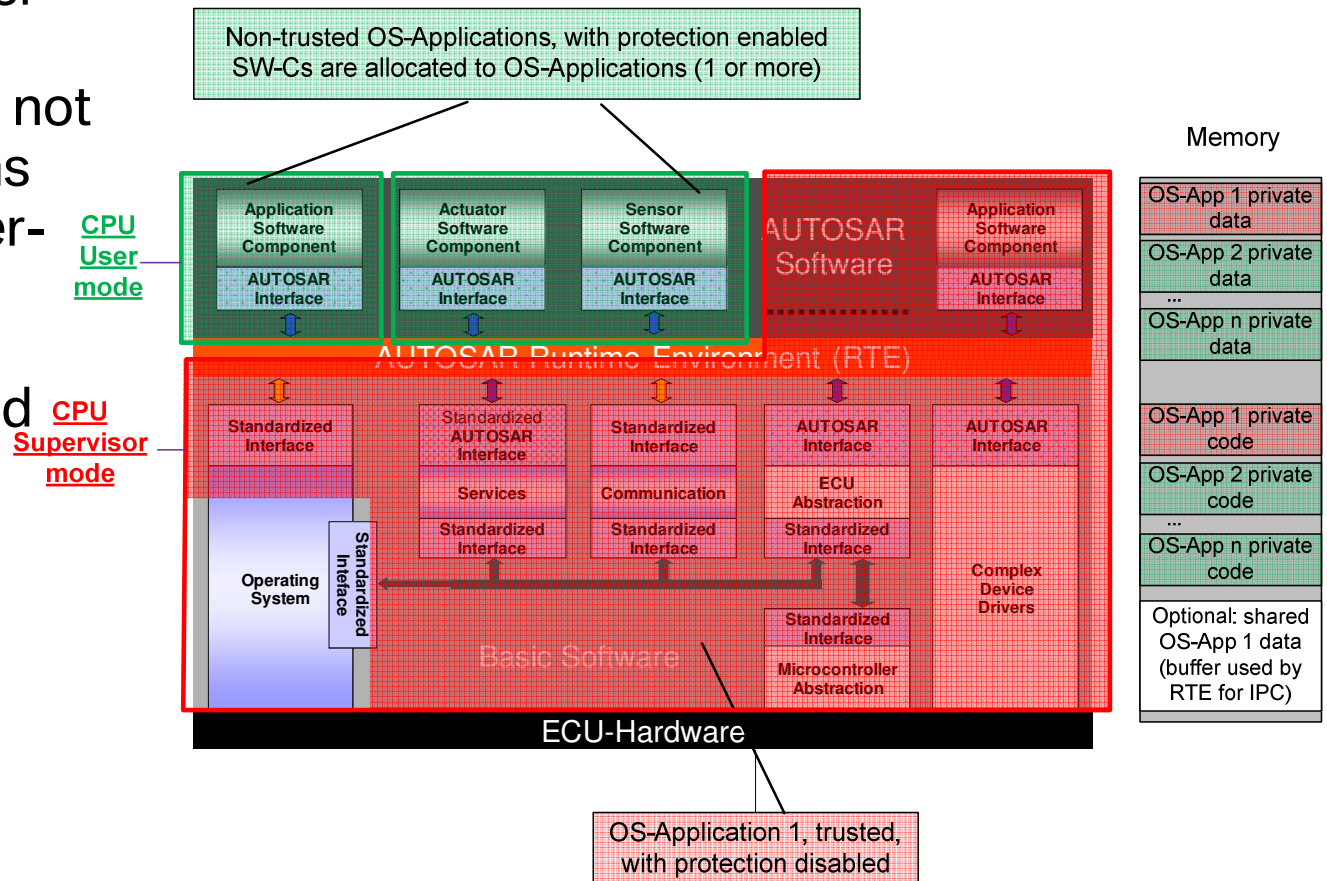
## Run-time mechanisms for error handling.

- Detected errors in the basic software:
  - Are reported through DEM to SW-Cs. SW-Cs then executes application-specific actions
  - Are reported to FIM, which permits to disable some functions of SW-C
- Detected hardware errors:
  - Arithmetic exceptions (e.g. division by 0): handled by OS callouts (small error handling routines in the context of basic software). Typical reaction – ECU reset
  - HW errors detected by HW testing: handled by callouts. Typical reaction – ECU reset
  - Errors detected by MMU/MPU (memory and time partitioning). It will shut down or restart the faulty SW-C partition

# ISO 26262 and AUTOSAR.

## Memory partitioning for SW-Cs.

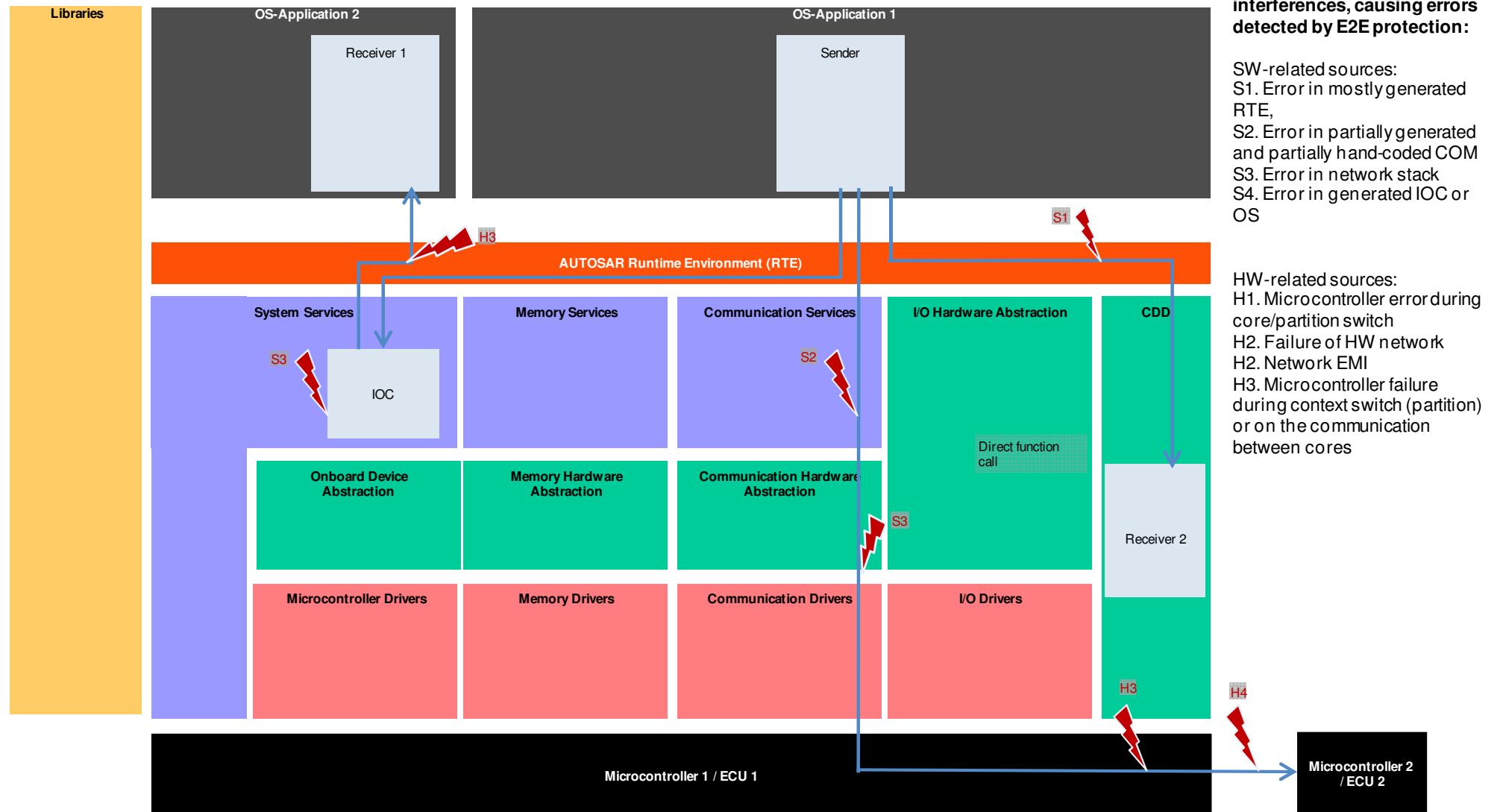
- Enables create protection boundaries around groups of SW-Cs
- This is realized by user-mode/non-trusted memory partitions (for groups of SW-Cs)
- This protects from interference:
  - (1) basic software and
  - (2) SW-Cs in other partitions
- Basic software is not partitioned. It runs with in CPU supervisor mode with full access to memory, CPU and all other hardware resources



# ISO 26262 and AUTOSAR.

## End-to-End communication protection (1/2).

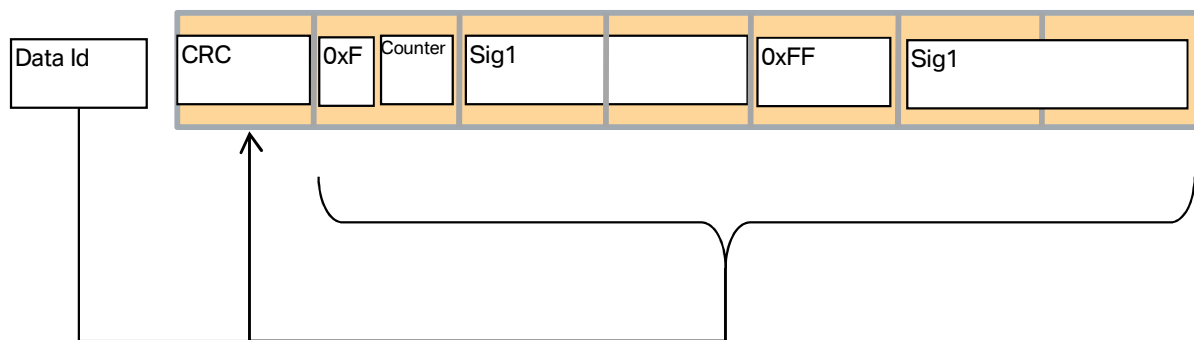
- E2E protection detects faults in data caused by both hardware and in software



# ISO 26262 and AUTOSAR.

## End-to-End communication protection (2/2).

- Protection of data exchanged over communication channels like FlexRay and CAN
- Failure modes addressed as defined by ISO DIS 26262 for communication (repetition, deletion, insertion, incorrect sequence, corruption, timing faults, addressing faults, inconsistency, masquerading)
- Three different protection mechanisms for data are used
  - CRC, counter, Data ID, timeout detection
  - Data ID included in to calculated CRC, but not sent



CRC := CRC8 over (1) Data Id, (2) all serialized signal (including empty areas, excluding CRC byte itself)

# ISO 26262 and AUTOSAR.

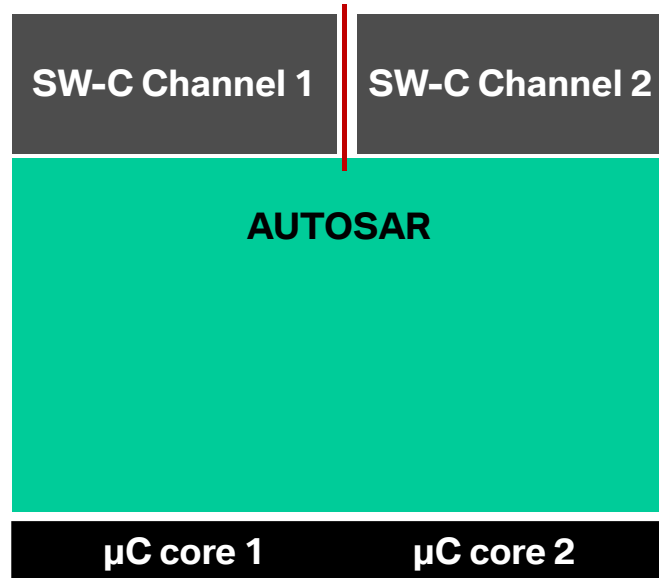
## Safety mechanisms supported by AUTOSAR.

- Implementation of typical safety concepts in the automotive domain
  - Intelligent HW watchdog (ASIC) / 3-level safety concept
  - Monitored channel (2  $\mu$ Cs, the second is a simple  $\mu$ C monitoring the first  $\mu$ C)
  - Dual channel (2 AUTOSAR  $\mu$ Cs)
- Application redundancy (on the same or different  $\mu$ Cs)
- Basic Software redundancy inside one ECU

# ISO 26262 and AUTOSAR.

## Application redundancy.

- Assuming integrity of HW/ECU and AUTOSAR Basic Software implementation, SW redundancy with ASIL decomposition can be used within the same ECU.
- Distribution of SW channels across ECUs is also possible.

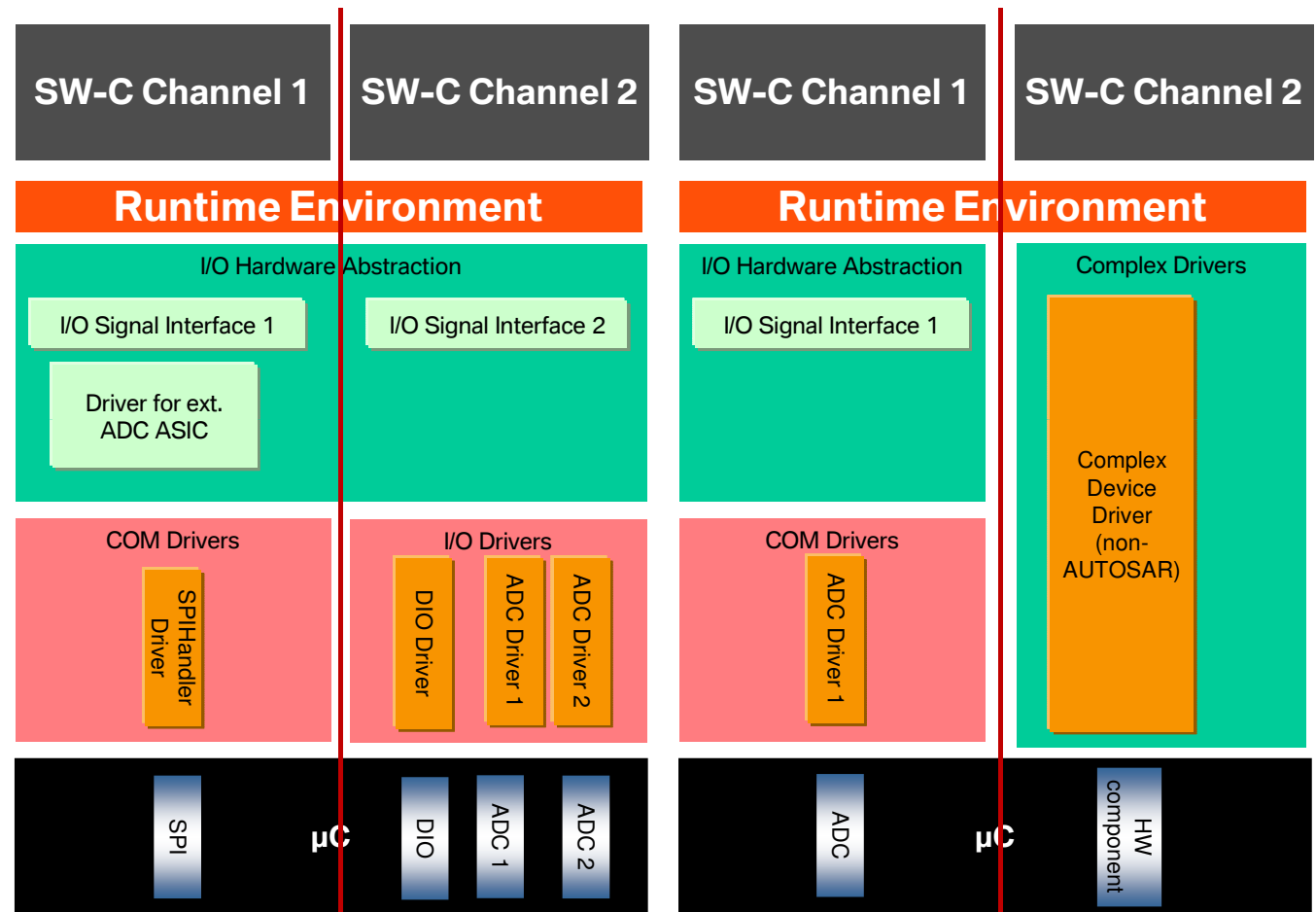


# ISO 26262 and AUTOSAR.

## Basic Software redundancy inside one ECU.

- Redundancy inside AUTOSAR e.g. double input/output data paths through
  - Redundant IO hardware abstraction and IO drivers
  - Redundant and diverse (e.g. ADC + DIO, internal ADC + external ADC)

- Redundancy through integration of complex drivers running on the same  $\mu\text{C}$  offering a redundant data path



# **ISO 26262 and AUTOSAR.**

## **AUTOSAR.**

# **What is ISO 26262 saying?**

# ISO 26262 and AUTOSAR.

## Safety Element out of Context.

### Idea

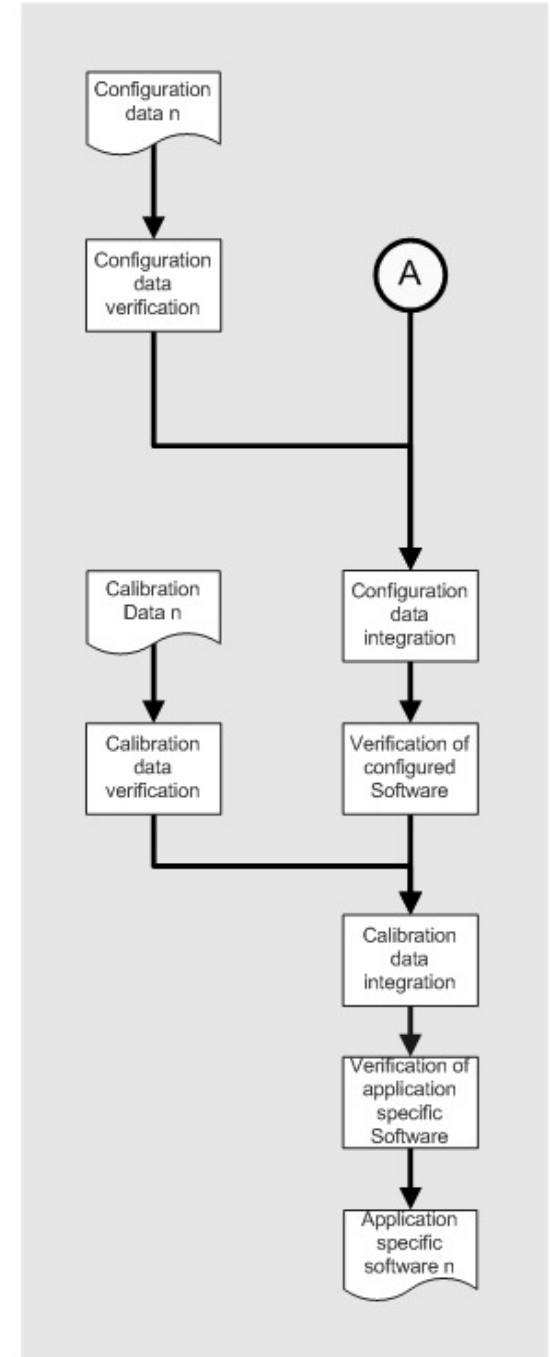
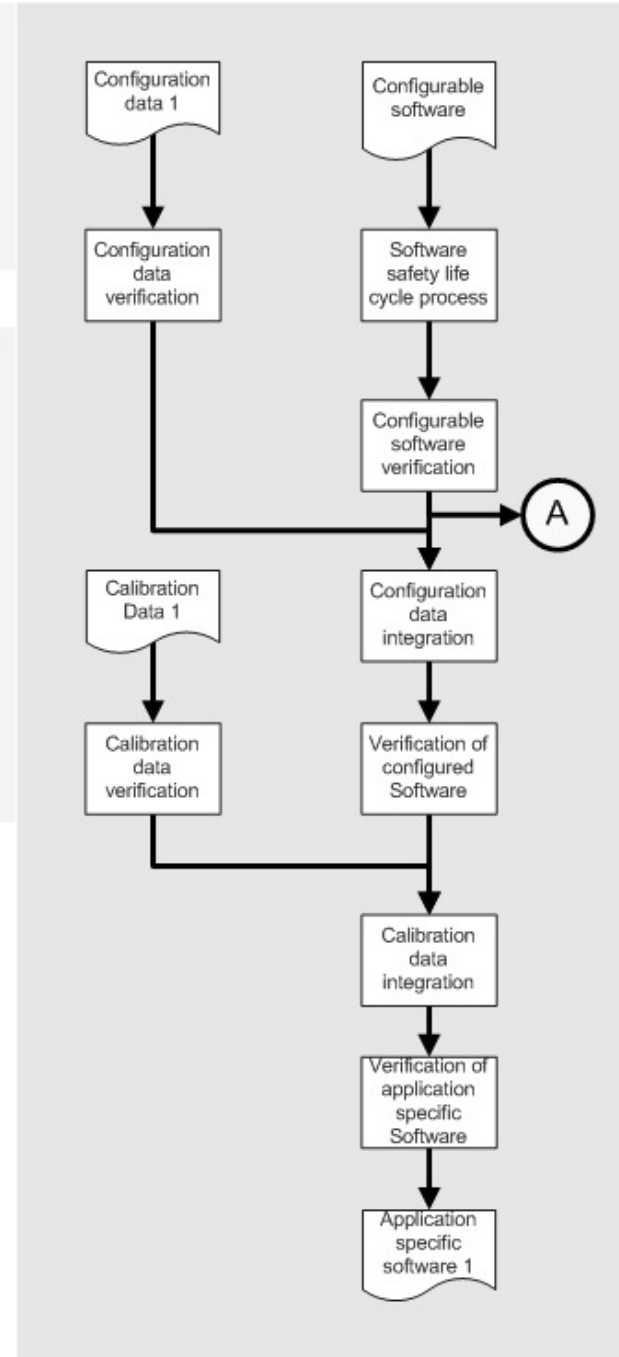
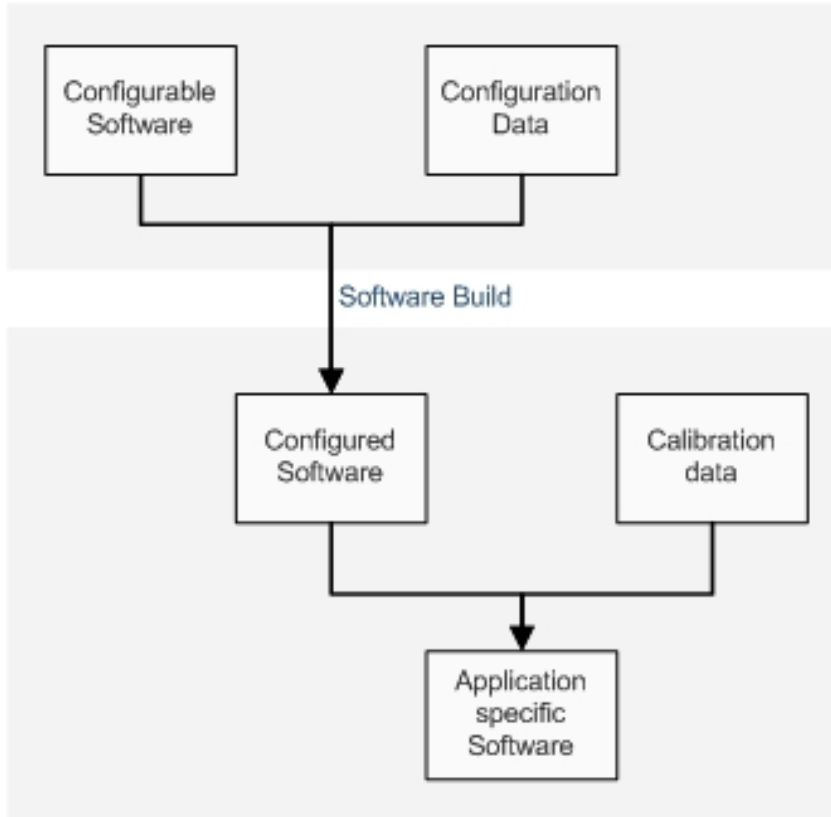
- A SEooC is a pre-qualified safety element that is developed independently from an item development
- A SEooC must be usable in an item development while taking benefit from the work done during the pre-qualification

### Definition

A Safety Element out of Context (SEooC) is a safety element for which an item does not exist at the time of the development. A SEooC can either be a subsystem, a software component, or a hardware component.

- A SEooC is never an item.
- A SEooC can either be a subsystem, a hardware component, or a software component.
- Typically, requirements at higher levels remain in the status "assumed" (see ISO°26262-8, Clause°5) and will be confirmed when the SEooC is used in an item development.
- The correct implementation of the assumed requirements will be verified during the SEooC development, but the validation takes place during the item development. The development of a SEooC starts at a certain level of requirements and design, and all information on requirements or design prerequisites, are pre-determined in the status "assumed".
- Non-functional requirements might be in the status "assumed" at the same level of requirements where functional ones are in the status "accepted".

# ISO 26262 and AUTOSAR. SEooC and configurable software.



Configuration of an AUTOSAR Basic Software Stack is heavily based on

- Configuration data and
- Calibration data

Configuration and calibration data is fully described in standardized XML templates

# ISO 26262 und AUTOSAR.

## Part 6: Software architectural design (1/2).

### Excerpt from chapter 7 “Software architectural design”

**7.4.3** The software architectural design shall exhibit the following properties by use of the principles listed in Table 4:

- a) modularity;
- b) encapsulation and;
- c) minimum complexity.

Table 4 — Principles for software architectural design

Methods		ASIL			
		A	B	C	D
1a	Hierarchical structure of software components	++	++	++	++
1b	Restricted size of software components	++	++	++	++
1c	Restricted size of interfaces	+	+	+	+
1d	High cohesion within each software component	+	++	++	++
1e	Restricted coupling between software components	+	++	++	++
1f	Appropriate scheduling properties	++	++	++	++
1g	Restricted use of interrupts	+	+	+	++

# ISO 26262 und AUTOSAR.

## Part 6: Software architectural design (2/2).

Table 5 — Mechanisms for error detection at software architectural level

Methods		ASIL			
		A	B	C	D
1a	Plausibility check <sup>a</sup>	++	++	++	++
1b	Detection of data errors <sup>b</sup>	+	+	+	+
1c	External monitoring facility	o	+	+	++
1d	Control flow monitoring	o	+	++	++
1e	Diverse software design <sup>c</sup>	o	o	+	++

a Plausibility checks include assertion checks. Complex plausibility checks can be realised by using a reference model of the desired behaviour.  
 b Types of methods that may be used to detect data errors include error detecting codes and multiple data storage.  
 c Diverse software design is not intended to imply n-version programming.

Table 6 — Mechanisms for error handling at software architectural level

Methods		ASIL			
		A	B	C	D
1a	Static recovery mechanism <sup>a</sup>	+	+	+	+
1b	Graceful degradation <sup>b</sup>	+	+	++	++
1c	Independent parallel redundancy <sup>c</sup>	o	o	+	++
1d	Correcting codes for data	+	+	+	+

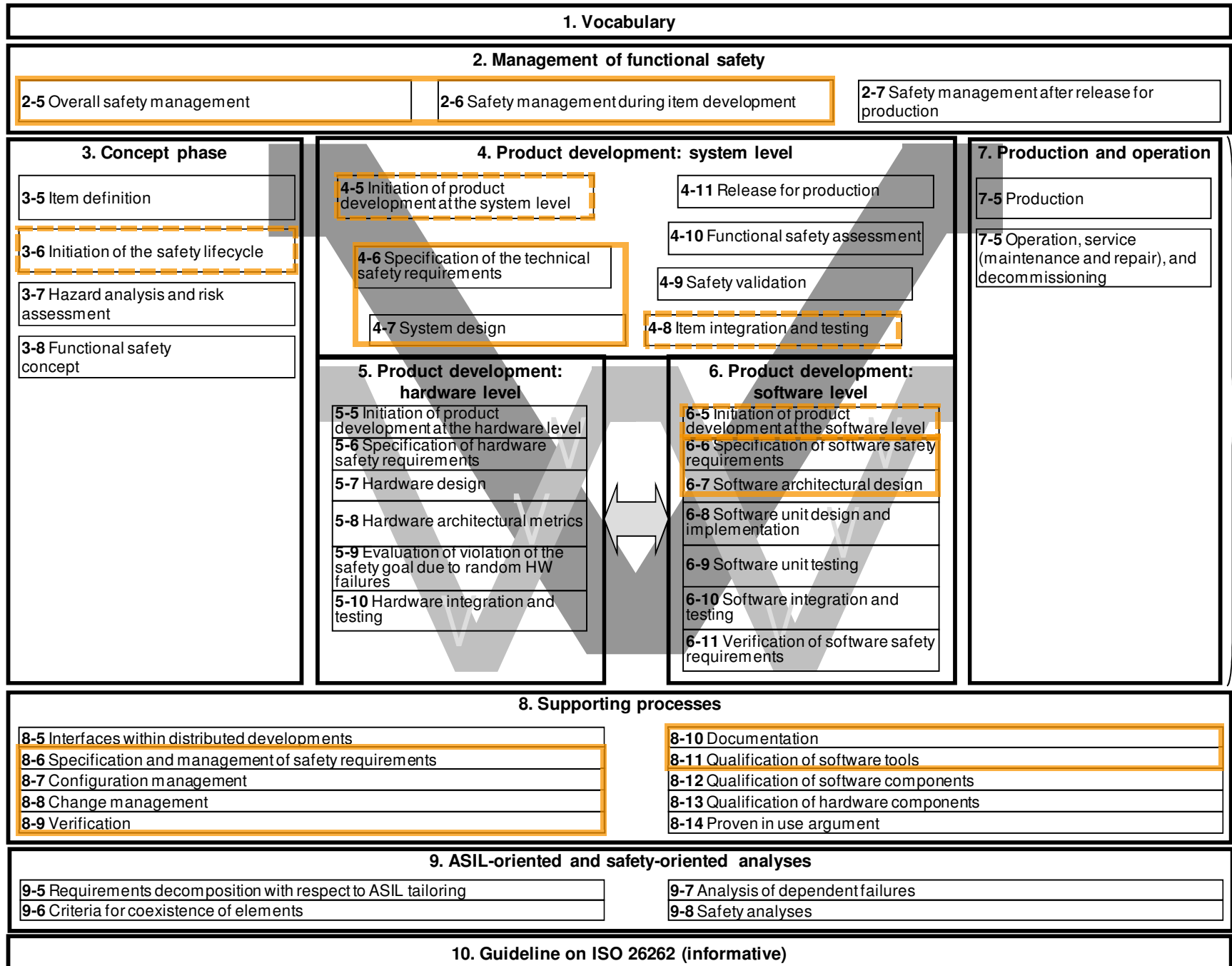
a Static recovery mechanisms can be realised by recovery blocks, backward recovery, forward recovery and recovery through repetition.  
 b Graceful degradation at the software level refers to prioritising functions to minimise the adverse effects of potential failures on functional safety.  
 c For parallel redundancy to be independent there has to be dissimilar software in each parallel path.

# ISO 26262 and AUTOSAR.

## Safety lifecycle for AUTOSAR as SEooC.



Chapters to  
 be considered  
 by AUTOSAR



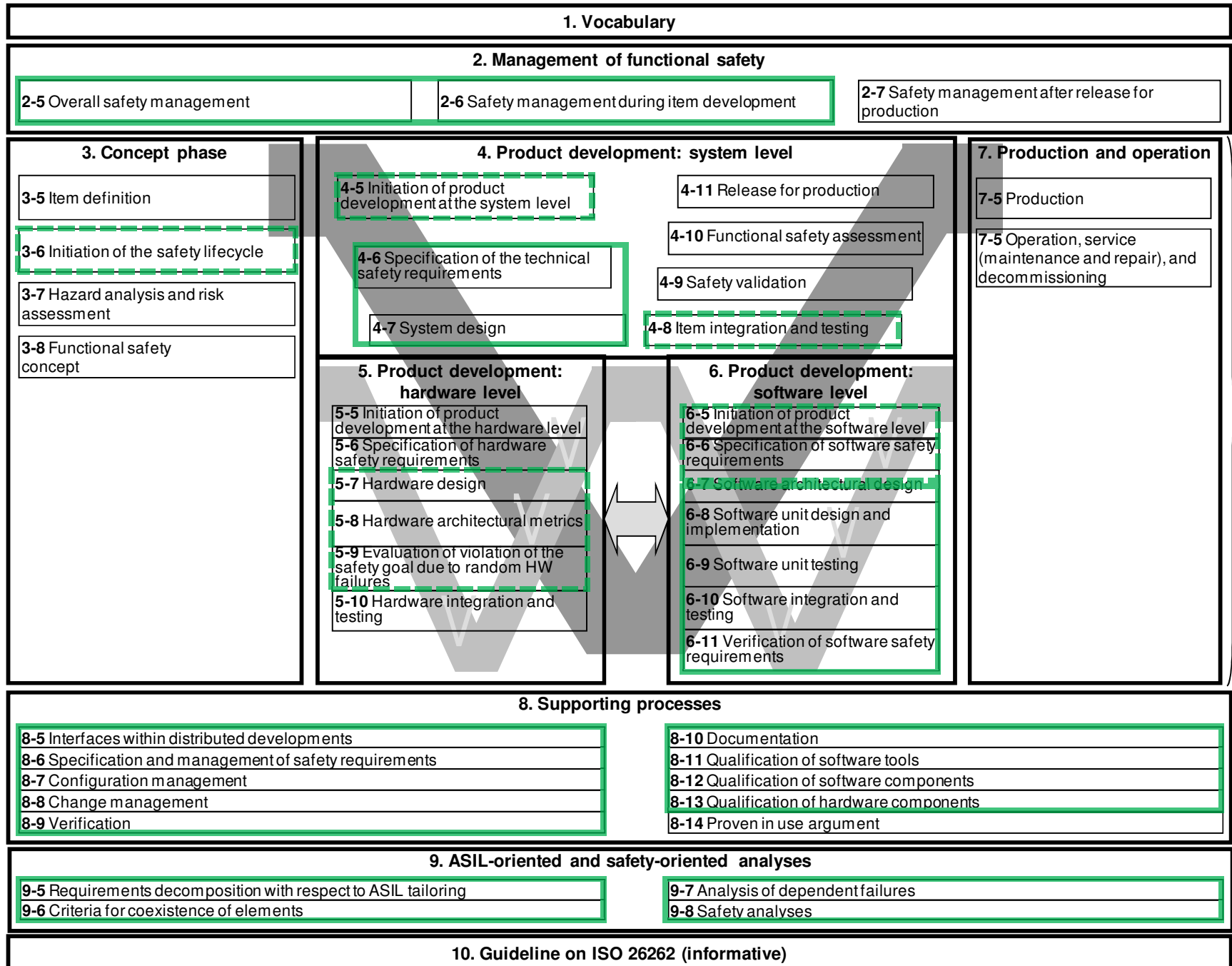
Core processes

# ISO 26262 and AUTOSAR.

## Safety lifecycle for Implementers of AUTOSAR.



Chapters to be considered by Implementers



Core processes

# ISO 26262 and AUTOSAR.

## Conclusion.

- AUTOSAR systematically derived safety mechanisms supported in R4.0
- AUTOSAR provides support for dedicated safety mechanisms with generic fault models
- Implementers have to tailor ISO 26262 according to their activities in the safety-lifecycle
- For all implemented safety mechanisms a safety manual is needed containing
  - The fault model according to which the safety mechanism was developed
  - The constraints that must be fulfilled when applying a safety mechanism
- During system and software design the safety manual is considered to appropriately use the safety mechanisms of an AUTOSAR implementation.
- **Safety related systems can be built with AUTOSAR compliant systems.**

# ISO 26262 und AUTOSAR.

## Thank you for your attention.

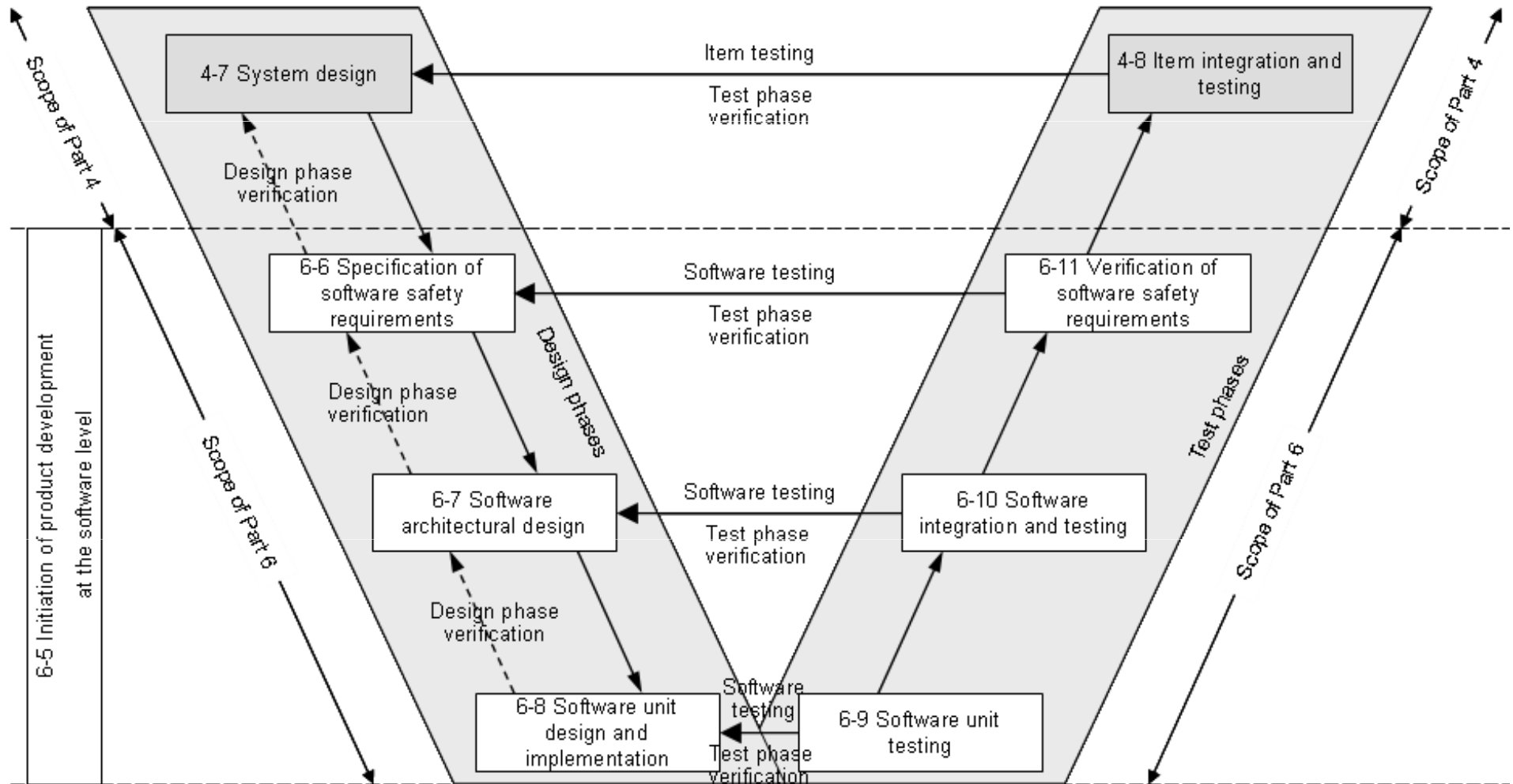


# ISO 26262 and AUTOSAR.

## Backup.

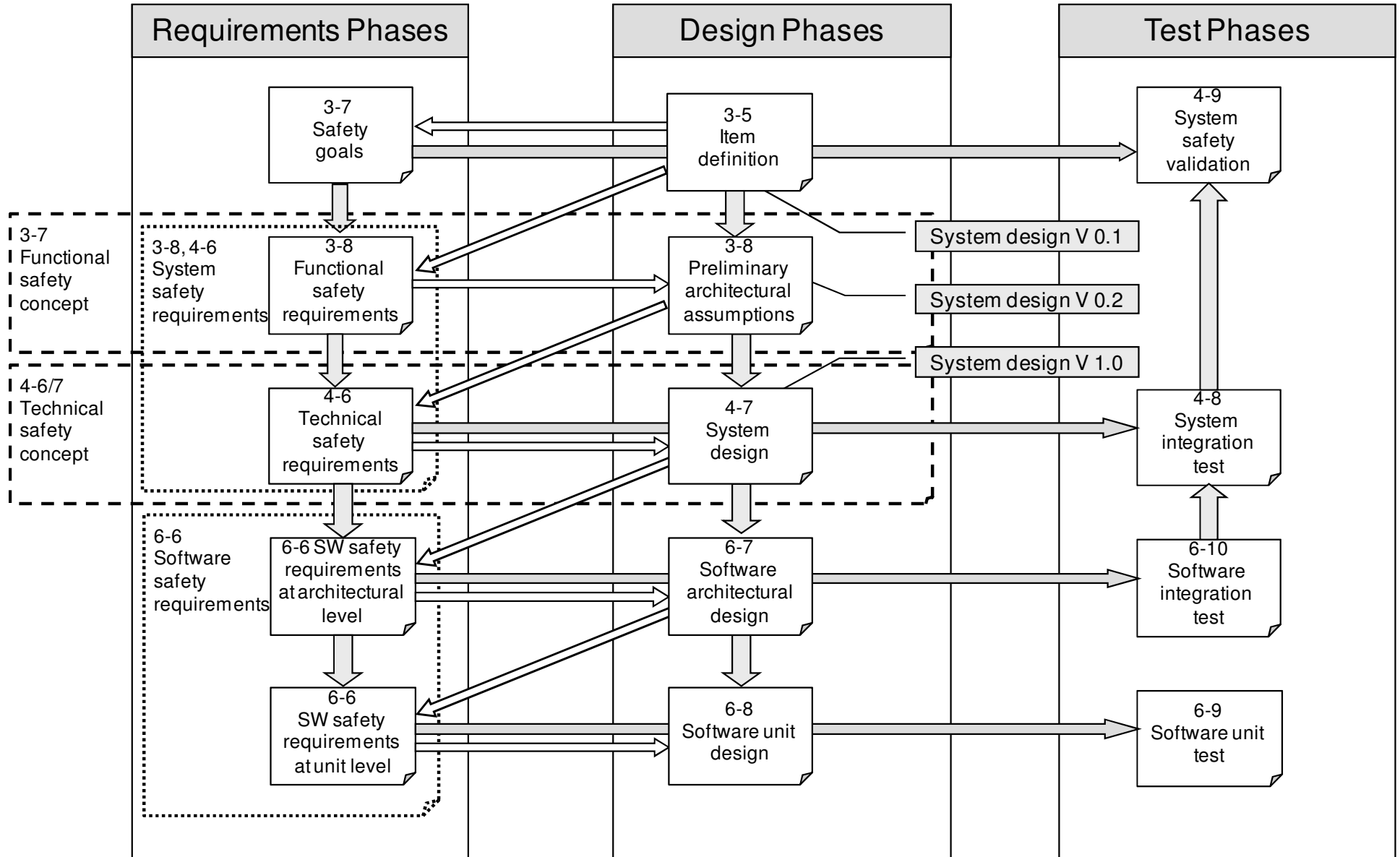
# ISO 26262 and AUTOSAR.

## Software reference phase model.



# ISO 26262 and AUTOSAR.

## Requirements, design and test phases.



⇌ Requirements / design interaction

➡ Requirements, design and testflow