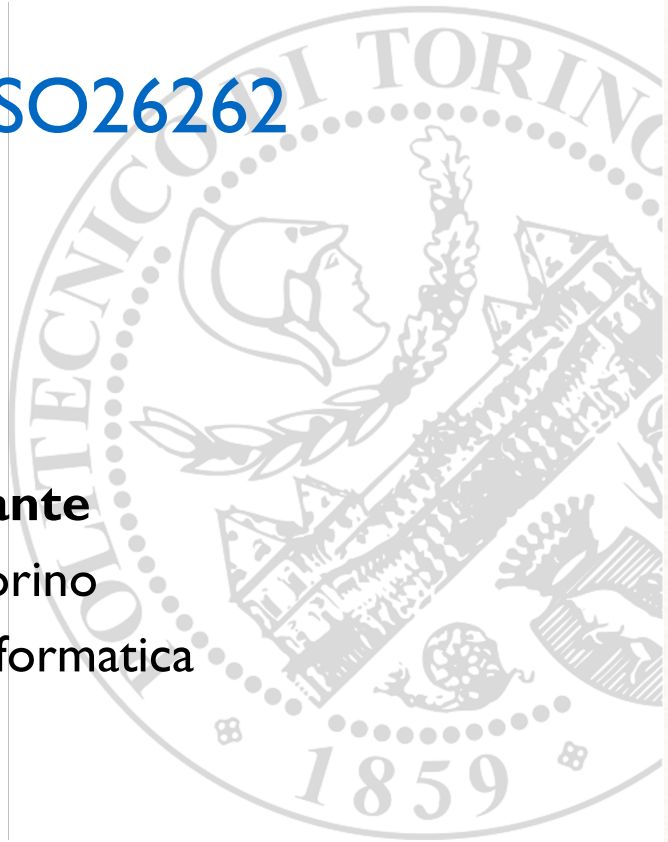
Introduction to ISO26262

Massimo Violante
Politecnico di Torino
Dip. Automatica e Informatica



Summary

- Introduction
- Structure & vocabulary
- The concept phase
- The implementation phases

Summary

- Introduction
- Structure & vocabulary
- The concept phase
- The implementation phases

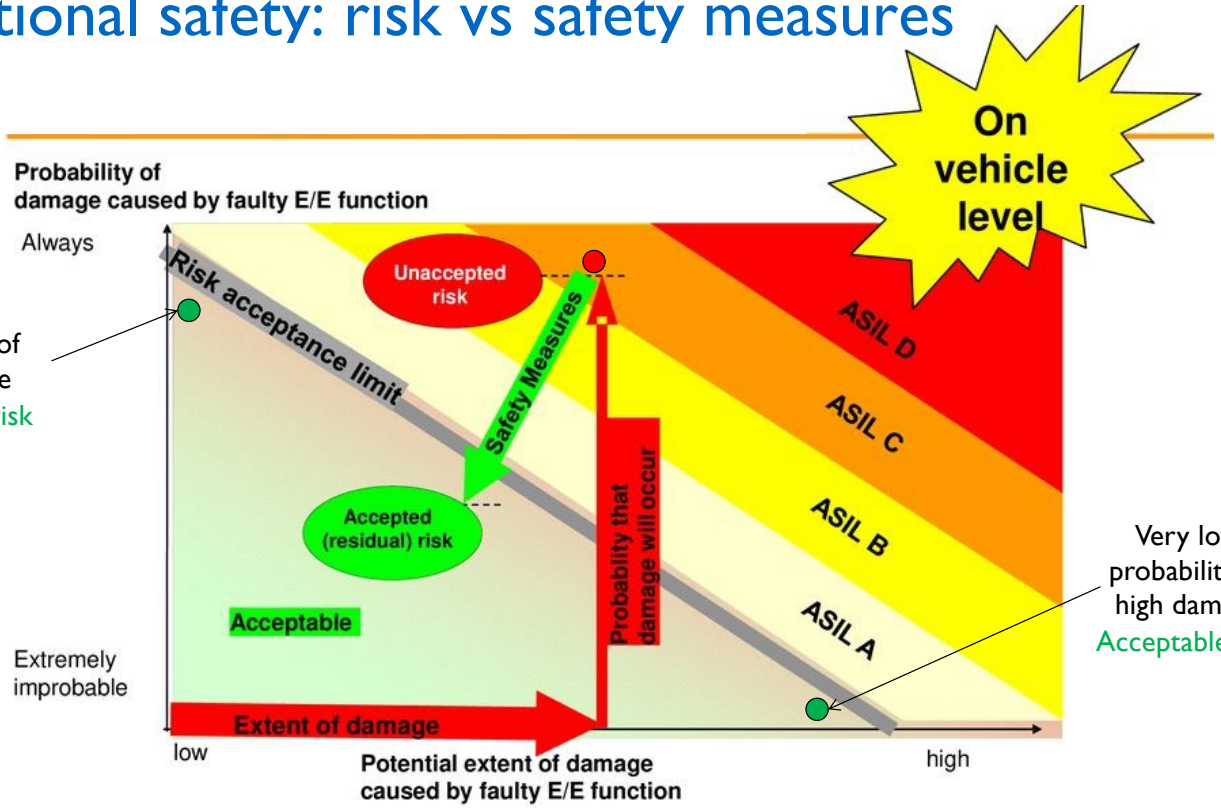
ISO26262

- ISO-26262 is an international standard specific for the automotive industry
- It applies to safety-related road vehicle electronic and electrical (E/E) systems, and addresses hazards due to malfunctions
- It provides requirements for the whole lifecycle of the E/E system (incl. H/w and S/w components)
  - It details how the system has to be conceived, which documents have to be produced, which design activity must be performed

ISO26262

- The requirements for the E/E system development depend on the risk for the customer
- Risk is determined based on customer risk by identifying the so-called Automotive Safety Integrity Level (ASIL) associated with each undesired effects
- ISO-26262 is focused on functional safety, i.e., the part of the overall safety of a system that depends on the system operating correctly in response to its inputs, including the safe management of likely operator errors, hardware failures and environmental changes
- Its objective is the freedom from unacceptable risk of physical injury or of damage to the health of people either directly or indirectly (through damage to property or to the environment)

Functional safety: risk vs safety measures



Very high probability of low damage
Acceptable risk

Very low probability of high damage
Acceptable risk

Basics of Functional Safety

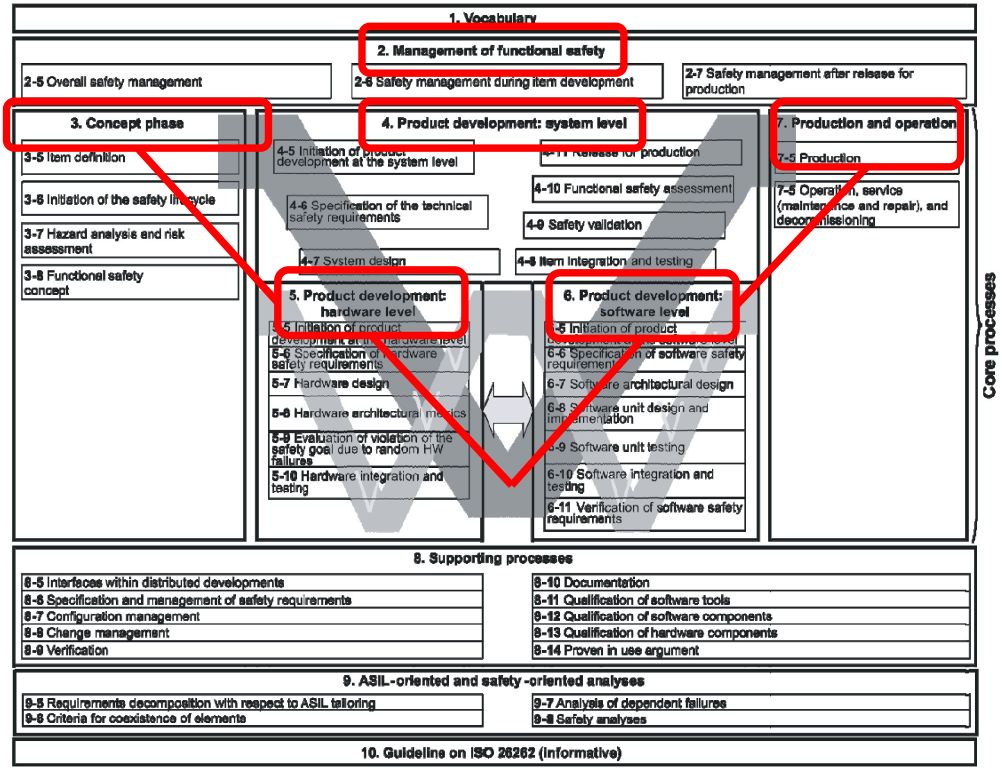
Functional Safety is the way to determine the risk of using complex and simple circuit to perform a safety function. The safety function must always be performed under normal/undisturbed conditions and under fault conditions

Functional Safety is achieved when there is the absence of unreasonable risk due to hazards caused by the malfunctioning of electrical / electronic systems

- Functional safety deals with what happens when systems responsible for safety don't work
- Functional safety is about the consequences of malfunctions

level : system, hardware and software, production

ISO26262: V model



Summary

- Introduction
- Structure & vocabulary
- The concept phase
- The implementation phases

ISO26262 structure

1. Vocabulary
2. Management of Functional Safety
3. Concept Phase
4. Product Development: System Level
5. Product Development: Hardware Level
6. Product Development: Software Level
7. Production and Operation
8. Supporting Processes
9. ASIL-oriented and Safety-oriented Analyses
10. Guidelines on ISO 26262

- only electrical/electronic are relevant -

Vocabulary

(target of our activity)

- Item: system or array of systems or a function to which ISO 26262 is applied
- Harm: physical injury or damage to the health of people
- Severity: measure of the extent of harm to an individual
- Failure: termination of the ability of an element or an item to perform a function as required
- Risk: combination of the probability of occurrence of harm and the severity of that harm

how to manage item (malfunction)

Vocabulary

- Controllability: avoidance of the specified harm or damage through the timely reaction of the persons involved quantify the risk
- Exposure: state of being in an operational situation that can be hazardous if coincident with the failure mode under analysis quantify the probabilily to stay on a risk situation
- Automotive Safety Integrity Level (ASIL): one of the four levels to specify the item's necessary requirements of ISO 26262 and safety measures for avoiding an unreasonable residual risk, with D representing the most stringent and A the least one

evaluation of the risk : value

Vocabulary

- Hazard: potential source of harm

procedure :

- Hazard Analysis and Risk Assessment: method to identify and categorize hazardous events of items and to specify safety goals and ASILs related to the prevention or mitigation of these hazards in order to avoid unreasonable risk

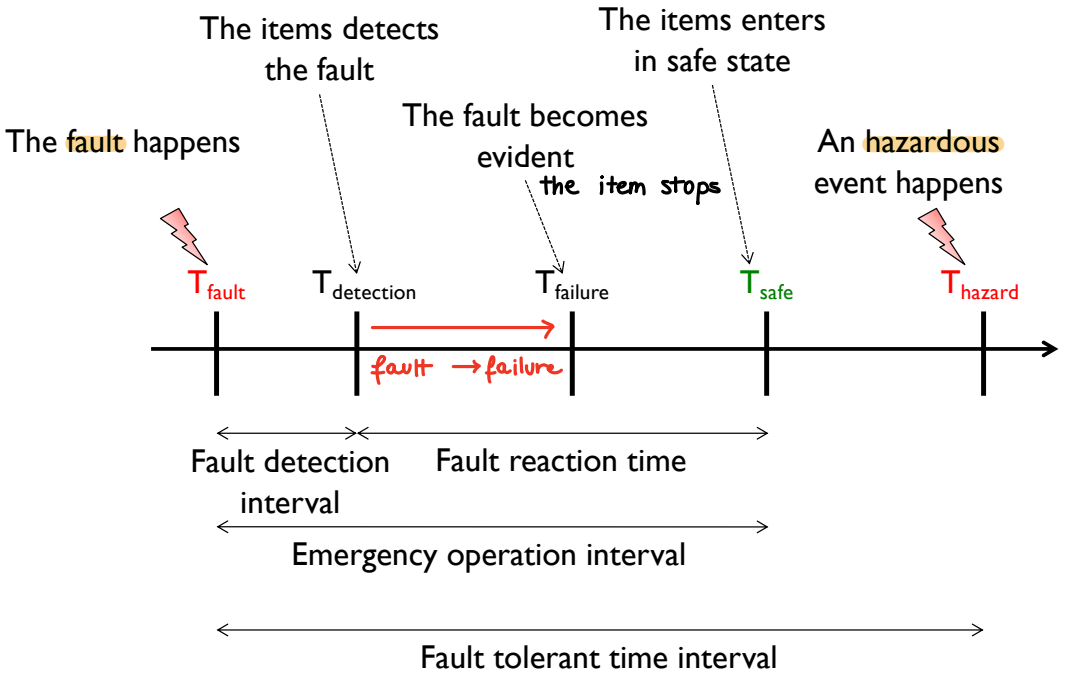
- Safety: absence of unreasonable risk

- Safety Goal: top-level safety requirement as a result of the hazard analysis and risk assessment

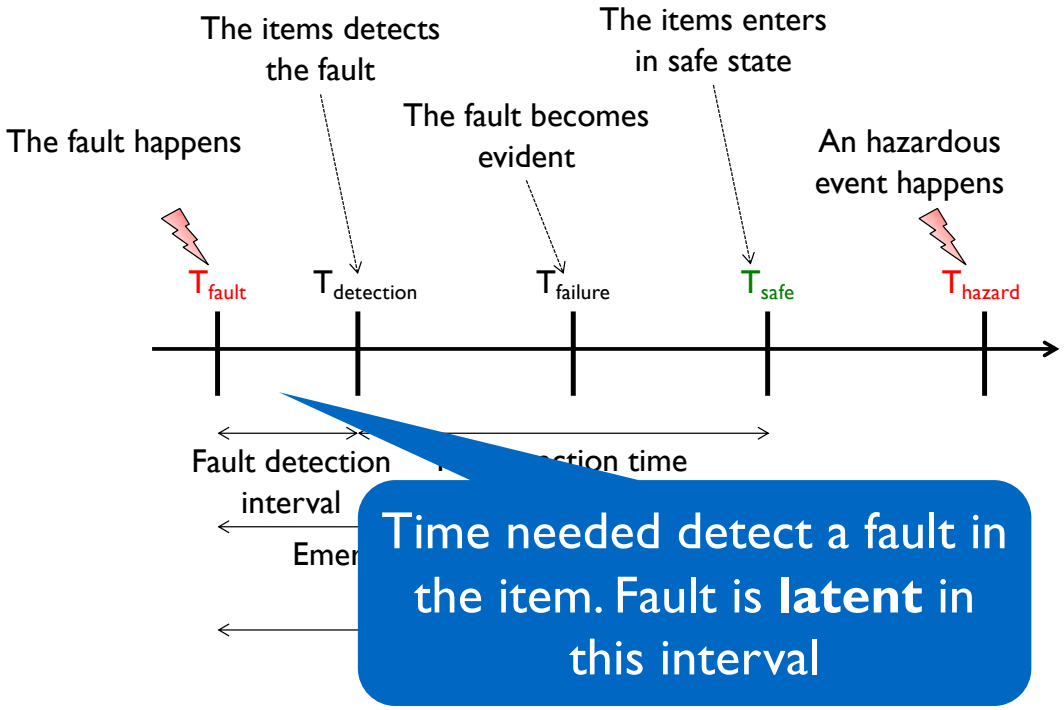
Vocabulary

- Safety element out of context (SEooC): it is a safety-related element that is not developed for a specific item (i.e. in the context of a particular vehicle). A SEooC can be a system, an array of systems, a subsystem, a software component or a hardware component
  - A given ASIL is assumed for the SEooC
  - All the development is based on the assumed ASIL
  - The assumption shall be validated when the SEooC is integrated into an item

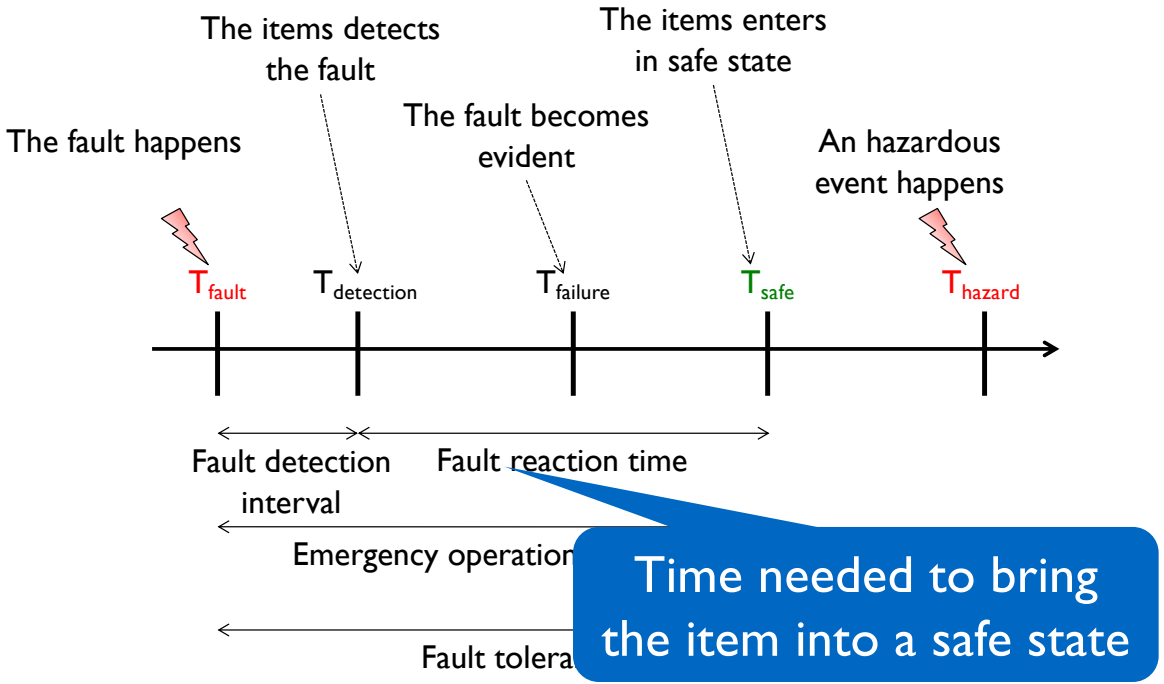
Timeline


The items detects the fault
The items enters in safe state
The fault becomes evident
the item stops
The fault happens
An hazardous event happens
T fault
T detection
T failure
T safe
T hazard
fault → failure
Fault detection interval
Fault reaction time
Emergency operation interval
Fault tolerant time interval

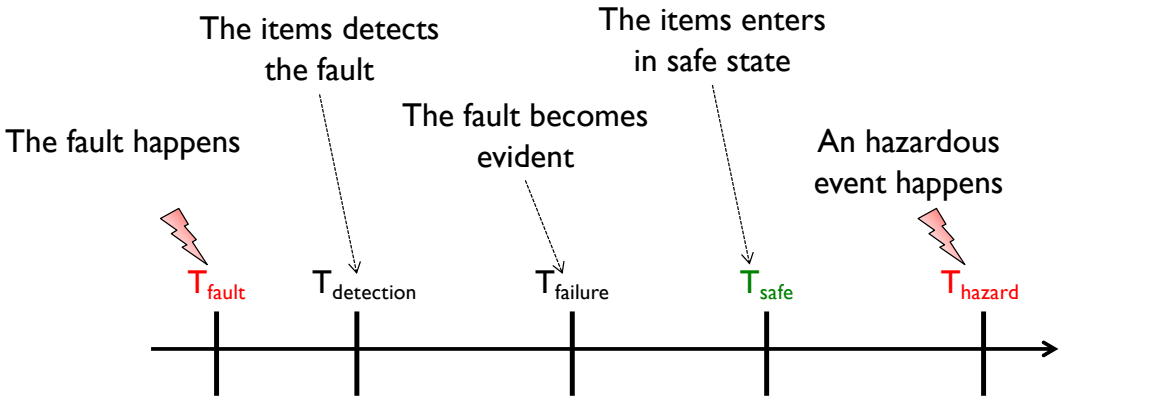
Timeline


The items detects the fault
The items enters in safe state
The fault becomes evident
The fault happens
An hazardous event happens
T fault
T detection
T failure
T safe
T hazard
Fault detection interval
Time needed detect a fault in the item. Fault is **latent** in this interval

Timeline

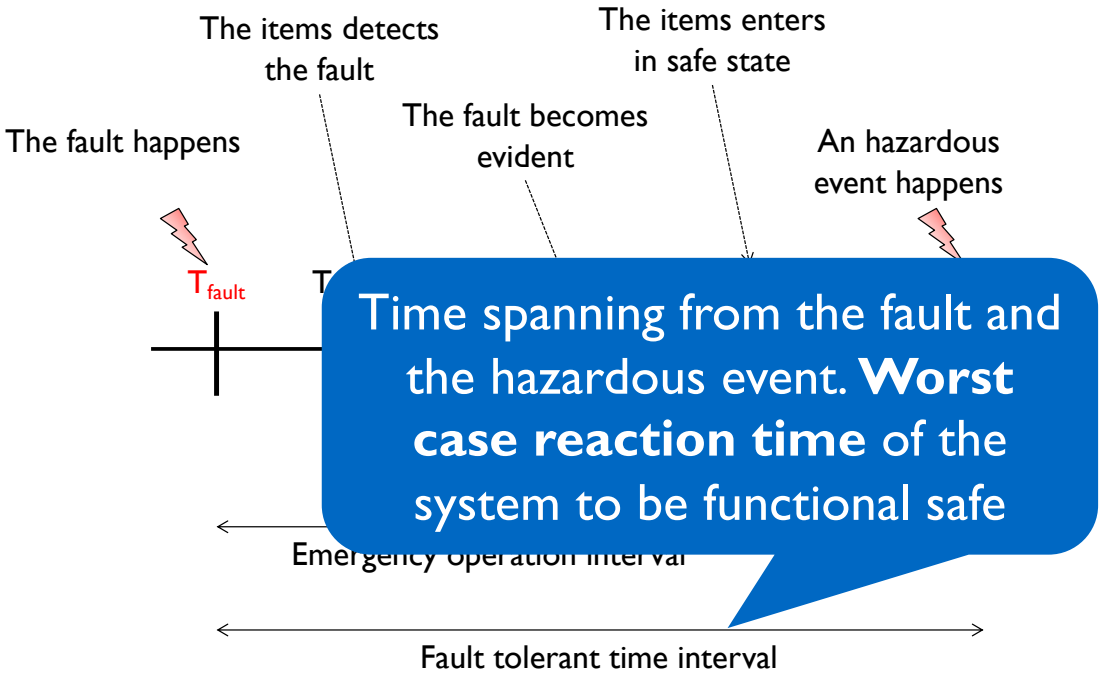

The items detects the fault
The items enters in safe state
The fault becomes evident
The fault happens
An hazardous event happens
T fault
T detection
T failure
T safe
T hazard
Fault detection interval
Fault reaction time
Emergency operation
Fault tolera
Time needed to bring the item into a safe state

Timeline



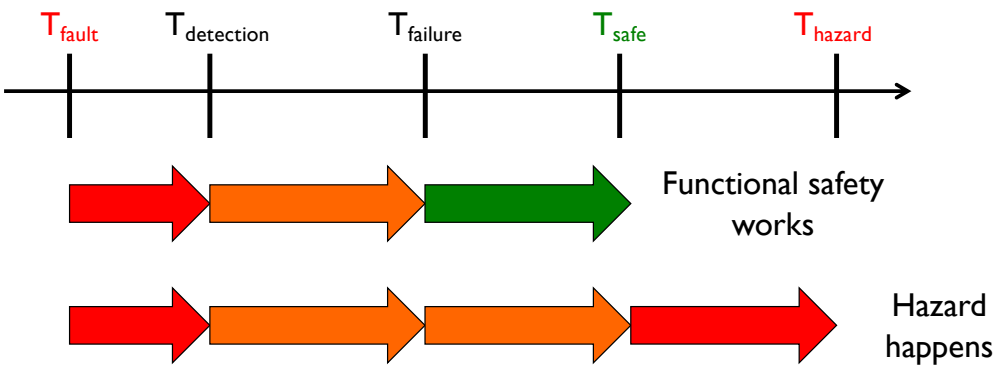
Time during with the item is unsafe

Timeline

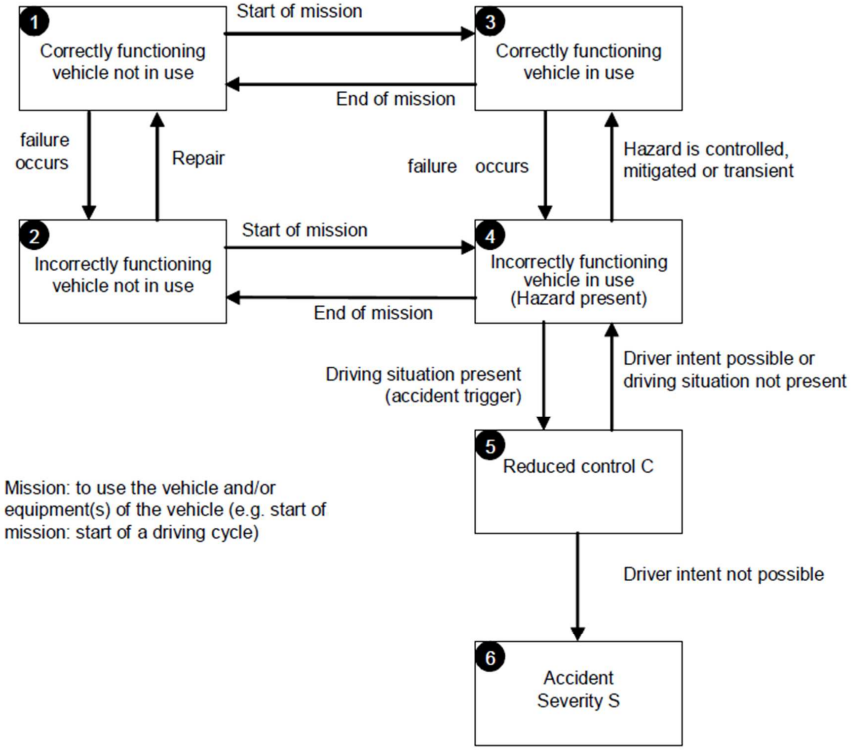

The items detects the fault
The items enters in safe state
The fault becomes evident
The fault happens
An hazardous event happens
T_{fault}
Time spanning from the fault and the hazardous event. **Worst case reaction time** of the system to be functional safe
Fault tolerant time interval

Timeline

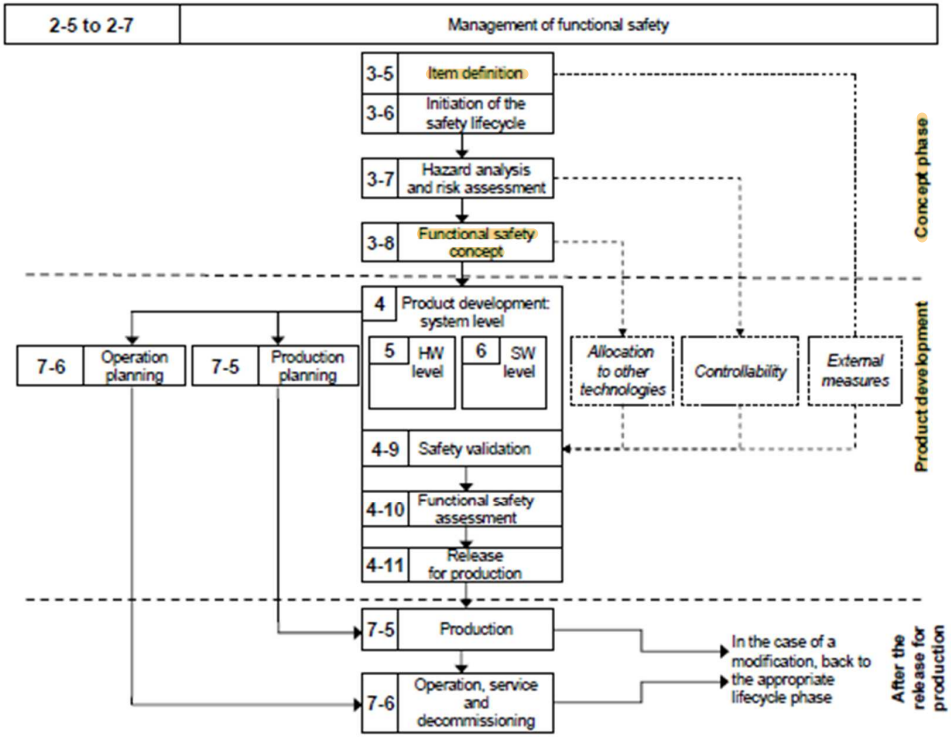
- Possible scenarios



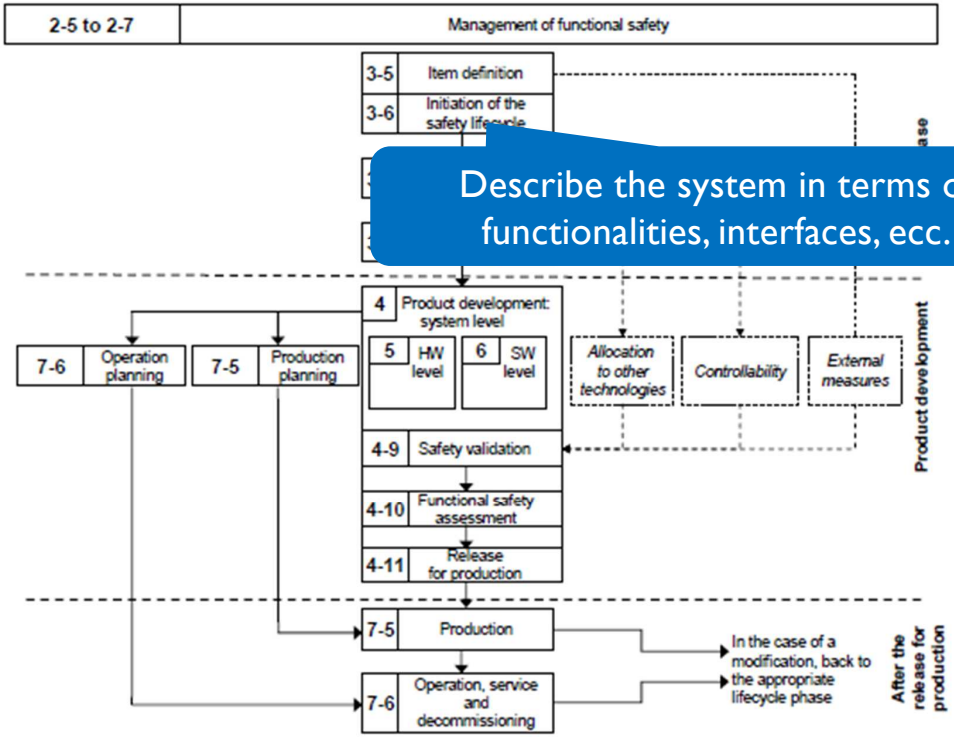
State model of automotive risk



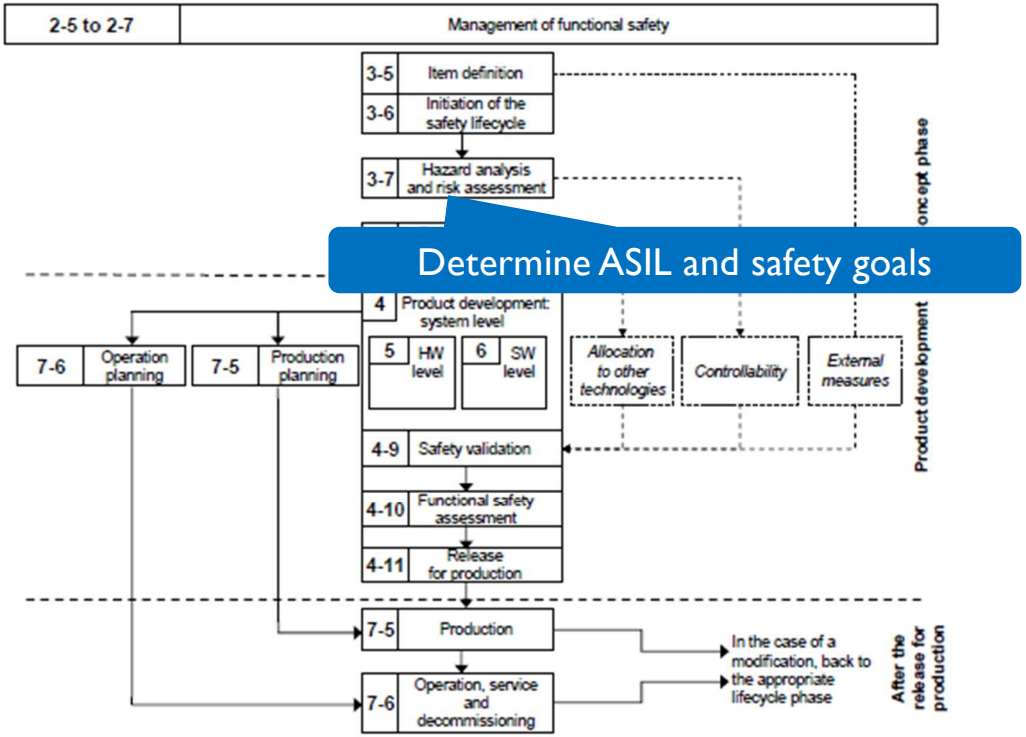
Safety life cycle



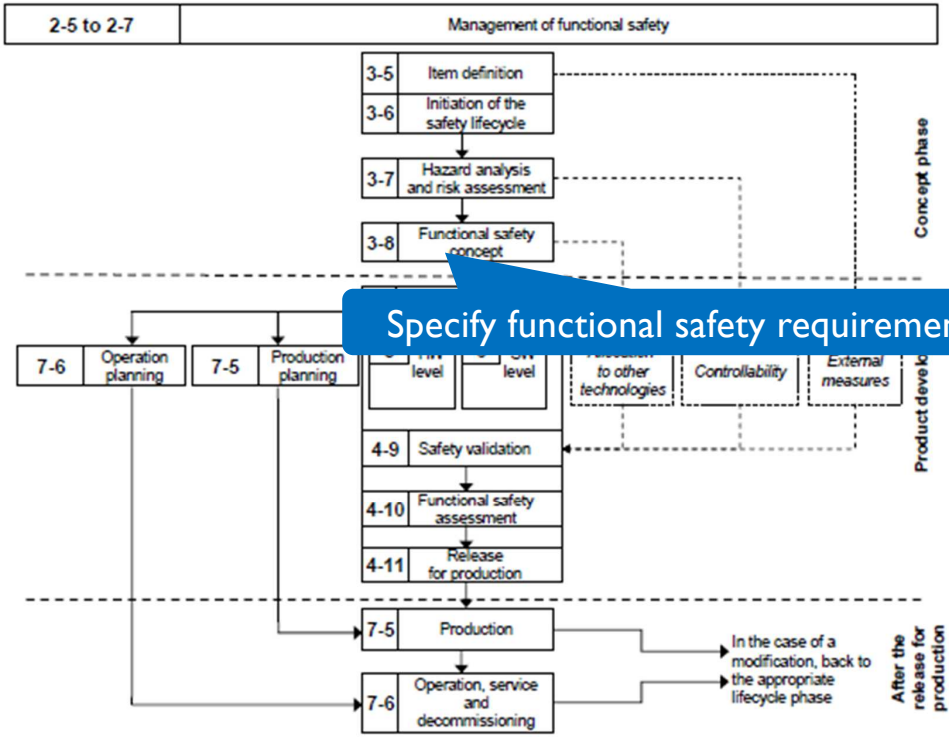
Safety life cycle



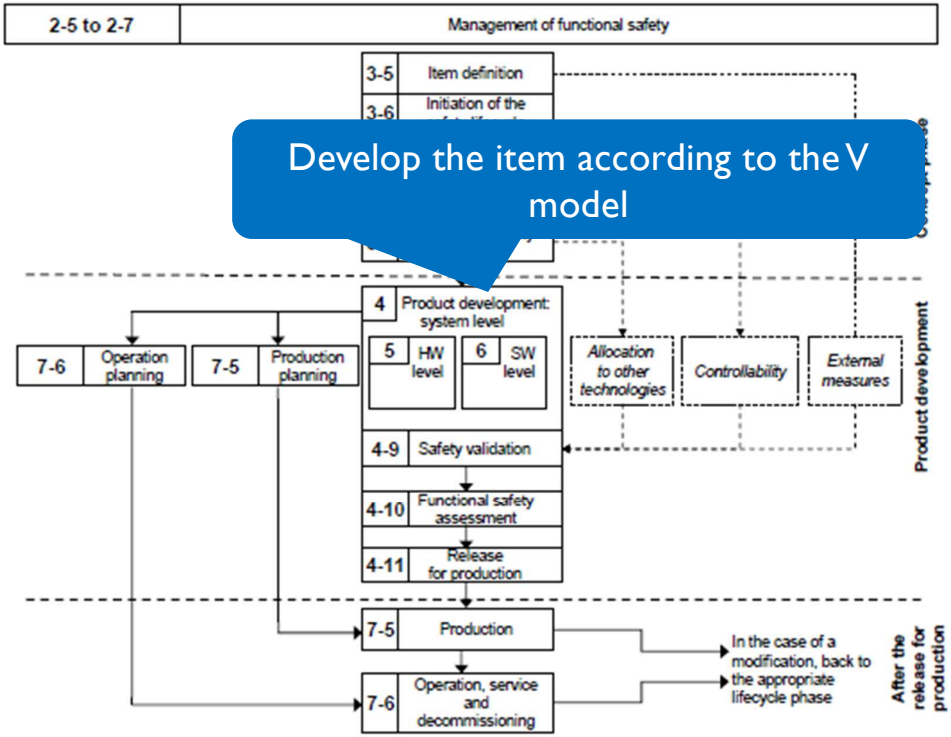
Safety life cycle



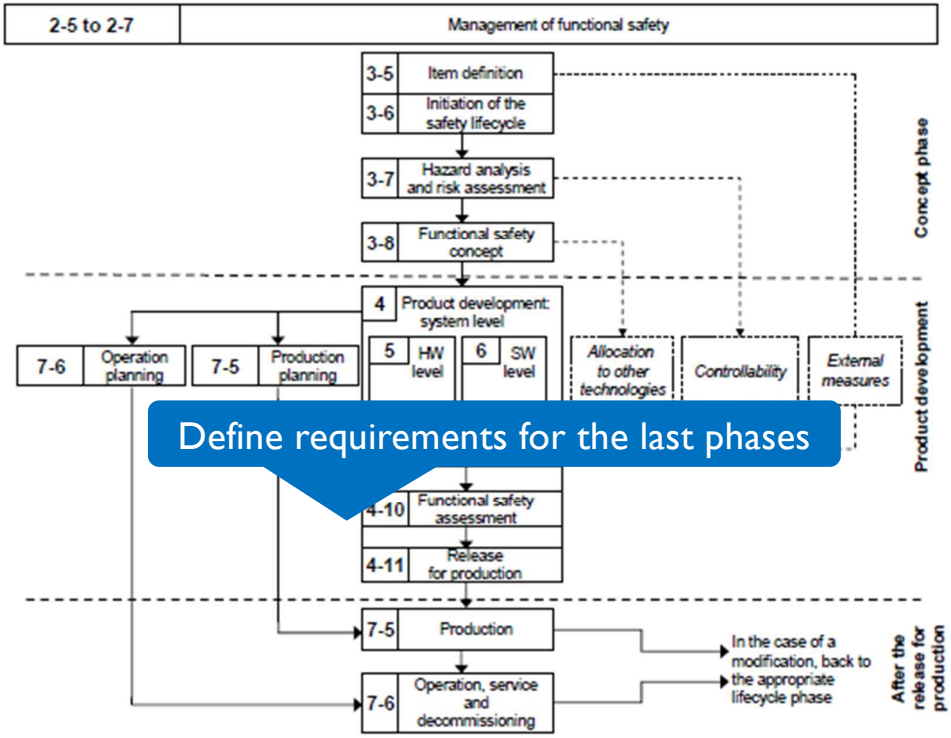
Safety life cycle



Safety life cycle



Safety life cycle



Summary

- Introduction
- Structure & vocabulary
- The concept phase
- The implementation phases

Item definition

- Goal: Define and describe the item, its dependencies on and interaction with the environment and other items
  - Elements of the item
  - Interactions of the item with other items or elements
  - Functionality provided to other items, elements and the environment
  - Functionality required from other items, elements and the environment

Hazard Analysis & Risk Assessment

- Goal: Identify and categorize hazards from malfunctions of item
- Formulate safety goals to prevent or mitigate hazards and avoid unreasonable risk
- Steps:
  - Situational analysis
  - Hazard identification and classification
  - ASIL determination
  - Safety goal determination

Situational analysis

- Describe operational situations and operating modes in which malfunction results in hazardous event

- Example:
  - Driving at low speed
  - Driving at high speed
  - Driving on icy road
  - Driving on wet road
  - …

Hazard identification

- Determine systematically hazards using appropriate techniques
  - Brainstorming, checklists, quality history, FMEA, field studies
- Determine hazardous events for relevant combinations of operational situations and hazards
- Determine consequences of hazardous events

Hazard classification

- Classify (define ASIL) for each hazardous event based on:
  - Severity
  - Exposure
  - Controllability

Severity S

- Measure of the extent of harm to an individual

Class	S0	S1	S2	S3
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

Controllability C

- Avoidance of the specified harm or damage through the timely reaction of the persons involved

Class	C0	C1	C2	C3
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Class	C0	C1	C2	C3
Description	Controllable in general	Simply Controllable	Normally Controllable	Difficult to Control or Uncontrollable
Driving Factors & Scenarios	Controllable in general	99% or more of all drivers or other traffic participants are usually able to avoid harm	90% or more of all drivers or other traffic participants are usually able to avoid harm	Less than 90% of all drivers or other traffic participants are usually able, or barely able, to avoid harm
Situations that are considered distracting	Maintain intended driving path			
Unexpected radio volume increase	Maintain intended driving path			
Warning message - gas low	Maintain intended driving path			
Unavailability of a driver assisting system	Maintain intended driving path			
Faulty adjustment of seat position while driving		Brake to slow/stop vehicle		

Exposure

- Being in an operational situation that can be hazardous if coincident with the failure

Class	E0	E1	E2	E3	E4
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Class	Temporal Exposure			
	E1	E2	E3	E4
Description	Very low probability	Low probability	Medium probability	High probability
Definition	Not specified	Duration (% of average operating time) <1%	1%-10%	>10%
Informative Examples				
Road layout		Mountain pass with unsecured steep slope	One-way street (city street)	Highway
		Country road intersection		Secondary Road
		Highway entrance ramp		Country Road
		Highway exit ramp		
Road surface		Snow and ice on road	Wet road	
		Slippery leaves on road		

ASIL

- QM (quality measures)
- A (least important)
- B
- C
- D (most important)

ASIL determination

(tabular representation of the risk)

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

happen always, can't control, die

Safety goal determination

- Determine a safety goal for each hazardous event with ASIL (not for QM)
  - The safety goal inherits the ASIL of the hazardous event

- Safety goals are top-level safety requirements for the item
  - They lead to the functional safety requirements needed to avoid an unreasonable risk for each hazardous event
  - Safety goals are not expressed in terms of technological solutions, but in terms of functional objectives

Functional safety concept

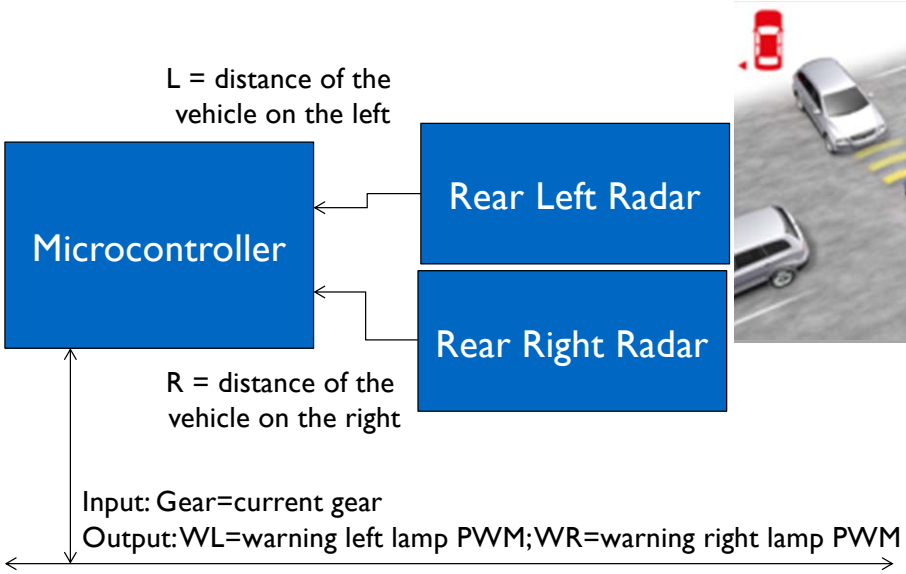
- Goal: Derive the functional safety requirements, from the safety goals. Allocate them to the preliminary architectural elements of the item, or to external measures

- Contains:
  - Safety measures, including the safety mechanisms, implemented in the item's architectural elements and specified in the functional safety requirements

An example

item to design:

- Cross Traffic Assist – CTA
  - The CTA detects incoming obstacles that may hit the vehicle and inform the driver

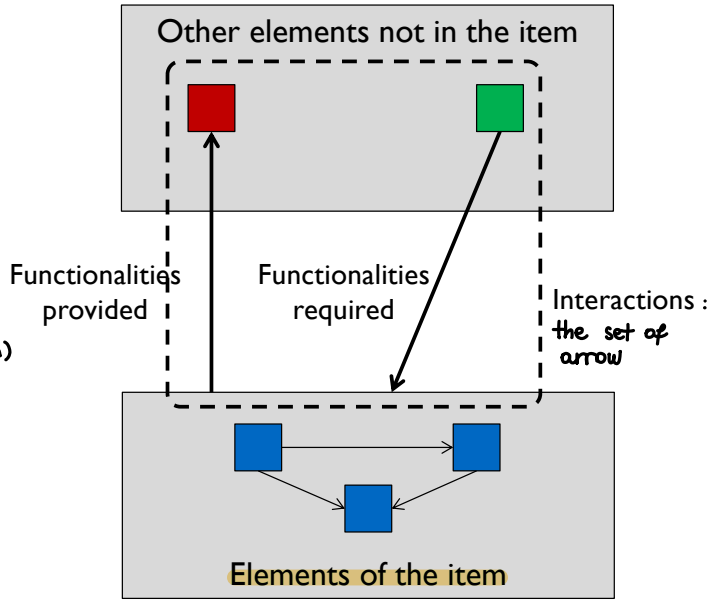


Item definition

1. define
- Elements of the item: microcontroller, left radar, right radar

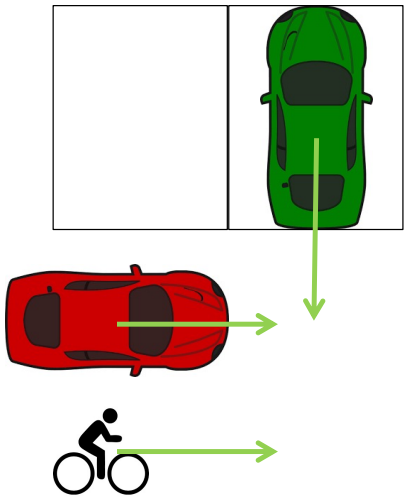
2.
- Interactions of the item with other items: the item interacts with the body computer through the vehicle network for reading the current gear

3. individuation of the obstacle through PWM (pulse with modulation)
- Functionality provided to other items, elements or the environment: turns on/off left/right warning lamps upon detection of incoming obstacles with PWM signal proportional to the distance of the incoming obstacle
- Functionality required from other items, element or the environment: body computer shall provide current gear, turn on/off the warning leds

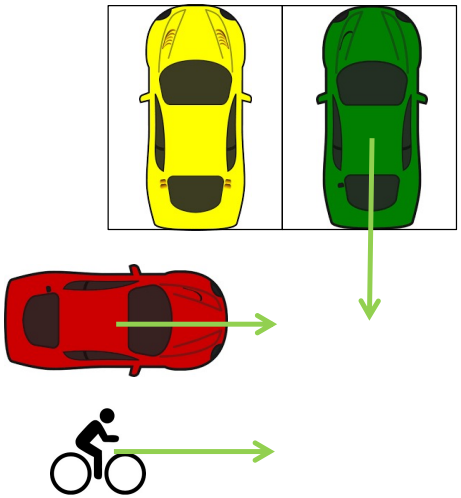


Operational situations

OS1: vehicle exiting from empty parking lot, vehicle approaching



OS2: vehicle exiting from full parking lot, vehicle approaching



OS3: vehicle exiting from empty parking lot, cycle approaching

OS4: vehicle exiting from full parking lot, cycle approaching

Hazard identification

- Based on Failure Mode and Effect Analysis (FMEA)
  (how the failure provide by the item)
  ↳ I know all the elements
  - The item in decomposed in components
  - For each component
    - The possible failure modes are identified
    - The possible effects of the failure on the item are identified

Hazard identification

Component	Failure mode	Effect on the item
Right sensor	FM1: No distance measurement (unresponsive sensor) FM2: Wrong distance measurement: the actual distance is higher than what measured FM3: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Left sensor	FM4: No distance measurement (unresponsive sensor) FM5: Wrong distance measurement: the actual distance is higher than what measured FM6: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Microprocessor	FM7: The microprocessor fails in executing instructions	H1: The item does not report any obstacle
Network	FM8: The network fails (no communication possible)	H1: The item does not report any obstacle

operation situation e Hazard are known:

Hazard Analysis and Risk Assesment

Operational Situation	Hazard		Effect	Comment
OS1	H1	S E C		
OS1	H2	S E C		
OS1	H3	S E C		
OS2	H1	S E C		
OS2	H2	S E C		
OS2	H3	S E C		

Hazard Analysis and Risk Assesment

Operational Situation	Hazard		Effect	Comment
OS3	H1	S E C		
OS3	H2	S E C		
OS3	H3	S E C		
OS4	H1	S E C		
OS4	H2	S E C		
OS4	H3	S E C		

Severity

- Severity of potential harm to persons
  - Damage to things is out of scope of 26262!

- All persons reasonably concerned:
  - Driver
  - Passengers
  - Pedestrians
  - Cyclists
  - Passengers in other vehicle

Severity

- Severity can be classified according to the type of collision and the velocity of the vehicles involved
  - Collision between two cars, Δv is the velocity difference
    - S1 if Δv less than 20 km/hour
    - S2 if Δv between 20 and 40 km/hour
    - S3 if Δv more than 40km/hour

- The severity may depend on the o.s.
  - An accident with a pedestrian or bicycle is almost always serious, at least S2 or S3
    - S2 when operating inside a built-up area (e.g. the city) because the vehicle is likely to be slower
    - S3 when outside a built-up area (faster vehicle speed)

Hazard identification

Component	Failure mode	Effect on the item
Right sensor	FM1: No distance measurement (unresponsive sensor) FM2: Wrong distance measurement: the actual distance is higher than what measured FM3: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Left sensor	FM4: No distance measurement (unresponsive sensor) FM5: Wrong distance measurement: the actual distance is higher than what measured FM6: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Microprocessor	FM7: The microprocessor fails in executing instructions	H1: The item does not report any obstacle
Network	FM8: The network fails (no communication possible)	H1: The item does not report any obstacle

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS1	H1	S=1 E C	Crash at low Dv (inside parking lot) happen
OS1	H2	S=0 E C	Driver stop the car, crah does not happen
OS1	H3	S=1 E C	Crash at low Dv (inside parking lot) happen
OS2	H1	S=1 E C	Crash at low Dv (inside parking lot) happen
OS2	H2	S=0 E C	Driver stop the car, crah does not happen
OS2	H3	S=1 E C	Crash at low Dv (inside parking lot) happen

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS3	H1	S=2 E C	Crash with cyclist at low speed happens
OS3	H2	S=0 E C	Driver stop the car, crah does not happen
OS3	H3	S=2 E C	Crash with cyclist at low speed happens
OS4	H1	S=2 E C	Crash with cyclist at low speed happens
OS4	H2	S=0 E C	Driver stop the car, crah does not happen
OS4	H3	S=2 E C	Crash with cyclist at low speed happens

Exposure

- Hazard and risk analysis identifies operating situations in which a malfunction in the item could cause an accident
- How likely are those operating situations to actually occur?
  - "Starting the car engine": very likely!
  - "Car is hit by a falling meteor": very unlikely!
- We should worry most about those operating situations that are most likely to occur
- We should worry about the probability of exposure to the operating situations

Exposure

- A simplified four-level classification scheme is used for probability of exposure

- Each level is an order of magnitude (i.e. 10 times previous level)
  - E1: Possible, but very low probability (e.g. < 0.1 %)
  - E2: Low probability (not more than 1% of operating time)
  - E3: Medium probability (up to10% of operating time)
  - E4: High probability (from 10% operating time up to "always")

Exposure

- Three approaches can be used to define the exposure:
  - Percentage of operating time
  - Frequency of occurrence
  - Mean time to situation

Percentage of operating time

- Sometimes it is most appropriate to think in terms of "temporal overlap"
  - The failure occurs during the time that the operating situation is present
    - Example: "power sliding door opens while travelling at high speed"

- In these cases, it makes sense to think in terms of percentage of operating time
  - Example: "What percentage of operating time does the vehicle travel at high speed?"

- What is operating time?
  - Usually it means "ignition on"(actually using the vehicle)

Percentage of operating time

26262 Class	E1	E2	E3	E4
Probability	Very low	Low	Medium	High
Definition of duration / probability of exposure	Negligible (not specified)	Less than 1% of operating time	Up to 10% of operating time	More than 10% of operating time
Examples	Highway – lost cargo/obstacle on road Vehicle during jump start	City driving – driving backwards Country road – crossing Highway – exit Overtaking	Night driving on roads without streetlights Wet roads Heavy traffic (stop and go)	Accelerating Braking Steering Parking Driving on highways

Frequency of occurrence

- Sometimes it is easier to think in terms of frequency of occurrence of an operating situation
  - "How many days per year are snow and ice encountered?"
  - "The car is started every time you drive it"
  - "Towing another car happens less than once a year"

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS1	H1	S=1 E=4 C	Crash at low Dv (inside parking lot) happen Parking is done every time the vehicle is used
OS1	H2	S=0 E=4 C	Driver stop the car, crah does not happen Parking is done every time the vehicle is used
OS1	H3	S=1 E=4 C	Crash at low Dv (inside parking lot) happen Parking is done every time the vehicle is used
OS2	H1	S=1 E=4 C	Crash at low Dv (inside parking lot) happen Parking is done every time the vehicle is used
OS2	H2	S=0 E=4 C	Driver stop the car, crah does not happen Parking is done every time the vehicle is used
OS2	H3	S=1 E=4 C	Crash at low Dv (inside parking lot) happen Parking is done every time the vehicle is used

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS3	H1	S=2 E=4 C	Crash with cyclist at low speed happens Parking is done every time the vehicle is used
OS3	H2	S=0 E=4 C	Driver stop the car, crah does not happen Parking is done every time the vehicle is used
OS3	H3	S=2 E=4 C	Crash with cyclist at low speed happens Parking is done every time the vehicle is used
OS4	H1	S=2 E=4 C	Crash with cyclist at low speed happens Parking is done every time the vehicle is used
OS4	H2	S=0 E=4 C	Driver stop the car, crah does not happen Parking is done every time the vehicle is used
OS4	H3	S=2 E=4 C	Crash with cyclist at low speed happens Parking is done every time the vehicle is used

Controllability

- Controllability is an estimation of the probability that the driver or other endangered persons are able to gain control of the hazardous event that is arising and able to avoid the specific harm

- Assumptions:
  - Driver is in normal condition to drive (e.g., not exhausted)
  - Driver is complying with laws and regulations
  - Driver is trained (e.g., has a license)

Controllability

- Controllability for the driver is dependent on
  - The possibility and driver's capability to perceive the criticality of a situation
  - The driver's capability to decide on appropriate countermeasures (e.g., override, system switch-off)
  - The driver's ability to perform the chosen countermeasure (e.g., reaction time, sensory-motor speed, accuracy)

Controllability

- Controllability for other participants is similarly dependent on
  - Ability to perceive the criticality of a situation
  - Ability to decide on appropriate countermeasures (e.g., whether to jump out of the way, whether to swerve to avoid a collision)
  - Ability to perform the chosen countermeasure (e.g., whether the car is moving too fast for a pedestrian to jump out of the way, whether the cars are too close to avoid a collision)

Controllability examples

26262 Class	C0	C1	C2	C3
Description	Generally controllable	Simply controllable	Normally controllable	Difficult or uncontrollable
Definition of controllability	Generally possible to control	99% or more drivers and other participants can avoid harm	90% or more drivers and other participants can avoid harm	Less than 90% of drivers and other participants can avoid harm
Examples	•Unexpected increase in radio volume •Situations that are considered distracting	When starting the vehicle with a locked steering column, the car can be brought to stop by almost all drivers early enough to avoid a specific harm to persons nearby	Driver can normally avoid departing from the lane in case of a failure of ABS during emergency braking	Driver normally cannot bring the vehicle to a stop if a total loss of braking performance occurs

Hazard identification

Component	Failure mode	Effect on the item
Right sensor	FM1: No distance measurement (unresponsive sensor) FM2: Wrong distance measurement: the actual distance is higher than what measured FM3: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Left sensor	FM4: No distance measurement (unresponsive sensor) FM5: Wrong distance measurement: the actual distance is higher than what measured FM6: Wrong distance measurement: the actual distance is smaller than what measured	H1: The item does not report any obstacle H2: The item reports the obstacle as closer than what it actually is H3: The item reports the obstacle more distant than what it actually is
Microprocessor	FM7: The microprocessor fails in executing instructions	H1: The item does not report any obstacle
Network	FM8: The network fails (no communication possible)	H1: The item does not report any obstacle

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS1	H1	S=1 E=4 C=1	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS1	H2	S=0 E=4 C=1	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Simply controllable, good visibility
OS1	H3	S=1 E=4 C=1	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS2	H1	S=1 E=4 C=2	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS2	H2	S=0 E=4 C=2	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS2	H3	S=1 E=4 C=2	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS3	H1	S=2 E=4 C=1	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS3	H2	S=0 E=4 C=1	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Simply controllable, good visibility
OS3	H3	S=2 E=4 C=1	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS4	H1	S=2 E=4 C=2	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS4	H2	S=0 E=4 C=2	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS4	H3	S=2 E=4 C=2	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility

ASIL allocation

- ASIL allocation as prescribed by the standard

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

- Shortcut

Sum of Parameters	ASIL
6 or less	QM
7	A
8	B
9	C
10	D

ASIL allocation

Operational Situation	Hazard	Effect	Comment
OS1	H1	S=1 E=4 QM C=1	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS1	H2	S=0 E=4 QM C=1	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Simply controllable, good visibility
OS1	H3	S=1 E=4 QM C=1	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS2	H1	S=1 E=4 A C=2	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS2	H2	S=0 E=4 QM C=2	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS2	H3	S=1 E=4 A C=2	Crash at low Dv (inside parking lot) happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility

Hazard Analysis and Risk Assesment

Operational Situation	Hazard	Effect	Comment
OS3	H1	S=2 E=4 A C=1	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS3	H2	S=0 E=4 QM C=1	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Simply controllable, good visibility
OS3	H3	S=2 E=4 A C=1	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Simply controllable, good visibility
OS4	H1	S=2 E=4 B C=2	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS4	H2	S=0 E=4 QM C=2	Driver stop the car, crah does not happen Parking is done every time the vehicle is used Normally controllable, although reduced visibility
OS4	H3	S=2 E=4 B C=2	Crash with cyclist at low speed happens Parking is done every time the vehicle is used Normally controllable, although reduced visibility

Safety Goals

- The safety goal describes in functional terms what must be done to mitigate the hazard
- The ASIL is the expression of how critical it is that the safety goal be achieved
- SG1: the item shall report obstacles if any, and the distance to the obstacle shall be reported correctly
- SG2: in case the item is not able to report obstacles it shall transit to the safe state

Safe state

working condition

(inform the user that he can't use it)

- The safe state is a key concept in all safety-related standards
  - "A safe state is a state of the system without any unacceptable risk caused by the system"
- This does not mean "the desired state"
  - A vehicle that is standing still is (usually) safe, but it is not necessarily in the desired state (e.g. running well)
- Safe state: the items shall be disabled and the driver must be informed properly

Functional safety concept

- The item is ASIL B, it can produce harm but hazardous situation is moderately unlikely
- Design overhead to guarantee functional safety is needed, but only low-cost solutions are justified
- FSC1: two redundant radars shall be used on each side of the vehicle, in case of mismatching reading the item shall transit to the safe state

redundancy : cost

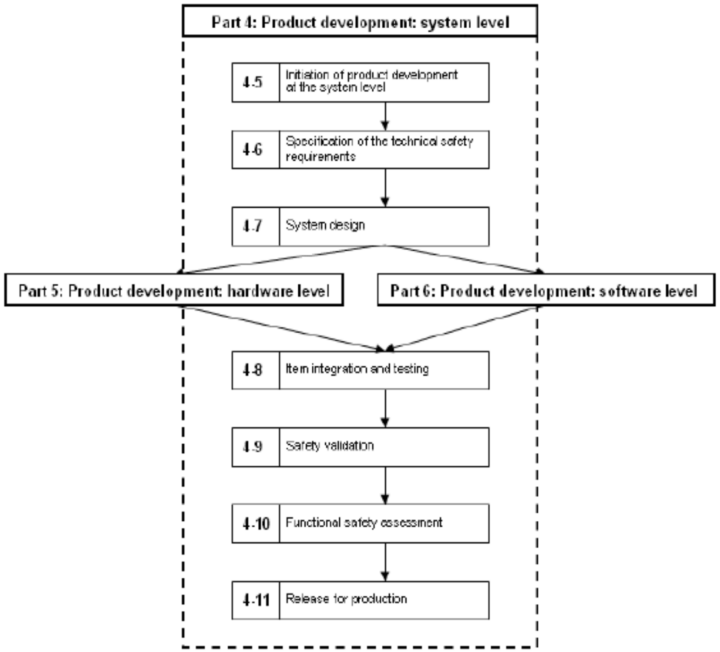
- FSC2: item shall periodically check the network and microprocessor integrity, in case of error detection the item shall transit to the safe state

Summary

- Introduction
- Structure & vocabulary
- The concept phase
- The implementation phases

The implementation phases

- The standard foresees two parallel implementation paths
  - Product development: hardware level
  - Product development: software level
- For each implementation path specific prescriptions are given about
  - Methods to be used
  - Measures to be collected

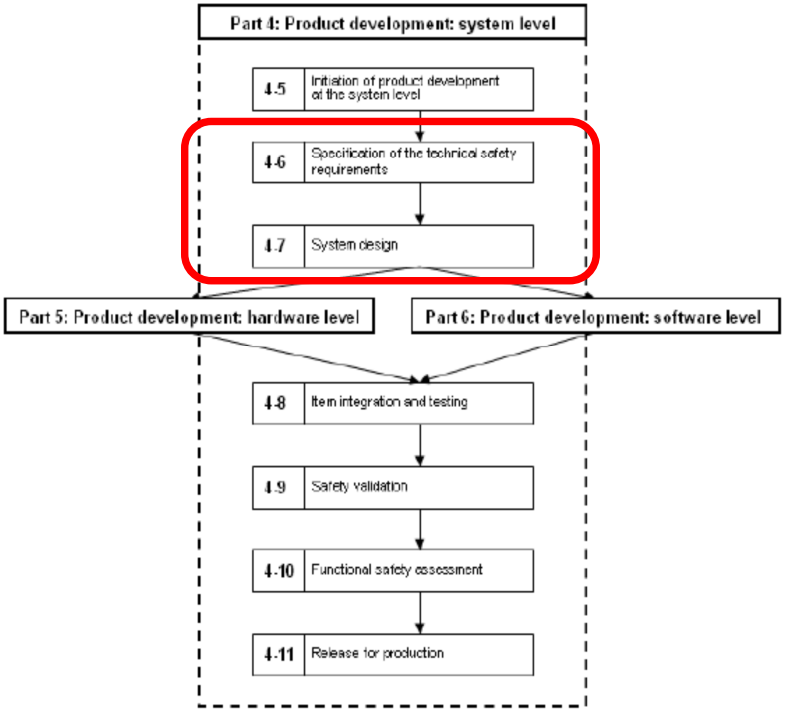


Methods and Measures

- Prescriptions for the different phases are described using tables as follows
  - Each method is either a consecutive entry (e.g. 1, 2, 3) or an alternative entry (e.g. 2a, 2b, 2c)
  - For consecutive entries all methods shall be applied as recommended in accordance with the ASIL
  - For alternative entries, an appropriate combination of methods shall be applied in accordance with the ASIL indicated. The methods with the higher recommendation should be preferred.

Methods and Measures		**ASIL**			
		A	B	C	D
1a	Method 1	o	+	++	++
1b	Method 2	+	+	+	+
2a	Method 3	o	o	+	+
2b	Method 4	o	+	++	++
3	Method 5	o	o	+	+

System level



Spec. of technical safety requirements

- Goal:
  - Specify the technical safety requirement, as a refinement of the functional safety concept, considering both the functional concept and the preliminary architectural assumptions
  - Verify through analysis that the technical safety requirements comply with the functional safety requirements

Spec. of technical safety requirements

- Shall specify:
  - Specify the response of the system to stimuli that affect the achievement of safety goals (failures, stimuli)
  - Specify the necessary safety mechanisms including
    - The measures relating to the detection, indication and control of faults in the system itself
    - The measures relating to the detection, indication and control of faults in external devices that interact with the system
    - The measures that enable the system to achieve or maintain a safe state
    - The measures to detail and implement the warning and degradation concept; and…
    - The measures which prevent faults from being latent

Spec. of technical safety requirements

- For each safety mechanism we shall specify:
  - The transition to the safe state
  - The fault tolerant time interval
  - The emergency operation interval, if the safe state cannot be reached immediately
  - The measures to maintain the safe state

System design

- Goal:
  - Develop the system design and the technical safety concept that comply with the functional requirements and the technical safety requirements specification of the item
  - Verify that the system design and the technical safety concept comply with the technical safety requirements specification

System design

- Activities to be performed:
  - Allocate technical safety requirements to the system design elements
    - To hardware, to software, or both
    - Define hw/sw interaction
  - The system design shall implement the technical safety requirements
  - Each element shall inherit the highest ASIL from the technical safety requirements that it implements
  - Safety analysis shall be applied

HW/SW interaction

- It shall include:
  - The relevant operating modes of hardware devices and the relevant configuration parameters
  - The hardware features that ensure the independence between elements and that support software partitioning
  - Shared and exclusive use of hardware resources
  - The access mechanism to hardware devices
  - The timing constraints defined for each service involved in the technical safety concept

HW/SW interaction

- The relevant diagnostic capabilities of the hardware and their use by the software shall be specified:
  - The hardware diagnostic features shall be defined
  - The diagnostic features concerning the hardware, to be implemented in software, shall be defined

HW/SW interaction

- Interface elements to be considered:
  - Memory
  - Bus interfaces (e.g. Controller Area Network (CAN), Local Interconnect Network (LIN), internal High-Speed Serial Link (HSSL))
  - Converter
  - Multiplexer
  - Electrical I/O
  - Watchdog

HW/SW interaction

- Interface characteristics to be considered:
  - Interrupts
  - Data integrity
  - Initialization
  - Message transfer
  - Network modes
  - Memory management
  - Real-time counter

Verification

- Verify through analysis that the technical safety requirements comply with the functional safety requirements

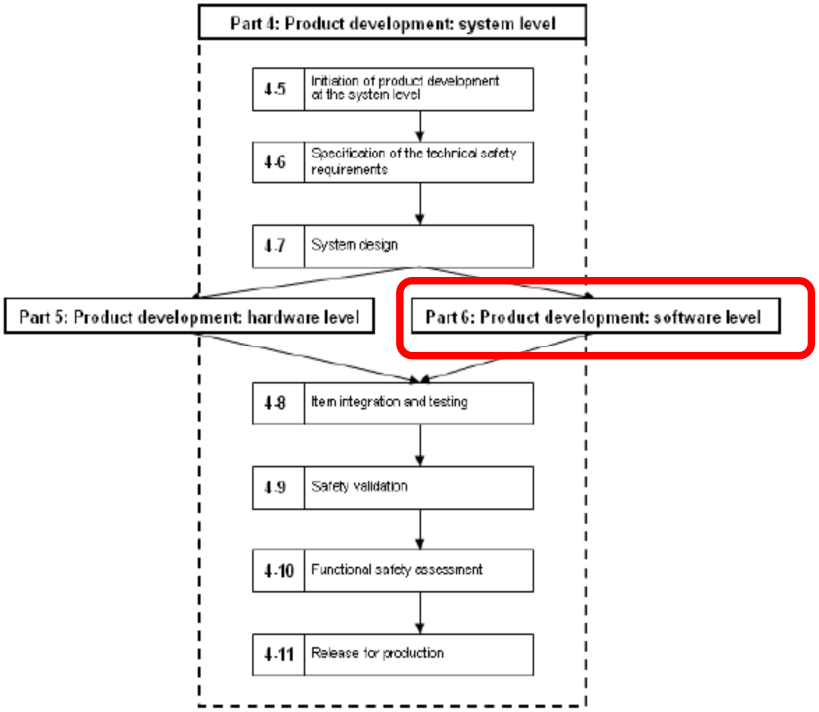
Methods		ASIL			
		A	B	C	D
1a	System design inspection[a]	+	++	++	++
1b	System design walkthrough[a]	++	+	o	o
2a	Simulation[b]	+	+	++	++
2b	System prototyping and vehicle tests[b]	+	+	++	++
3	System design analyses[c]	see Table 1			

[a] Methods 1a and 1b serve as check of complete and correct implementation of the technical safety requirements.

[b] Methods 2a and 2b can be used advantageously as a fault injection technique.

[c] For conducting safety analyses, see ISO 26262-9: —, Clause 8 (Safety analyses).

Product development: sw level



Product development: sw level

- It is composed of the following phases:
  - Initiation
  - Specification of software safety requirements
  - Software architectural design
  - Software unit design and implementation
  - Software unit testing
  - Software integration and testing
  - Verification of software safety requirement

Initiation

- Goal:
  - Plan the functional safety activities during sub phases of software development
  - Determine methods, guidelines, tools considering qualified software tools, qualified software components, design and coding guidelines

Initiation

- Select suitable modeling or programming language considering:
  - An unambiguous definition
  - The support for embedded real time software and runtime error handling
  - The support for modularity, abstraction and structured constructs
  - Define coding and programming guidelines

Initiation

Table 1 — Topics to be covered by modelling and coding guidelines

	Topics	ASIL A	ASIL B	ASIL C	ASIL D
1a	Enforcement of low complexity[a]	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++

[a] An appropriate compromise of this topic with other methods in ISO 26262-6 may be required.

[b] The objectives of method 1b are

- Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.
- Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
- Exclusion of language constructs which could result in unhandled run-time errors.

[c] The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

Spec. of sw safety requirements

- Goal:
  - Specify sw safety requirements
  - Detail HSI
  - Verify that sw safety requirements and HSI are consistent with system design and technical safety concept

Spec. of sw safety requirements

- Address each sw-based function whose failure could lead to a violation of a safety requirement allocated to software
  - Functions that enable the system to achieve or maintain a safe state
  - Functions related to the detection, indication and handling of faults of safety-related hw elements
  - Functions related to the detection, notification and mitigation of faults in the software itself
    - At OS level
    - At application level
  - Functions related to on-board and off-board (production, service) tests

Software architectural design

- Goal:
  - Develop a software architectural design that realizes the software safety requirements
  - Verify the sw architecture design

Software architectural design

- Represents all software components and their interactions in a hierarchical structure
  - Static aspects, such as interfaces and data paths between all software components
  - Dynamic aspects, such as process sequences and timing behaviour

- Handles both safety-related and non-safety related requirements

Software architectural design

Table 2 — Notations for software architectural design

Methods		ASIL			
		A	B	C	D
1a	Informal notations	++	++	+	+
1b	Semi-formal notations	+	++	++	++
1c	Formal notations	+	+	+	+

Software architectural design

- Design shall consider:
  - The verifiability of the software architectural design
    - Bi-directional traceability between the software architectural design and the software safety requirements
  - The suitability for configurable software
  - The feasibility for the design and implementation of the software units
  - The testability of the software architecture during software integration testing
  - The maintainability of the software architectural design

Software architectural design

Table 3 — Principles for software architectural design

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

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

[b] Methods 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] Method 1e addresses the limitation of the external coupling of software components.

[d] Any interrupts used have to be priority-based.

Software architectural design

- The software safety requirements shall be allocated to the software components
- Each software component shall be developed in compliance with the highest ASIL of any of the requirements allocated to it

Error recovery

Table 4 — Mechanisms for error detection at the software architectural level

Methods		ASIL			
		A	B	C	D
1a	Range checks of input and output data	++	++	++	++
1b	Plausibility check[a]	+	+	+	++
1c	Detection of data errors[b]	+	+	+	+
1d	External monitoring facility[c]	o	+	+	++
1e	Control flow monitoring	o	+	++	++
1f	Diverse software design	o	o	+	++

[a] Plausibility checks can include using a reference model of the desired behaviour, assertion checks, or comparing signals from different sources.

[b] Types of methods that may be used to detect data errors include error detecting codes and multiple data storage.

[c] An external monitoring facility can be for example an ASIC or another software element performing a watchdog function.

Error handling

Table 5 — Mechanisms for error handling at the software architectural level

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

[a] Static recovery mechanisms can include the use of recovery blocks, backward recovery, forward recovery and recovery through repetition.

[b] Graceful degradation at the software level refers to prioritizing functions to minimize the adverse effects of potential failures on functional safety.

[c] Independent parallel redundancy can be realized as dissimilar software in each parallel path

Verify the sw architecture design

- The following elements shall be verified:
  - Compliance with the software safety requirements
  - Compatibility with the target hardware
  - Adherence to design guidelines

Verify the sw architecture design

Table 6 — Methods for the verification of the software architectural design

Methods		ASIL			
		A	B	C	D
1a	Walk-through of the design[a]	++	+	o	o
1b	Inspection of the design[a]	+	++	++	++
1c	Simulation of dynamic parts of the design[b]	+	+	+	++
1d	Prototype generation	o	o	+	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis[c]	+	+	++	++
1g	Data flow analysis[c]	+	+	++	++

[a] In the case of model-based development these methods can be applied to the model.

[b] Method 1c requires the usage of executable models for the dynamic parts of the software architecture.

[c] Control and data flow analysis may be limited to safety-related components and their interfaces.

Sw unit design and implementation

- Goal:
  - Design and implement sw units
    - Source code, object code
  - Verify sw unit design and implementation

Sw unit design and implementation

Table 7 — Notations for software unit design

Methods		ASIL			
		A	B	C	D
1a	Natural language	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations	+	++	++	++
1d	Formal notations	+	+	+	+

Sw unit design and implementation

Table 8 — Design principles for software unit design and implementation

Methods		ASIL			
		A	B	C	D
1a	One entry and one exit point in subprograms and functions[a]	++	++	++	++
1b	No dynamic objects or variables, or else online test during their creation[a, b]	+	++	++	++
1c	Initialization of variables	++	++	++	++
1d	No multiple use of variable names[a]	+	++	++	++
1e	Avoid global variables or else justify their usage[a]	+	+	++	++
1f	Limited use of pointers[a]	o	+	+	++
1g	No implicit type conversions[a, b]	+	++	++	++
1h	No hidden data flow or control flow[c]	+	++	++	++
1i	No unconditional jumps[a, b, c]	++	++	++	++
1j	No recursions	+	+	++	++

[a] Methods 1a, 1b, 1d, 1e, 1f, 1g and 1i may not be applicable for graphical modelling notations used in model-based development.

[b] Methods 1g and 1i are not applicable in assembler programming.

[c] Methods 1h and 1i reduce the potential for modelling data flow and control flow through jumps or global variables.

Verify sw unit design and impl.

- Shall verify:
  - The compliance with the hardware-software interface specification
  - The fulfillment of the software safety requirements as allocated to the software units through traceability
  - The compliance of the source code with its design specification
  - The compliance of the source code with the coding guidelines
  - The compatibility of the software unit implementations with the target hardware

Verify sw unit design and impl.

Table 9 — Methods for the verification of software unit design and implementation

	Methods	ASIL			
		A	B	C	D
1a	Walk-through[a]	++	+	o	o
1b	Inspection[a]	+	++	++	++
1c	Semi-formal verification	+	+	++	++
1d	Formal verification	o	o	+	+
1e	Control flow analysis[b, c]	+	+	++	++
1f	Data flow analysis[b, c]	+	+	++	++
1g	Static code analysis	+	++	++	++
1h	Semantic code analysis[d]	+	+	+	+

[a] In the case of model-based software development the software unit specification design and implementation can be verified at the model level.

[b] Methods 1e and 1f can be applied at the source code level. These methods are applicable both to manual code development and to model-based development.

[c] Methods 1e and 1f can be part of methods 1d, 1g or 1h.

[d] Method 1h is used for mathematical analysis of source code by use of an abstract representation of possible values for the variables. For this it is not necessary to translate and execute the source code.

Sw unit testing

- Goal:
  - Demonstrate that the software units fulfill the software unit design specifications and do not contain undesired functionality
  - Establish test procedure
  - Perform the test

Sw unit testing

- It shall demonstrate:
  - Compliance with the software unit design specification
  - Compliance with the specification of the hardware-software interface
  - The specified functionality
  - Confidence in the absence of unintended functionality
  - Robustness
  - Sufficient resources to support their functionality

Testing methods

Table 10 — Methods for software unit testing

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[b]	+	+	+	++
1d	Resource usage test[c]	+	+	+	++
1e	Back-to-back comparison test between model and code, if applicable[d]	+	+	++	++

[a] The software requirements at the unit level are the basis for this requirements-based test.

[b] This includes injection of arbitrary faults (e.g. by corrupting values of variables, by introducing code mutations, or by corrupting values of CPU registers).

[c] Some aspects of the resource usage test can only be evaluated properly when the software unit tests are executed on the target hardware or if the emulator for the target processor supports resource usage tests.

[d] This method requires a model that can simulate the functionality of the software units. Here, the model and code are stimulated in the same way and results compared with each other.

Testing methods

Table 12 — Structural coverage metrics at the software unit level

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

Test environment

- The test environment for software unit testing shall correspond as closely as possible to the target environment

- If the software unit testing is not carried out in the target environment, the differences in the source and object code, and the differences between the test environment and the target environment, shall be analysed in order to specify additional tests in the target environment during the subsequent test phases

Sw integration and testing

- Goal:
  - Integrate the software elements
  - Demonstrate that the software architectural design is realized by the embedded software
- The embedded software can consist of safety- related and non-safety-related software elements
- It shall consider
  - The functional dependencies that are relevant for software integration
  - The dependencies between the software integration and the hardware-software integration

Sw integration and testing

- It shall demonstrate:
  - Compliance with the software architectural design
  - Compliance with the specification of the hardware-software interface
  - The specified functionality
  - Robustness
    - Example: absence of inaccessible software; effective error detection and handling
  - Sufficient resources to support the functionality

Sw integration and testing

Table 13 — Methods for software integration testing

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[b]	+	+	++	++
1d	Resource usage test[c, d]	+	+	+	++
1e	Back-to-back comparison test between model and code, if applicable[e]	+	+	++	++

[a] The software requirements at the architectural level are the basis for this requirements-based test.

[b] This includes injection of arbitrary faults in order to test safety mechanisms (e.g. by corrupting software or hardware components).

[c] To ensure the fulfilment of requirements influenced by the hardware architectural design with sufficient tolerance, properties such as average and maximum processor performance, minimum or maximum execution times, storage usage (e.g. RAM for stack and heap, ROM for program and data) and the bandwidth of communication links (e.g. data buses) have to be determined.

[d] Some aspects of the resource usage test can only be evaluated properly when the software integration tests are executed on the target hardware or if the emulator for the target processor supports resource usage tests.

[e] This method requires a model that can simulate the functionality of the software components. Here, the model and code are stimulated in the same way and results compared with each other.

Sw integration and testing

Table 14 — Methods for deriving test cases for software integration testing

Methods		ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Generation and analysis of equivalence classes[a]	+	++	++	++
1c	Analysis of boundary values[b]	+	++	++	++
1d	Error guessing[c]	+	+	+	+

[a] Equivalence classes can be identified based on the division of inputs and outputs, such that a representative test value can be selected for each class.

[b] This method applies to parameters or variables, values approaching and crossing the boundaries and out of range values.

[c] Error guessing tests can be based on data collected through a "lessons learned" process and expert judgment.

Sw integration and testing

Table 15 — Structural coverage metrics at the software architectural level

Methods		ASIL			
		A	B	C	D
1a	Function coverage[a]	+	+	++	++
1b	Call coverage[b]	+	+	++	++

[a] Method 1a refers to the percentage of executed software functions. This evidence can be achieved by an appropriate software integration strategy.

[b] Method 1b refers to the percentage of executed software function calls.

Test environment

- The test environment for software integration testing shall correspond as closely as possible to the target environment

- If the software integration testing is not carried out in the target environment, the differences in the source and object code and the differences between the test environment and the target environment shall be analysed in order to specify additional tests in the target environment during the subsequent test phases.

Verification of sw safety req.

- Goal:
  - Demonstrate that the embedded software satisfies its requirements in the target environment
- Criteria:
  - Compliance with the expected results
  - Coverage of the software safety requirements
  - Pass or fail criteria

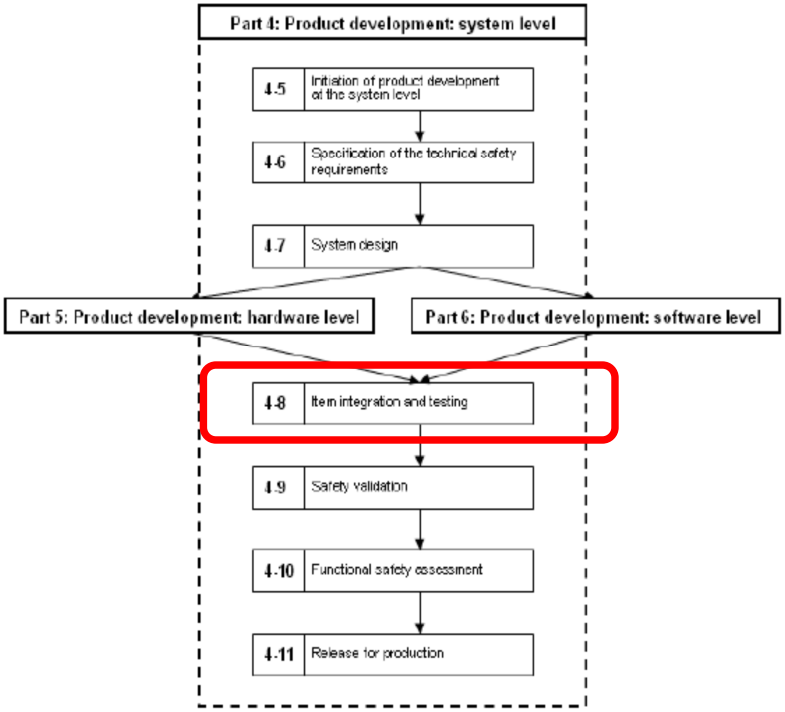
Verification of sw safety req.

Table 16 — Test environments for conducting the software safety requirements verification

Methods		ASIL			
		A	B	C	D
1a	Hardware-in-the-loop	+	+	++	++
1b	Electronic control unit network environments[a]	++	++	++	++
1c	Vehicles	++	++	++	++

[a] Examples include test benches partially or fully integrating the electrical systems of a vehicle, "lab-cars" or "mule" vehicles, and "rest of the bus" simulations.

Integration and testing

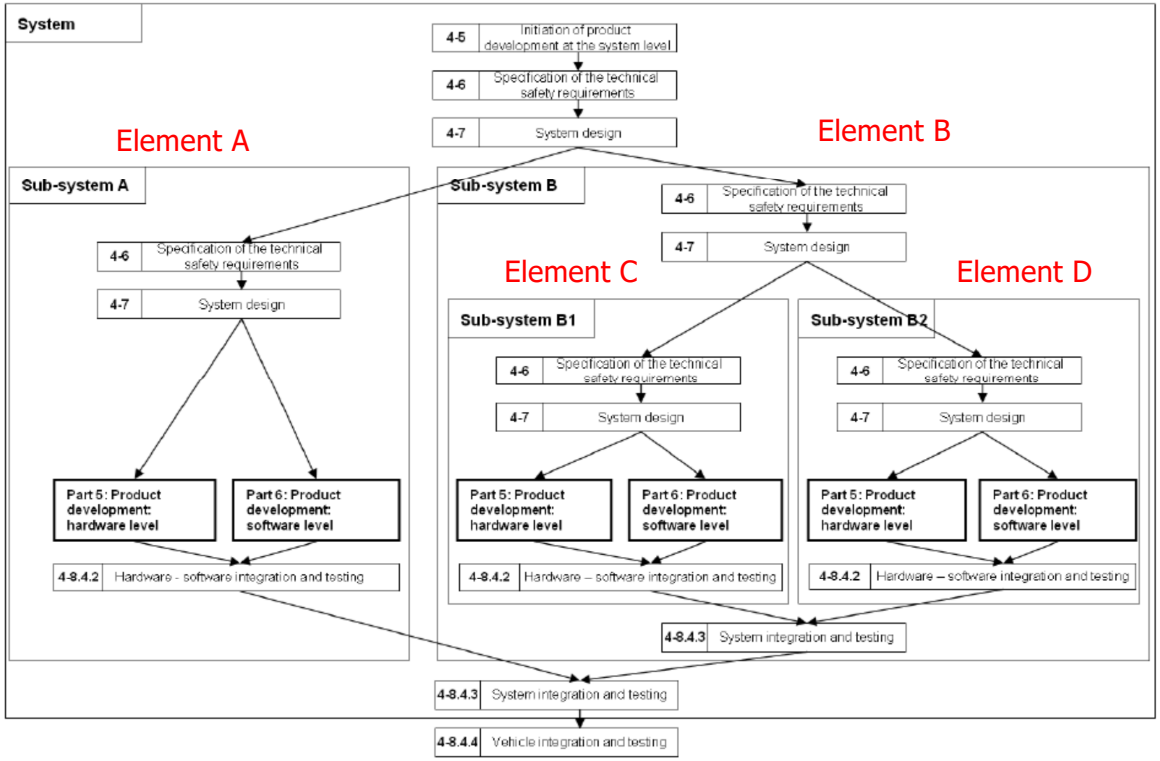


Integration and testing

- Goal:
  - Test compliance with each safety requirement
  - Verify that item implements safety requirements

- Three phases:
  - Integration of the hardware and software of each element in the item
  - Integration of the elements to form item
  - Integration of the item with other systems within a vehicle and with the vehicle itself

Integration



Integration and testing

- Define integration and test strategy for:
  - HW/SW level
  - System level
  - Vehicle level

HW/SW level

Table 5 — The correct implementation of technical safety requirements at the hardware-software level

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test[a]	++	++	++	++
1b	Fault injection test[b]	+	++	++	++
1c	Back-to-back test[c]	+	+	++	++

[a] A requirements-based test denotes a test against functional and non-functional requirements.

[b] A fault injection test uses special means to introduce faults into the test object during runtime. This can be done within the software via a special test interface or specially prepared hardware. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety mechanisms are not invoked.

[c] A back-to-back test compares the responses of the test object with the responses of a simulation model to the same stimuli, to detect differences between the behaviour of the model and its implementation.

Does functional safety of the element work when fault happens?

HW/SW level

Table 6 — The correct functional performance, accuracy and timing of safety mechanisms at the hardware-software level

Methods		ASIL			
		A	B	C	D
1a	Back-to-back test[a]	+	+	++	++
1b	Performance test[b]	+	++	++	++

[a] A back-to-back test compares the responses of the test object with the responses of a simulation model to the same stimuli, to detect differences between the behaviour of the model and its implementation.

[b] A performance test can verify the performance (e.g. task scheduling, timing, power output) in the context of the whole test object, and can verify the ability of the intended control software to run with the hardware.

Is element performance what it should be?

HW/SW level

Table 7 — The consistent and correct implementation of external and internal interfaces at the hardware-software level

	Methods	ASIL			
		A	B	C	D
1a	Test of external interfaces[a]	+	++	++	++
1b	Test of internal interfaces[a]	+	++	++	++
1c	Interface consistency check[a]	+	++	++	++

[a] Interface tests of the test object include tests of analogue and digital inputs and outputs, boundary tests and equivalence-class tests to completely test the specified interfaces, compatibility, timings and other specified ratings for the test object. Internal interfaces of an ECU can be tested by static tests for the compatibility of software and hardware as well as dynamic tests of Serial Peripheral Interface- (SPI) or Integrated Circuit- (I^2C) communications or any other interface between elements of an ECU.

Are element interfaces what they should be?

HW/SW level

Table 8 — The effectiveness of a safety mechanism's diagnostic coverage at the hardware-software level

	Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a	Fault injection test[a]	+	+	++	++
1b	Error guessing test[b]	+	+	++	++

[a] A fault injection test uses special means to introduce faults into the test object during runtime. This can be done within the software via a special test interface or specially prepared hardware. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety mechanisms are not invoked.

[b] An error guessing test uses expert knowledge and data collected through lessons learned to anticipate errors in the test object. Then a set of tests along with adequate test facilities is designed to check for these errors. Error guessing is an effective method given a tester who has previous experience with similar test objects.

Does element functional safety work for all the possible faults?

HW/SW level

Table 9 — The level of robustness at the hardware-software level

Methods		ASIL			
		A	B	C	D
1a	Resource usage test[a]	+	+	+	++
1b	Stress test[b]	+	+	+	++

[a] A resources usage test can be done statically, (e.g. by checking for code sizes or analyzing the code regarding interrupt usage, in order to verify that worst-case scenarios do not run out of resources), or dynamically by runtime monitoring.

[b] A stress test verifies the test object for correct operation under high operational loads or high demands from the environment. Therefore, tests under high loads on the test object, or with exceptional interface loads, or values (bus loads, electrical shocks etc.), as well as tests with extreme temperatures, humidity or mechanical shocks, can be applied.

Is the element robust under any workloads?

System level

Table 10 — The correct implementation of functional safety and technical safety requirements at the system level

Methods		ASIL			
		A	B	C	D
1a	Requirement-based test[a]	++	++	++	++
1b	Fault injection test[b]	+	+	++	++
1c	Back-to-back test[c]	o	+	+	++

[a] A requirements-based test denotes a test against functional and non-functional requirements.

[b] A fault injection test uses special means to introduce faults into the system. This can be done within the system via a special test interface or specially prepared elements or communication devices. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety mechanisms are not invoked.

[c] A back-to-back test compares the responses of the test object with the responses of a simulation model to the same stimuli, to detect differences between the behaviour of the model and its implementation.

System level

Table 11— The correct functional performance, accuracy and timing of safety mechanisms at the system level

Methods		ASIL			
		A	B	C	D
1a	Back-to-back test[a]	o	+	+	++
1b	Performance test[b]	o	+	+	++

[a] A back-to-back test compares the responses of the test object with the responses of a simulation model to the same stimuli, to detect differences between the behaviour of the model and its implementation.

[b] A performance test can verify the performance (e.g. actuator speed or strength, whole system response times) of the safety mechanisms concerning the system.

System level

Table 12 — The consistent and correct implementation of external and internal interfaces at the system level

	Methods	ASIL			
		A	B	C	D
1a	Test of external interfaces[a]	+	++	++	++
1b	Test of internal interfaces[a]	+	++	++	++
1c	Interface consistency check[a]	o	+	++	++
1d	Test of interaction/communication[b]	++	++	++	++

[a] An interface test of the system includes tests of analogue and digital inputs and outputs, boundary tests, and equivalence-class tests, to completely test the specified interfaces, compatibility, timings, and other specified characteristics of the system. Internal interfaces of the system can be tested by static tests, (e.g. match of plug connectors) as well as by dynamic tests concerning bus communications or any other interface between system elements.

[b] A communication and interaction test includes tests of the communication between the system elements, as well as between the system under test and other vehicle systems during runtime, against the functional and non-functional requirements.

System level

Table 13 — The effectiveness of a safety mechanism's failure coverage at the system level

Test methods		ASIL			
		A	B	C	D
1a	Fault injection test[a]	+	+	++	++
1b	Error guessing test[b]	+	+	++	++
1c	Test derived from field experience[c]	o	+	++	++

[a] A fault injection test uses special means to introduce faults into the system. This can be done within the system via a special test interface, specially prepared elements, or communication devices. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety measures are not invoked.

[b] An error guessing test uses expert knowledge and data collected through lessons learned and field experience to anticipate errors in the system. Then a set of tests along with adequate test facilities is designed to check for these errors. Error guessing is an effective method given a tester who has previous experience with similar systems.

System level

Table 14 — The level of robustness at the system level

Methods		ASIL			
		A	B	C	D
1a	Resource usage test[a]	o	+	++	++
1b	Stress test[b]	o	+	++	++
1c	Test for interference resistance and robustness under certain environmental conditions[c]	++	++	++	++

[a] At the sytem level resource usage testing is usually performed in dynamic environments (e.g. lab cars or prototypes). Issues to test include power consumption and bus load.

[b] A stress test verifies the correct operation of the system under high operational loads or high demands from the environment. Therefore, tests under high loads on the system, or with extreme user inputs or requests from other systems, as well as tests with extreme temperatures, humidity or mechanical shocks, can be applied.

[c] A test for interference resistance and robustness, under certain environmental conditions, is a special case of stress testing. This includes EMC and ESD tests (e.g. see [2], [3]).

Vehicle level

Table 15 — The correct implementation of the functional safety requirements at the vehicle level

Methods		ASIL			
		A	B	C	D
1a	Requirement-based test[a]	++	++	++	++
1b	Fault injection test[b]	++	++	++	++
1c	Long term test[c]	++	++	++	++
1d	User test under real-life conditions[c]	++	++	++	++

[a] A requirements-based test denotes a test against functional and non-functional requirements.

[b] A fault injection test uses special means to introduce faults into the item. This can be done within the item via a special test interface or specially prepared elements or communication devices. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety mechanisms are not invoked

[c] A long term test and a user test under real-life conditions are similar to tests derived from field experience but use a larger sample size, normal users as testers, and are not bound to prior specified test scenarios, but performed under real-life conditions during everyday life. These test can have limitations if necessary to ensure the safety of the testers, e.g. with additional safety measures or disabled actuators.

Vehicle level

Table 16 — The correct functional performance, accuracy and timing of safety mechanisms at the vehicle level

Methods		ASIL			
		A	B	C	D
1a	Performance test[a]	+	+	++	++
1b	Long term test[b]	+	+	++	++
1c	User test under real-life conditions[b]	+	+	++	++

[a] A performance test can verify the performance (e.g. fault tolerant time intervals and vehicle controllability in the presence of faults) of the safety mechanisms concerning the item.

[b] A long term test and a user test under real-life conditions are similar to tests derived from field experience but use a larger sample size, normal users as testers, and are not bound to prior specified test scenarios, but performed under real-life conditions during everyday life. These test can have limitations if necessary to ensure the safety of the testers, e.g. with additional safety measures or disabled actuators.

Vehicle level

Table 17 — The consistent and correct implementation of internal and external interfaces at the vehicle level

Methods		ASIL			
		A	B	C	D
1a	Test of external interfaces[a]	o	+	++	++
1b	Test of interaction/communication[b]	o	+	++	++

[a] An interface test at the vehicle level tests the interfaces of the vehicle systems for compatibility. This can be done statically by validating value ranges, ratings or geometries as well as dynamically during operation of the whole vehicle.

[b] A communication and interaction test includes tests of the communication between the systems of the vehicle during runtime against functional and non-functional requirements.

Vehicle level

Table 18 — The effectiveness a safety mechanism's failure coverage at the vehicle level

Methods		ASIL			
		A	B	C	D
1a	Fault injection test[a]	o	+	++	++
1b	Error guessing test[b]	o	+	++	++
1c	Test derived from field experience[c]	o	+	++	++

[a] A fault injection test uses special means to introduce faults into the vehicle. This can be done within the vehicle via a special test interface, specially prepared hardware or communication devices. The method is often used to improve the test coverage of the safety requirements, because during normal operation safety measures are not invoked.

[b] An error guessing test uses expert knowledge and data collected through lessons learned to anticipate errors in the vehicle. Then a set of tests along with adequate test facilities is designed to check for these errors. Error Guessing is an effective method given a tester who has previous experience with similar vehicle applications.

[c] A test derived from field experience uses the experience and data gathered from the field. Erroneous vehicle behaviour or newly discovered operational situations are analysed and a set of tests is designed to check the vehicle with respect to the new findings.

Vehicle level

Table 19 — The level of robustness at the vehicle level

Methods		ASIL			
		A	B	C	D
1a	Resource usage test[a]	o	+	++	++
1b	Stress test[b]	o	+	++	++
1c	Test for interference resistance and robustness under certain environmental conditions[c]	o	+	++	++
1d	Long term test[d]	o	+	++	++

[a] At the item level resource usage testing is usually performed in dynamic environments (e.g. lab cars or prototypes). Issues to test include item internal resources, power consumption or limited resources of other vehicle systems.

[b] A stress test verifies the correct operation of the vehicle under high operational loads or high demands from the environment. Therefore tests under high loads on the vehicle or with extreme user inputs or requests from other systems as well as tests with extreme temperatures, humidity or mechanical shocks can be applied.

[c] A test for interference resistance and robustness, under certain environmental conditions, is a special case of stress testing. This includes EMC and ESD tests (e.g. see [2], [3]).

[d] A long term test and a user test under real-life conditions are similar to tests derived from field experience but use a larger sample size, normal users as testers, and are not bound to prior specified test scenarios, but performed under real-life conditions during everyday life.