RENAULT-NISSAN-MITSUBISHI RESTRICTED B

RENAULT NISSAN MITSUBISHI

Safety SW development

Agenda

Introduction to ISO

From FSC to SW Specification

SW Specification

SW Architectural Design

Software safety analyses

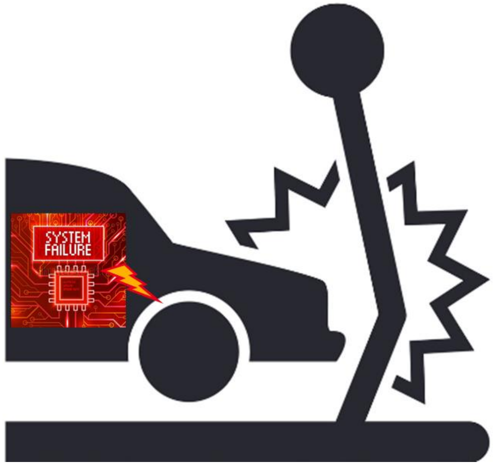
Conceptual approach and practical examples

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety purpose

- ISO 26262 is a **functional safety** standard
- Main purpose of **Safety**: Prevent or mitigate hazard

- **Hazard:** potential source of harm caused by malfunctioning behaviour of the item



Safety scope

Fault type and fault origin

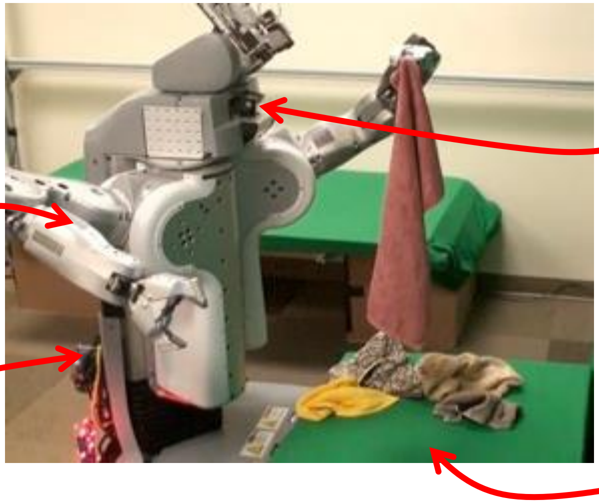
Internal hazard

HW component fault

Random faults (λ)

Development fault

Systematic faults

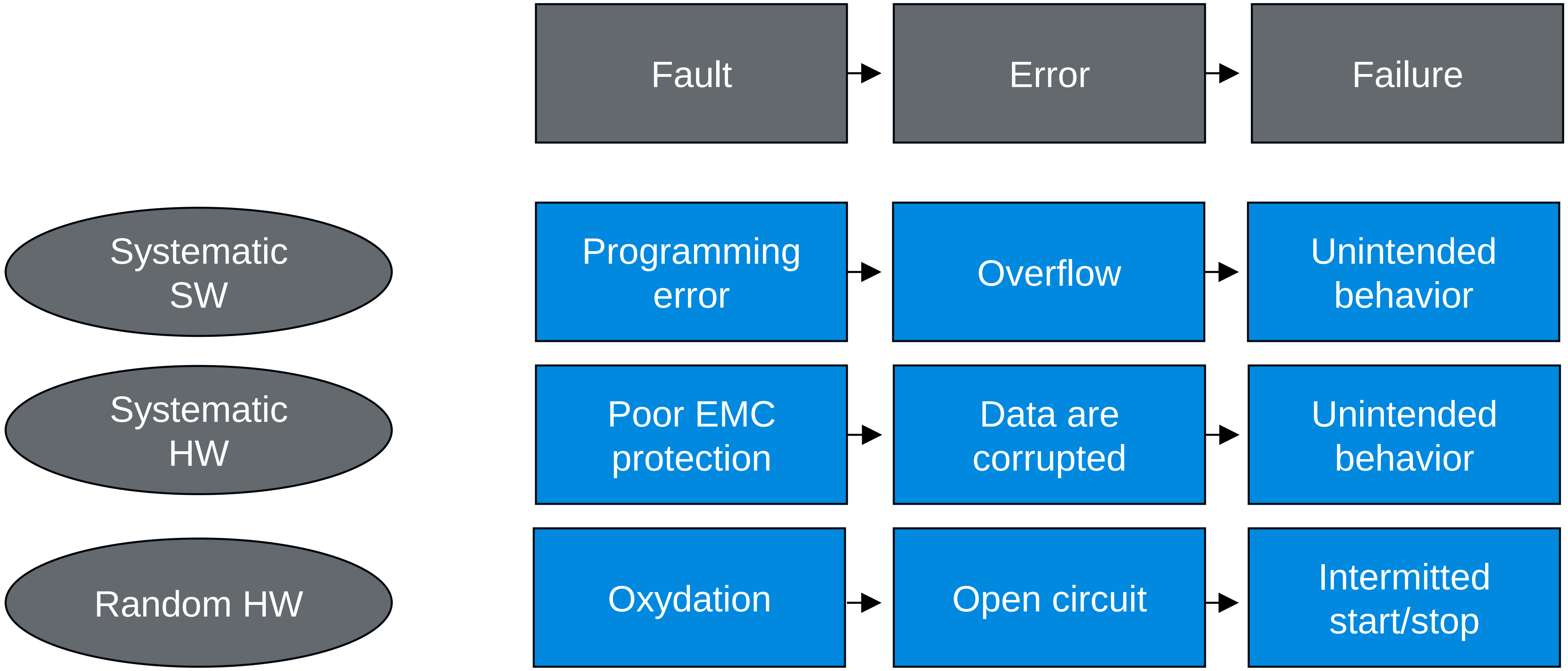


External hazard

Wrong interpretation of outside world

Environmental conditions

From fault to failure



	Fault	→	Error	→	Failure
Systematic SW	Programming error	→	Overflow	→	Unintended behavior
Systematic HW	Poor EMC protection	→	Data are corrupted	→	Unintended behavior
Random HW	Oxydation	→	Open circuit	→	Intermitted start/stop

Example of Hazard

https://www.youtube.com/watch?v=kYUrqdUyEpI

On June 4, 1996 Ariane 5 rocket launched by the European Space Agency **exploded** just forty seconds after its lift-off. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million.

The guidance system shutdown occurred 36.7 seconds after launch, when the guidance system's own computer tried to convert one piece of data -- the sideways velocity of the rocket -- from a 64-bit format to a 16-bit format. The number was too big, and an **overflow error** resulted. When the guidance system shut down, it passed control to an identical, **redundant unit**, which was there to provide backup in case of just such a failure. But the second unit had **failed in the identical manner** a few milliseconds before. It was running the same software…

Root cause:

In this case, the programmers had decided that this particular velocity figure would never be large enough to cause trouble. Moreover the guidance system was **re-used** form Ariane 4, and worked perfectly 23 times.
Unluckily, Ariane 5 was a faster rocket than Ariane 4…

Source: https://around.com/ariane.html

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

ISO 26262:2011/2018 Overview

- **ISO 26262:2018**

functional safety standard for road vehicles (cars, trucks, buses, trailers and semi-trailers).

- **12 parts in the ISO 26262:2018**

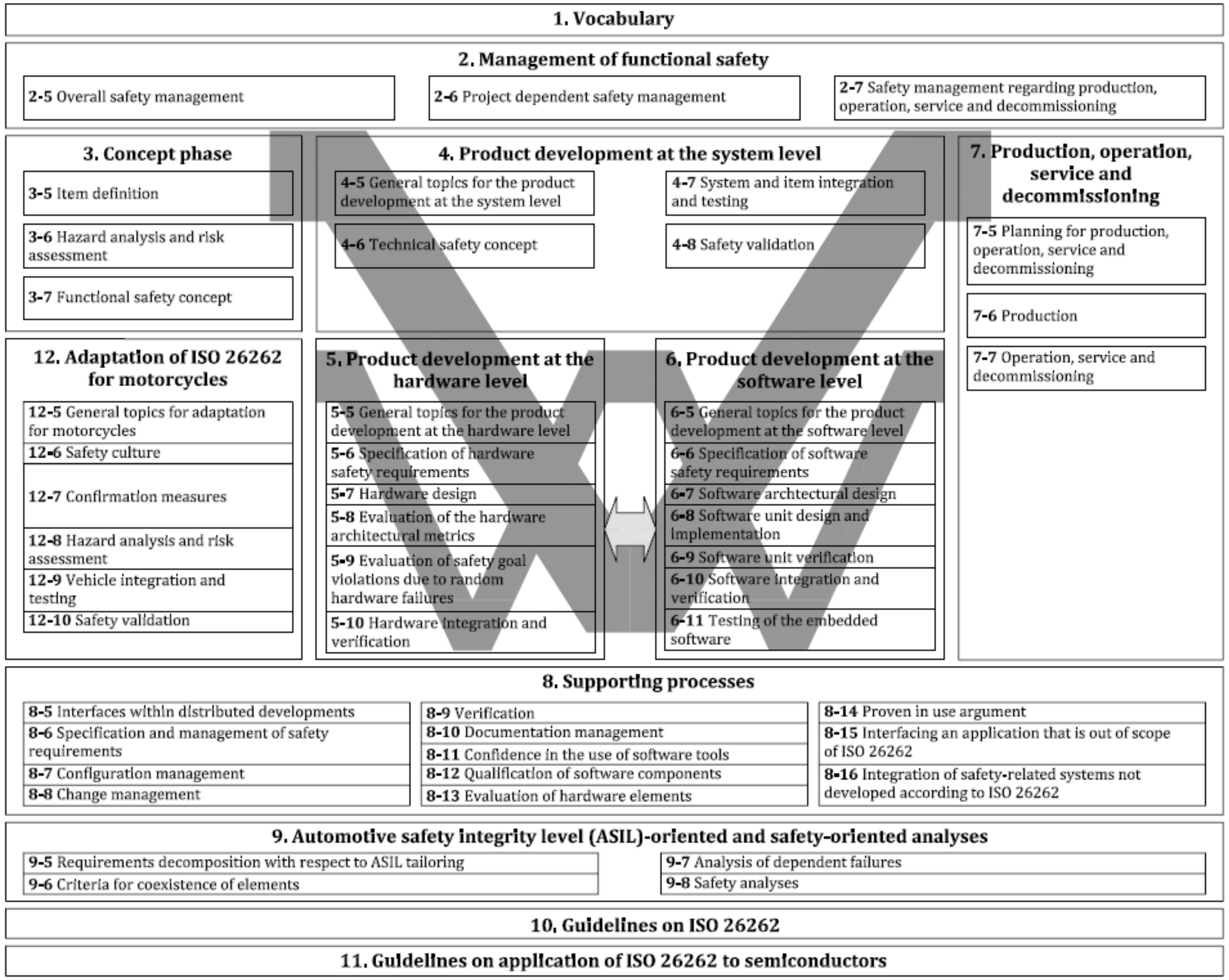
  - **10 are normative**
    - Management/Support
    - Conception
    - Development
    - Production
    - Adaptation of ISO 26262 for motorcycles

  - **2 are informative**
    - Guidelines :
      - on ISO 26262
      - on application of ISO 26262 to semiconductors

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

ISO 26262:2018 Overview

- ISO 26262:2018



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. Adaptation of ISO 26262 for motorcycles

12-5 General topics for adaptation for motorcycles

12-6 Safety culture

12-7 Confirmation measures

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 product 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 violations 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 that is out of scope of ISO 26262

8-16 Integration of safety-related systems not developed according to ISO 26262

9. Automotive safety integrity level (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 analyses

10. Guidelines on ISO 26262

11. Guidelines on application of ISO 26262 to semiconductors

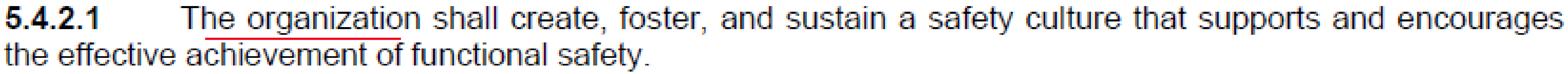
Safety competence

In addition to requirement, methods, work product definition (previous slide)

ISO 26262 shall be carried out by everyone

ISO26262 clearly specifies 2 chapters:

ISO26262-2 5.4.2 Safety culture



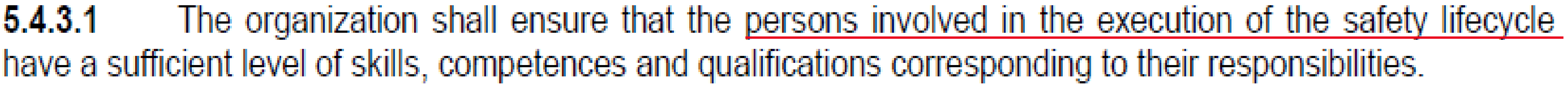
5.4.2.1 The organization shall create, foster, and sustain a safety culture that supports and encourages the effective achievement of functional safety.

✓ Safety and quality issues are discovered and resolved from the earliest stage in the product lifecycle

✓ Safety is the highest priority

✓ Process/Guideline/Template compliant with ISO26262

ISO26262-2 5.4.3 Competence management



5.4.3.1 The organization shall ensure that the persons involved in the execution of the safety lifecycle have a sufficient level of skills, competences and qualifications corresponding to their responsibilities.

✓ Specific role training (for Architect..)

✓ ISO 26262 training

✓ ISO 26262 support (Safety Manager or Safety expert)

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Vocabulary (1/3)

Hazardous event, or Unwanted Customer Event (UCE) safety related

- Combination of a *hazard* and an *operational situation*

  Hazard: Potential source of *harm* caused by *malfunctioning behaviour* of the *item*

  Operational situation: Scenario that can occur during a vehicle's life
- Example: "Unexpected take off, requested by longitudinal TJP feature, while stopped in a traffic jam."

Safety goal

- Top-level *safety* requirement as a result of the *hazard analysis and risk assessment* at the vehicle level
- Example: "No unexpected take off while stopping, requested by longitudinal TJP feature."
  TJP = Traffic Jam Pilot

ASIL (Automotive Safety Integrity Level) A,B,C,D

Functional Safety Concept

- Specification of the functional safety requirements, with associated information, their allocation to elements within the architecture, and their interaction necessary to achieve the safety Goals

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Vocabulary (2/3)

Functional Safety Requirement

- Specification of implementation-independent safety behaviour or implementation-independent safety measure including its safety-related attributes

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 Requirement

- Requirement derived for implementation of associated functional safety requirements

FTTI (Fault time tolerance interval)

Safe state

How the ISO 26262 is structured?

ISO 26262 Overview

- **Every chapter of the standard is presented in the following way:**

X Chapter Title

X.1 Objectives

X.2 General

X.3 Input of this clause

X.3.1 Prerequisites

(Documentation/necessary study before to begin the sub-phase)

X.3.2 Further supporting information

(Documentation/study to be considered before to begin the sub-phase)

X.4 Requirement and recommendation

(List of requirements to be complied with during the sub-phase)

X.5 Work products

(Documentation/study to achieve during the sub-phase)

How to read tables ?

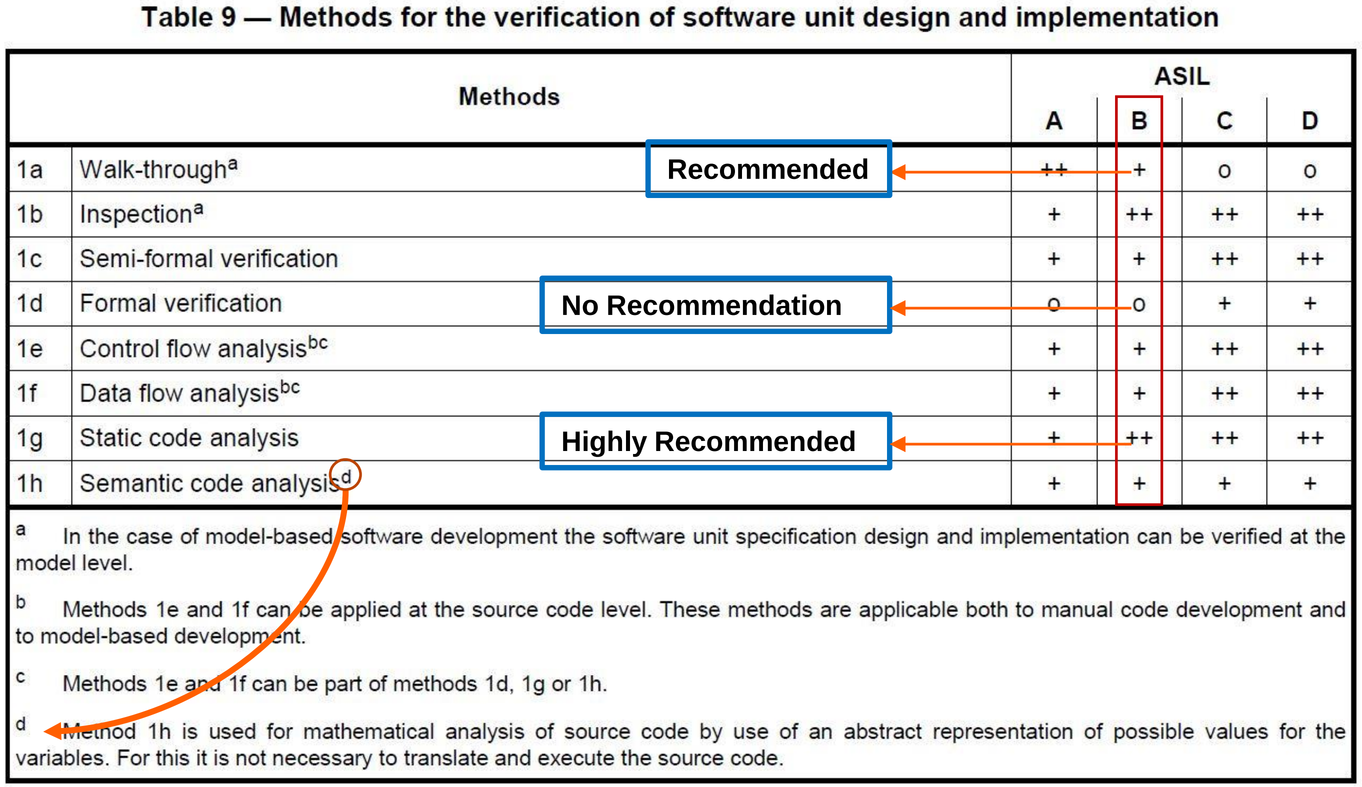


Table 9 — Methods for the verification of software unit design and implementation

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[bc]	+	+	++	++
1f	Data flow analysis[bc]	+	+	++	++
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.

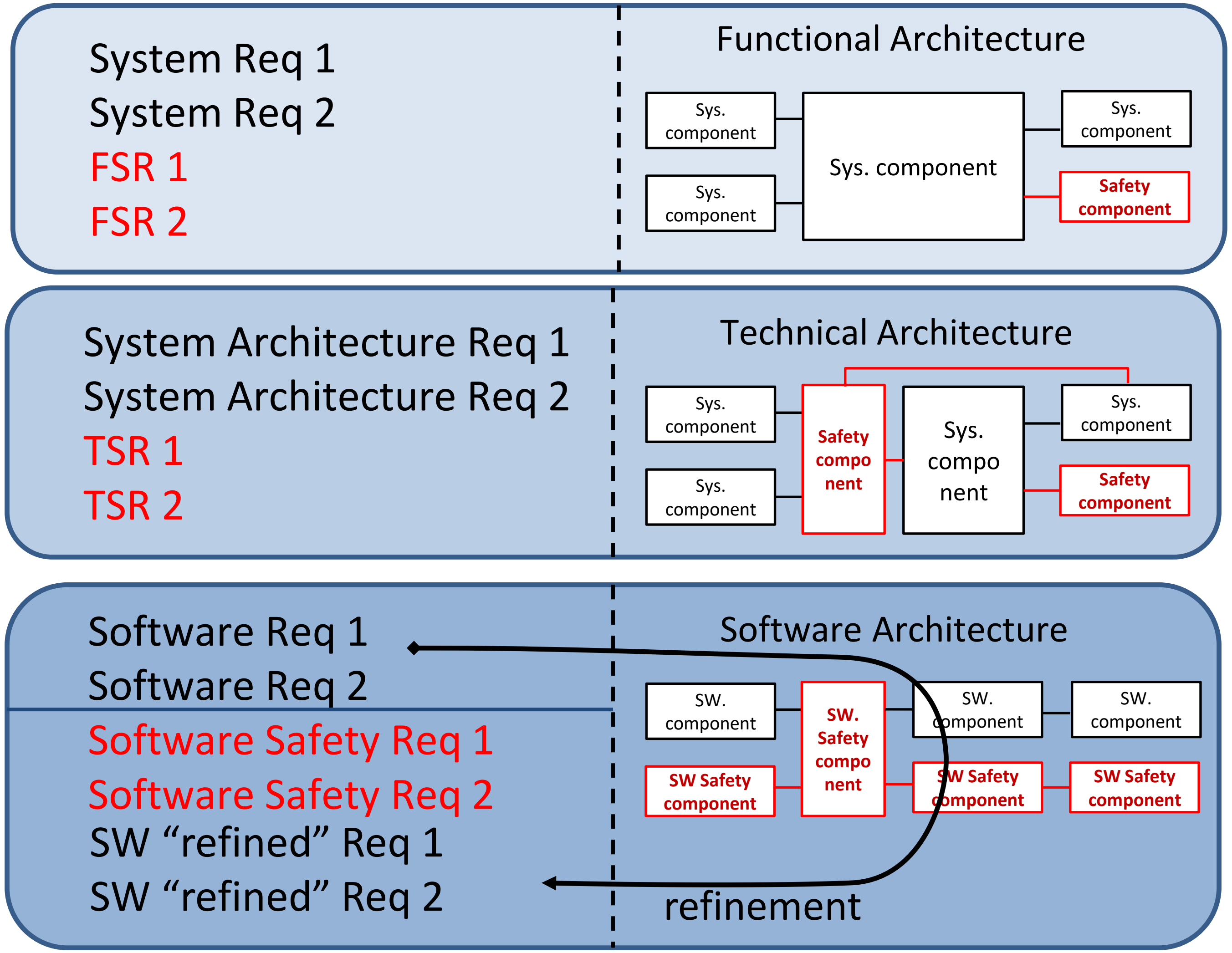
Recommended

No Recommendation

Highly Recommended

From Functional Safety requirements To SW Safety requirements

Requirement and architecture cycle



Only testable at vehicle level

Not testable. Functional architecture has no meaning

System Tests

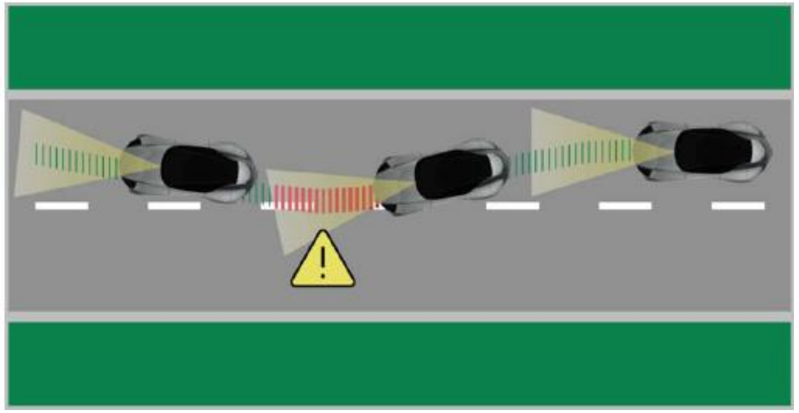
System Integration Tests

Software Tests

Software Integration Tests

ISO26262 training example: LKA (1/2)

- Disclaimer: The « Lane Keep Assist » example below does not claim to be complete in any sense.
- The main purpose is to illustrate the derivation of requirements



- The basic goal of the LKA system is to serve “as a mechanism designed to warn a driver when the vehicle begins to move out of its lane (unless a turn signal is on in that direction or the brake is used) and to perform correcting measures if necessary.”

ISO26262 training example: LKA (2/2)

- Risk: Delivery of counter steering in the wrong direction (ASIL B)

Safety Goals

Hazard Analysis and Risk Assessment

SG1 No unwanted application of counter torque on steering shaft (ASIL B)

Functional safety requirements

Functional Architecture

FR1 The LKA shall be disabled when turn signal is in use

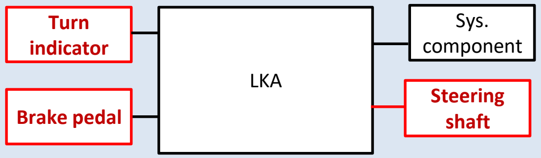
FSR1.1 The turn indicator information shall be secured

FR1.2 If the turn indicator information is unavailable or not reliable, LKA shall be disabled

FR2 The LKA shall be disabled when brake pedal is pressed

FSR2.1 The brake pedal information shall be secured

FR2.2 If the brake pedal information is unavailable or not reliable, LKA shall be disabled



Technical safety requirements

Technical Architecture

TR1.1 Turn signal state shall be acquired from CAN message FlashingIndicatorStatus every 100 ms

TR1.2 LKA shall be disabled when turn indicator value is “left” or “right”

TSR1.3 The turn indicator information shall be secured with Clock and CRC

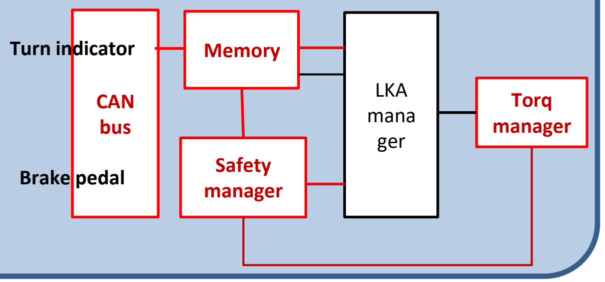
TSR1.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled

TR2.1 Consolidated brake pedal state shall be acquired from CAN every 20 ms

TR2.2 LKA shall be disabled when consolidated brake pedal state is “pedal pressed”

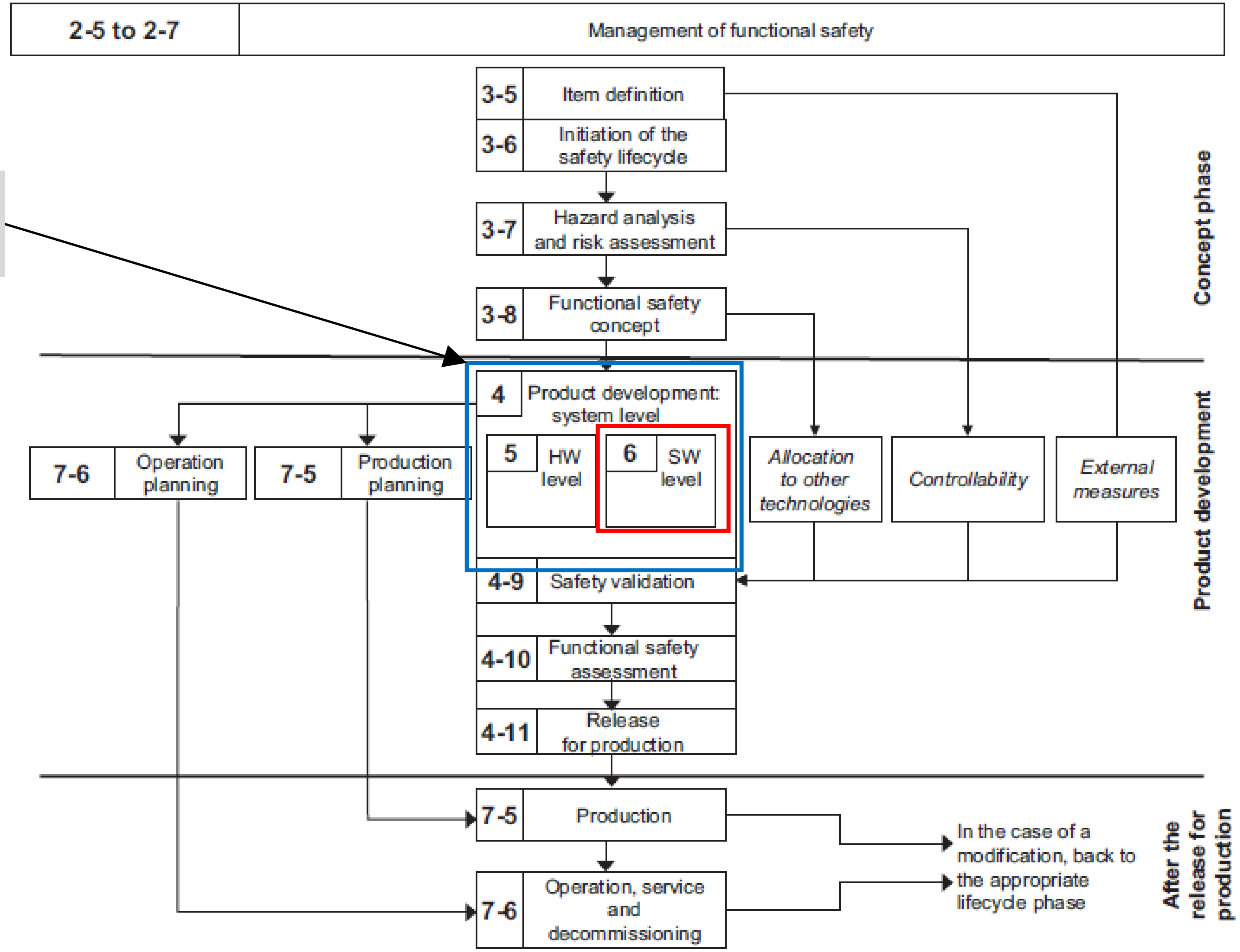
TSR2.3 The brake pedal information shall be secured with Clock and CRC

TSR2.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled



ISO 26262 activity flow

Product development parts


2-5 to 2-7
Management of functional safety
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4 Product development: system level
5 HW level
6 SW level
7-6 Operation planning
7-5 Production planning
Allocation to other technologies
Controllability
External measures
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
7-5 Production
7-6 Operation, service and decommissioning
In the case of a modification, back to the appropriate lifecycle phase
Concept phase
Product development
After the release for production

Software Lifecycle

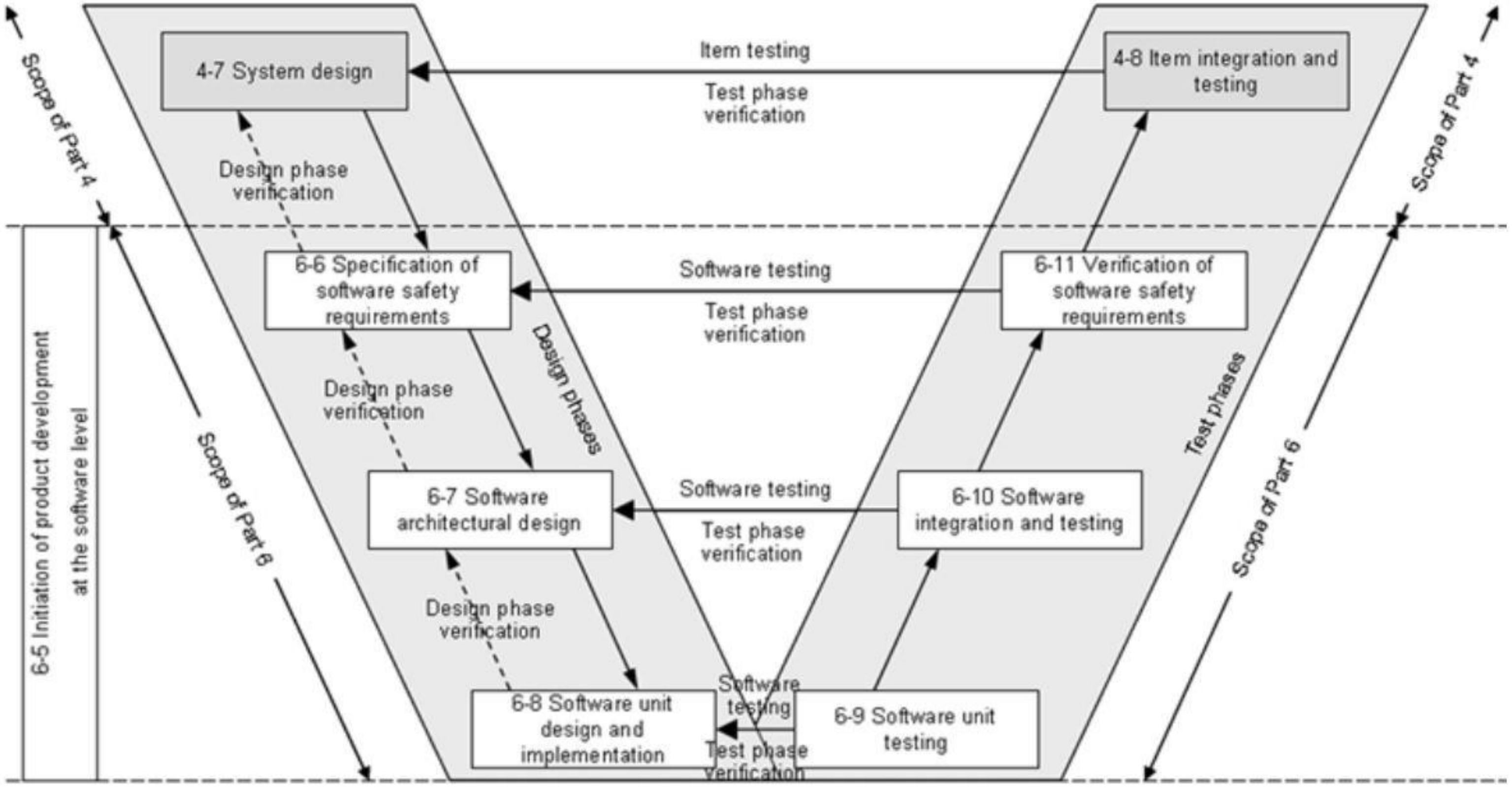

4-7 System design
Item testing
Test phase verification
4-8 Item integration and testing
Scope of Part 4
Scope of Part 4
Design phase verification
6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phases
Test phases
6-5 Initiation of product development at the software level
Scope of Part 6
Scope of Part 6
Design phase verification
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

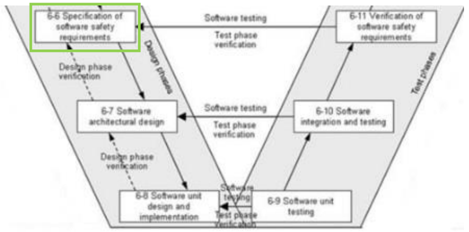
Figure 2 — Reference phase model for the software development

6-6 Specification of software safety requirements

6-6 Specification of software safety requirements

Objectives

- Specify the software safety requirements derived from the:
  - System design specification
  - With System design specification derived from Technical Safety Requirement Specification
- Detail the HW / SW interface and consider the impact of HW constraints on the SW
- Verify the consistency of software safety requirements


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phases
Test phases
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

Work products

- Software Safety Specification
- Hardware-Software Interface (HSI) specification (updated)
- Software verification report

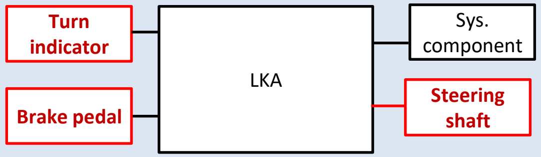
6-6 Specification of software safety requirements

Functional safety requirements

FR1 The LKA shall be disabled when turn signal is in use
FSR1.1 The turn signal information shall be secured
FR1.2 If the turn signal information is unavailable or not reliable, LKA shall be disabled

FR2 The LKA shall be disabled when brake pedal is pressed
FSR2.1 The brake pedal information shall be secured
FR2.2 If the brake pedal information is unavailable or not reliable, LKA shall be disabled

Functional Architecture

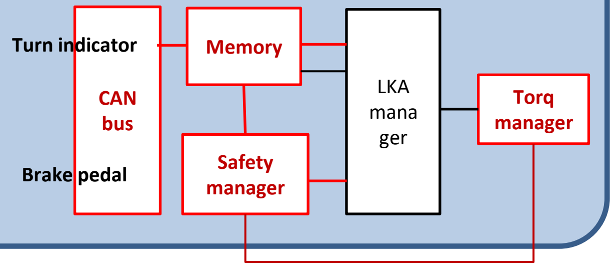


Technical safety requirements

TR1.1 Turn signal state shall be acquired from CAN message FlashingIndicatorStatus every 100 ms
TR1.2 LKA shall be disabled when turn signal value is “left” or “right”
TSR1.3 The turn signal information shall be secured with Clock and CRC
TSR1.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled

TR2.1 Consolidated BrakeInfoStatus shall be acquired from CAN every 20 ms
TR2.2 LKA shall be disabled when consolidated BrakeInfoStatus is “pedal pressed”
TSR2.3 The brake pedal information shall be secured with Clock and CRC
TSR2.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled

Technical Architecture



Hardware Safety Requirements

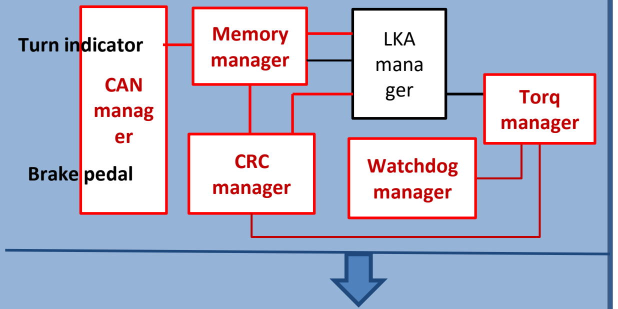
SR1.1.1 The software shall acquire FlashingIndicatorStatus from CAN every 100 ms
SR1.2.1 The software shall set lka_authorization to false when turn signal value is “left” or “right”
…
SR2.1.1 The software shall acquire BrakeInfoStatus from CAN every 20 ms
SR2.2.1 The software shall set lka_authorization to false when brake pedal signal value is “pedal pressed”
SSR2.3.1 The software shall monitor Clock and CRC for BrakeInfoStatus
SSR2.4.1 The software shall set lka_authorization to false if Clock or CRC for BrakeInfoStatus is erroneous
SR2.4.2 The software shall set lka_authorization to false if BrakeInfoStatus is unavailable

SR0.0.1 Store lka_authorization data in RAM from section x to y

SW safety mechanism

ISSR0.0.2 Store lka_authorization with a CRC in protected RAM
SSR0.0.3 Check lka_authorization ‘s CRC before using it. If the check fails, use the default FALSE value.

Software Architecture



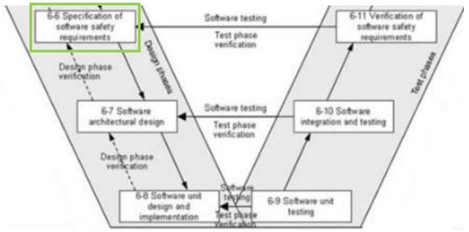
Software Safety analysis (CPA, FMEA)

6-6 Specification of software safety requirements
8-6 Specification and management of Safety requirements

Req ID	Text	Type	ASIL class.	Component allocation	Release	Linked to	Justification	Product Config.	Status
1	**FUNCTIONAL REQUIREMENTS**	Head.	NA	NA	NA	NA	NA	NA	NA
SR1.1.1	The software shall acquire FlashingIndicatorStatus from CAN every 100 ms	Req.	QM	LKA_Enable	Rel. 1	TR1.1	N/A	All	Accepted
SR1.2.1	The software shall set lka_authorization to false when turn signal value is “left” or “right”	Req.	QM	LKA_Enable	Rel. 1	TR1.2	N/A	All	Accepted
SR2.1.1	The software shall acquire BrakeInfoStatus from CAN every 20 ms	Req.	QM	LKA_Enable	Rel. 1	TR2.1	N/A	All	Accepted
SR2.2.1	The software shall set lka_authorization to false when brake pedal signal value is “pedal pressed”	Req.	QM	LKA_Enable	Rel. 1	TR2.2	N/A	All	Accepted
SSR2.3.1	The software shall monitor Clock and CRC for BrakeInfoStatus	Req.	ASIL B	LKA_Enable	Rel. 1	TSR2.3	N/A	All	Accepted
SSR2.4.1	The software shall set lka_authorization to false if Clock or CRC for BrakeInfoStatus is erroneous	Req.	ASIL B	LKA_Enable	Rel. 1	TSR2.4	N/A	All	Accepted
SR2.4.2	The software shall set lka_authorization to false if BrakeInfoStatus is unavailable	Req.	ASIL B	LKA_Enable	Rel. 2	TSR2.4	N/A	Conf. A	In work
SR0.0.1	Store lka_authorization data in RAM from section x to y	Req.	QM	LKA_Enable	Rel. 2	SYSREQ0	N/A	Conf. A	In work
SSR0.0.2	Store lka_authorization with a CRC in protected RAM	Req.	ASIL B	LKA_Enable	Rel. 1	Derived	Safety analysis	Conf. A	Ready for review
SSR0.0.3	Check lka_authorization ‘s CRC before using it. If the check fails, use the default FALSE value.	Req.	ASIL	LKA_Enable	Rel. 1	Derived	Safety analysis	Conf. A	Ready for review

Renault example in ***Annex: Requirements refinement and allocation @Renault***

6-6 Specification of software safety requirements


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phases
Test phases
Design phase verification
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

Safety requirements shall be:

- Specified by an appropriate combination

  Natural language and

  Methods listed in this table (8-6)

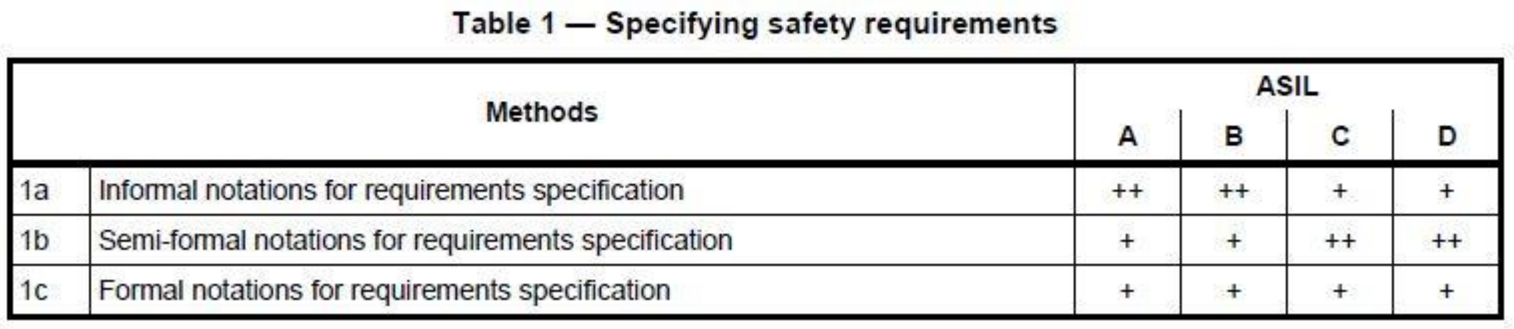


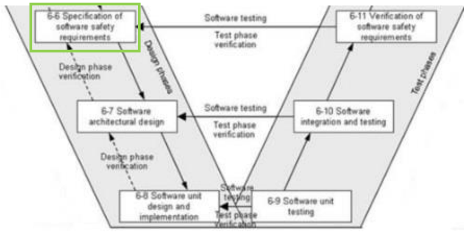
Table 1 — Specifying safety requirements

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

- Unambiguous, comprehensible, atomic, internal/external consistent, verifiable
- With a unique identification, status and an ASIL
- Completeness, maintainability and with no duplication of information

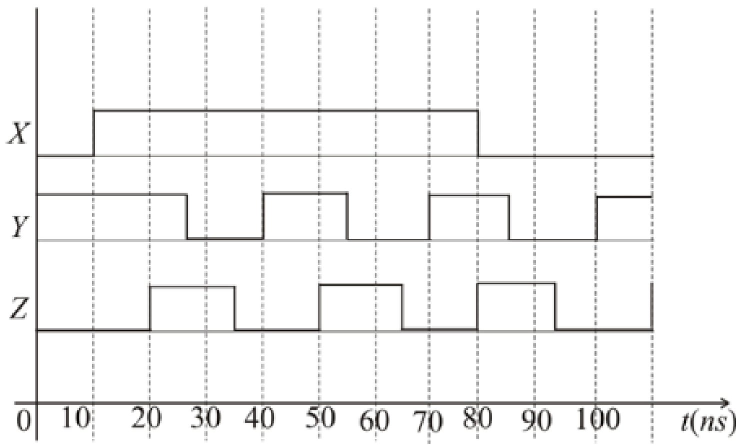
6-6 Specification of software safety requirements

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

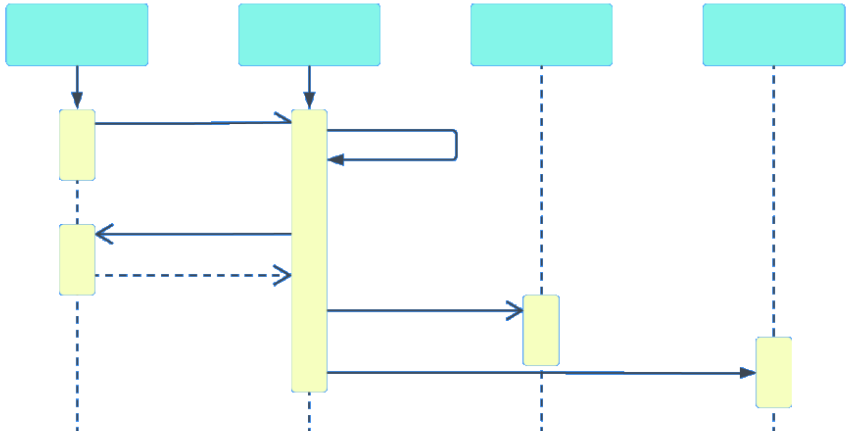

6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phases
Test phases
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase Verification
6-9 Software unit testing

- Informal diagram

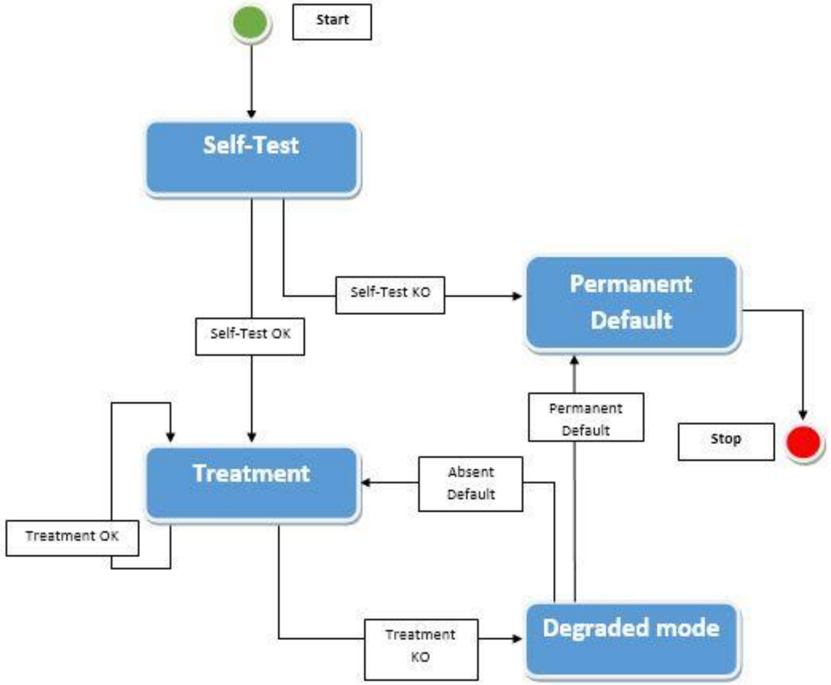
Timing Diagram



- Semi-formal diagram (UML sequence diagram)

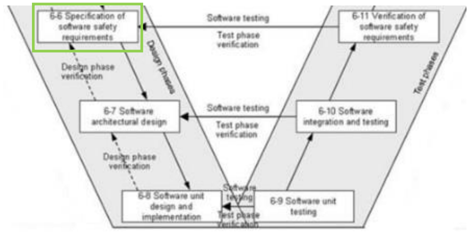


- Semi-formal State diagram



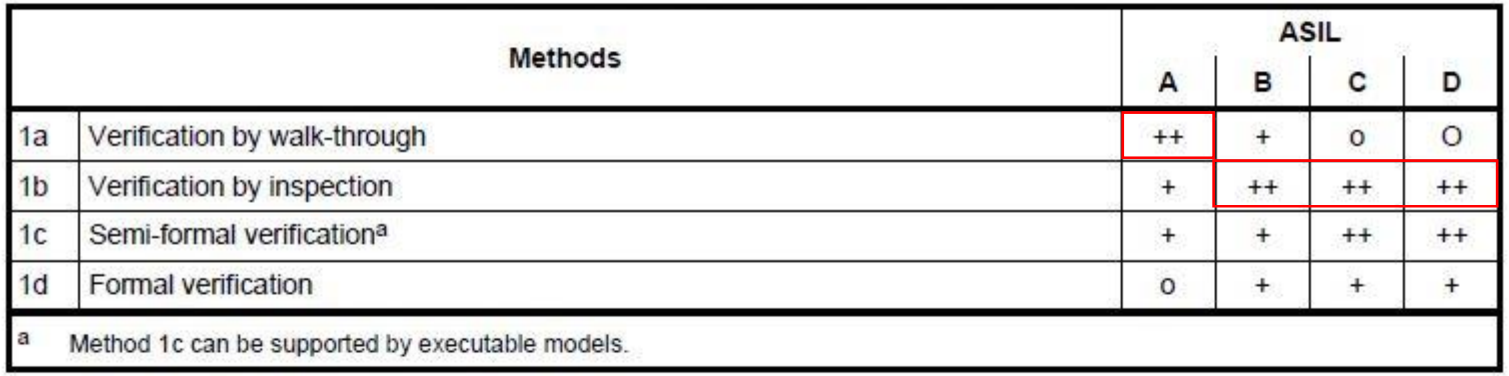
8-9 Verification

RENAULT-NISSAN-MITSUBISHI RESTRICTED B


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phases
Test phases
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

- An appropriate combination of the verification methods shall be applied (8-6)

Table 2 — Methods for the verification of safety requirements



Methods		ASIL			
		A	B	C	D
1a	Verification by walk-through	++	+	o	O
1b	Verification by inspection	+	++	++	++
1c	Semi-formal verification[a]	+	+	++	++
1d	Formal verification	o	+	+	+

[a] Method 1c can be supported by executable models.

- **Verification by walk-through**
  - Systematic examination of work products in order to detect anomalies

- **Verification by inspection**
  - Examination of work products, following a formal procedure, in order to detect anomalies
  - Formal procedure: includes a previously defined procedure, checklist, moderator and review of the results.

Exercise : Verification

8-9 Verification – Exercise 1

Verification by inspection

- Is the requirement comprehensible ?
- Is the requirement atomic ?
- Is the requirement internally/externally consistent ?
- Is the requirement feasible ?
- Is the requirement verifiable ?

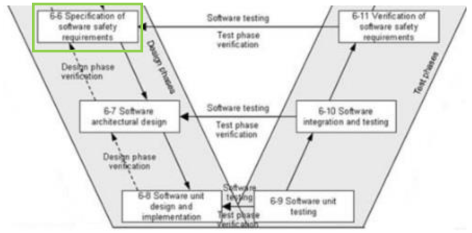
TR1.1 Turn signal state shall be acquired from CAN message FlashingIndicatorStatus every 100 ms
TR1.2 LKA shall be disabled when turn signal value is “left” or “right”
TSR1.3 The turn signal information shall be secured with Clock and CRC
TSR1.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled

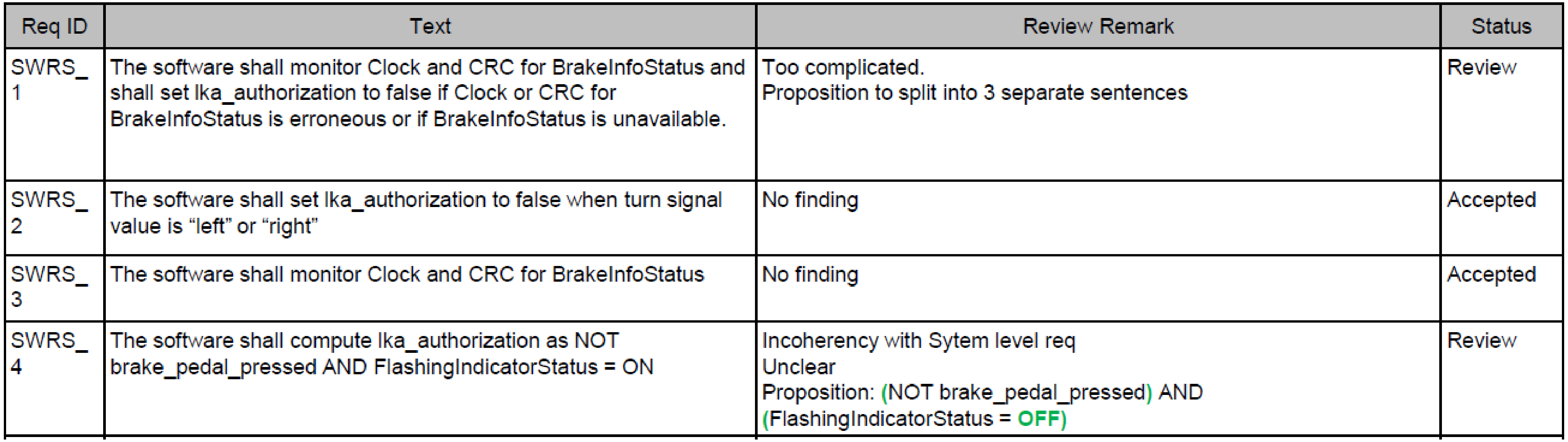
TR2.1 Consolidated BrakeInfoStatus shall be acquired from CAN every 20 ms
TR2.2 LKA shall be disabled when consolidated BrakeInfoStatus is “pedal pressed”
TSR2.3 The brake pedal information shall be secured with Clock and CRC
TSR2.4 If Clock or CRC is wrong, or if value is unavailable, LKA shall be disabled

Req ID	Text	Review Remark	Status
SWRS_1	The software shall monitor Clock and CRC for BrakeInfoStatus and shall set lka_authorization to false if Clock or CRC for BrakeInfoStatus is erroneous or if BrakeInfoStatus is unavailable.		Ready for review
SWRS_2	The software shall set lka_authorization to false when turn signal value is “left” or “right”		Ready for review
SWRS_3	The software shall monitor Clock and CRC for BrakeInfoStatus		Ready for review
SWRS_4	The software shall compute lka_authorization as NOT brake_pedal_pressed AND FlashingIndicatorStatus = ON		Ready for review

8-9 Verification – Exercise 1

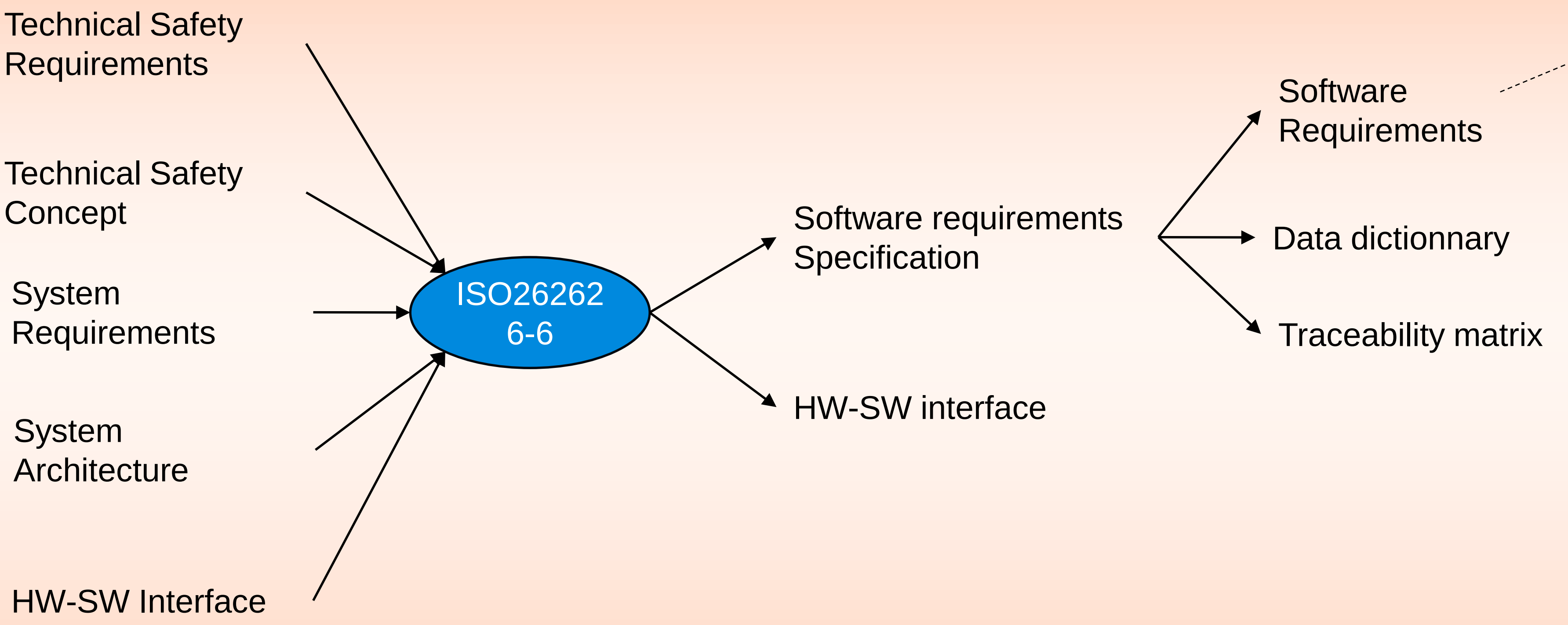
- **Verification by inspection**
  - Is the requirement comprehensible ?
  - Is the requirement atomic ?
  - Is the requirement internally/externally consistent ?
  - Is the requirement feasible ?
  - Is the requirement verifiable ?


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phases
Design phase verification
Test phases
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase Verification
6-9 Software unit testing



Req ID	Text	Review Remark	Status
SWRS_1	The software shall monitor Clock and CRC for BrakeInfoStatus and shall set lka_authorization to false if Clock or CRC for BrakeInfoStatus is erroneous or if BrakeInfoStatus is unavailable.	Too complicated. Proposition to split into 3 separate sentences	Review
SWRS_2	The software shall set lka_authorization to false when turn signal value is “left” or “right”	No finding	Accepted
SWRS_3	The software shall monitor Clock and CRC for BrakeInfoStatus	No finding	Accepted
SWRS_4	The software shall compute lka_authorization as NOT brake_pedal_pressed AND FlashingIndicatorStatus = ON	Incoherency with Sytem level req Unclear Proposition: (NOT brake_pedal_pressed) AND (FlashingIndicatorStatus = OFF)	Review

Summary
Specification of software safety requirements



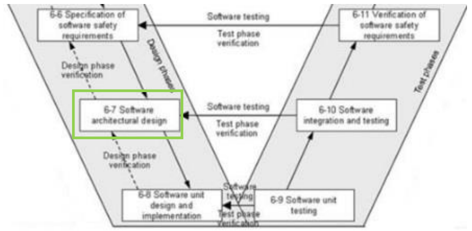
- Functional/Non-functional requirements
- Operating mode defined
- Constraint considered (timing, communication channel, HW constraints…)
- Attributes fulfilled

Clear requirement = **Understandable requirement**
(simple wording, use of diagrams/tables)

Understandable requirement = **safe requirement** (free of interpretation)

6-7 SW Architectural design

6-7 Software architectural design


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phase
Test phase
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

Objectives

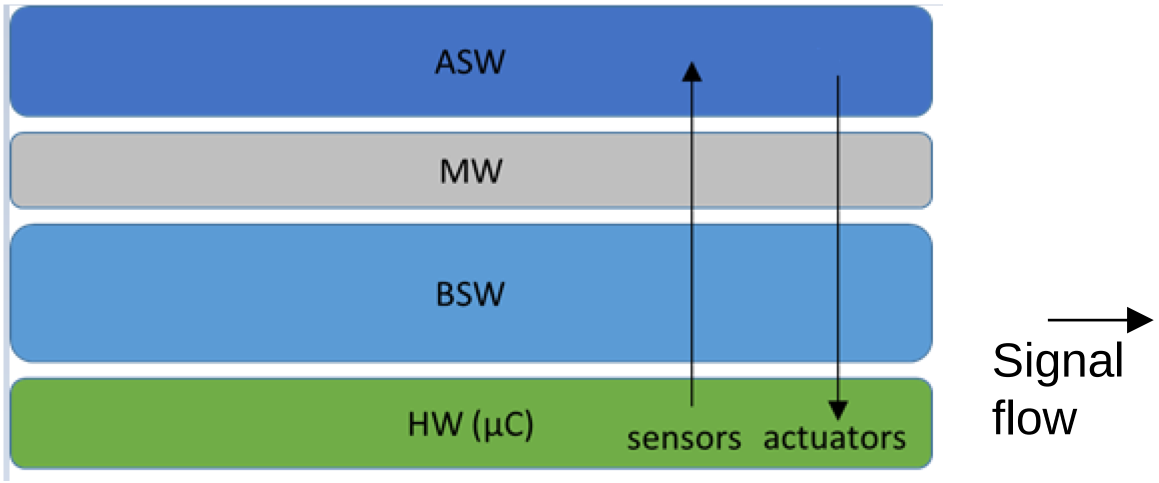
- Develop a software architectural design that realizes the software safety requirements
- Verify consistency of software architectural design

Work products

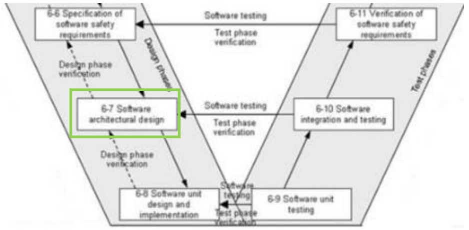
- Software architectural design Specification
- Safety analysis report
- Dependent failures analysis report
- Software verification report (updated)

SW Layers & AUTOSAR

- **SW Architecture definition: Structure of the Software that identify SW elements with their boundaries and interfaces.**
- **In Automotive industry, the SW is made of 3 layers:**
  - Applicative SW (ASW): Define the behavior, requirements at this level are very similar to System requirements
  - Basic SW (BSW): SW layer in direct relationship with the HW components. It has various roles: HW drivers and abstract data from HW components to allocate them to the ASW.
  - MiddleWare (MW or RTE for AutoSAR): Interface layer between the ASW and the BSW. It is made of standardized interfaces in order to ease the reusability of the ASW

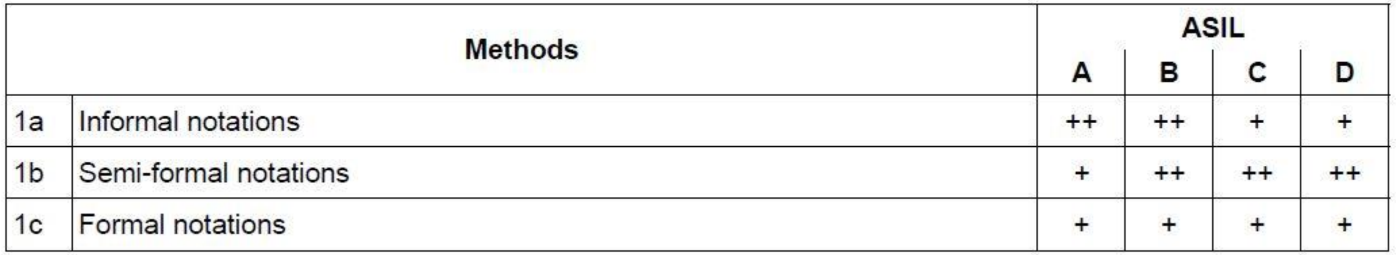


6-7 Software architectural design


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phase
Test phases
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

- Software architectural design shall be described by using the notations listed in Table 2.

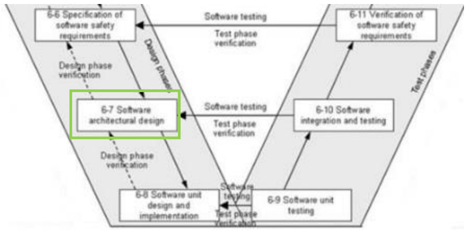
Table 2 — Notations for software architectural design



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

- Software architectural design shall be considered:
  - Verifiability (bi-directional traceability)
  - Suitability for configuration software
  - Feasibility for the design
  - Testability and Maintainability

6-7 Software architectural design


6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phase
Test phase
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

- The software architectural design shall exhibit the following properties (listed in Table 3):
  - Modularity
  - Encapsulation
  - Simplicity

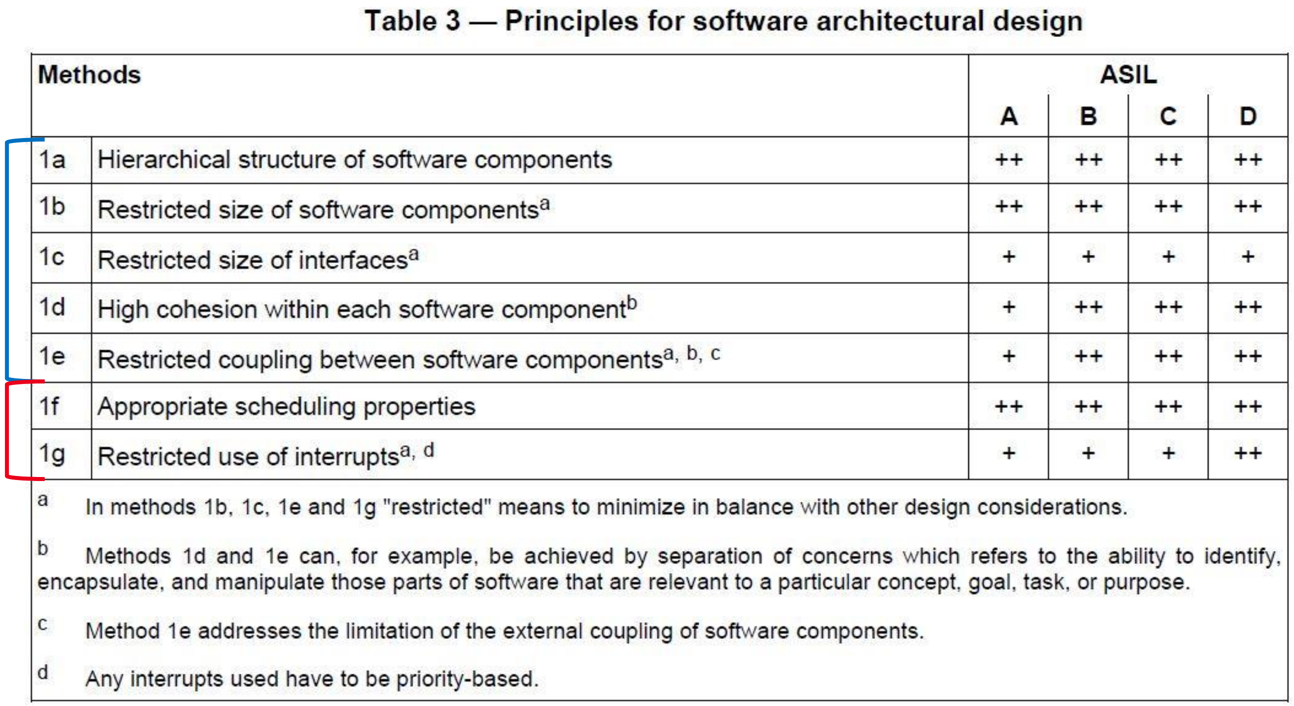


Table 3 — Principles for software architectural design

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.

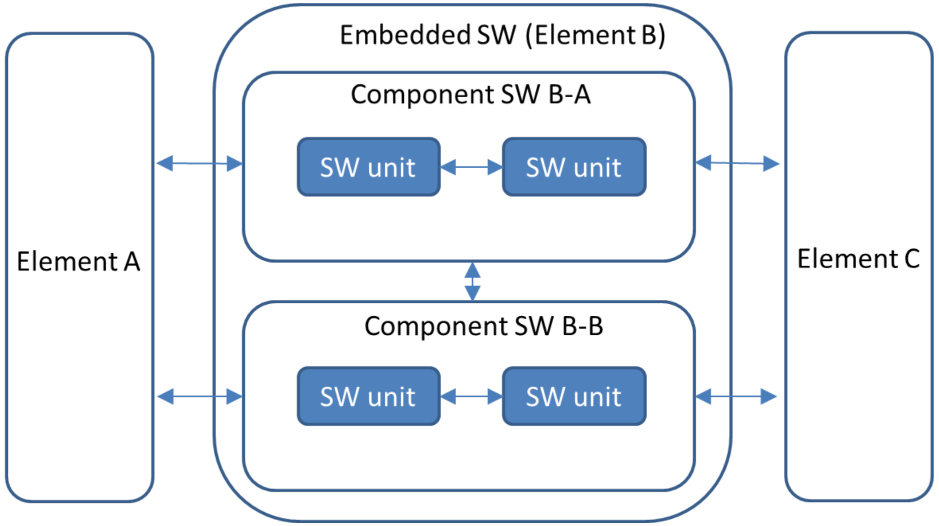
[b] Methods 1d and 1e can, for example, be achieved by separation of concerns which refers to the ability to identify, encapsulate, and manipulate those parts of software that are relevant to a particular concept, goal, task, or purpose.

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

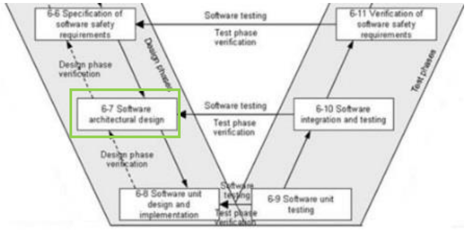
[d] Any interrupts used have to be priority-based.

Static architecture

Dynamic architecture



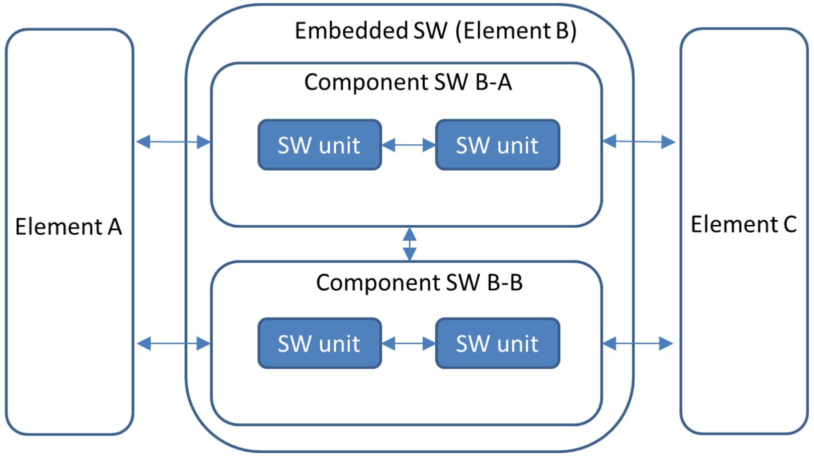
6-7 Software architectural design

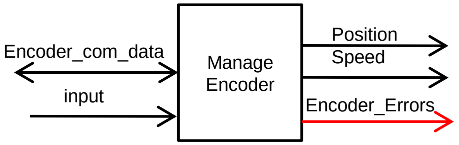

6-6 Specification of software safety requirements
Software testing
Test phase verification
6-11 Verification of software safety requirements
Design phase verification
Design phase
6-7 Software architectural design
Software testing
Test phase verification
6-10 Software integration and testing
Test phases
Design phase verification
6-8 Software unit design and implementation
Software testing
Test phase verification
6-9 Software unit testing

Component B-A:
- ASIL B
- Sched. at 20ms

Component B-B:
- ASIL C
- Sched. at 10ms

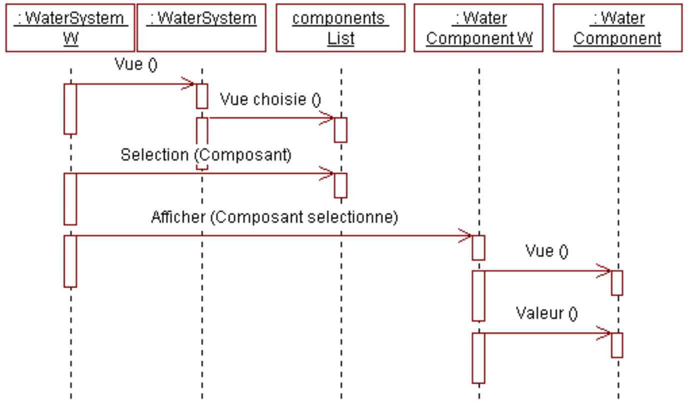
Static architecture design

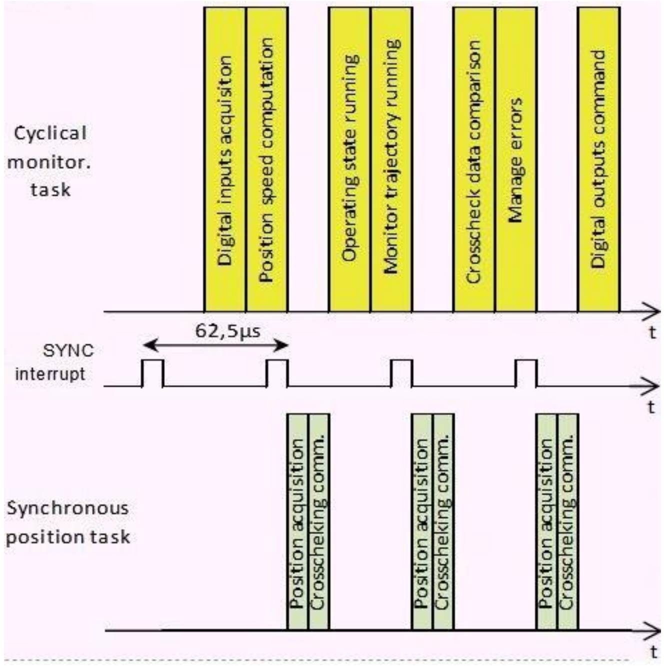




Data name	Data type	Frequency
Encoder_com_data	uint32	20 ms
input	uint16	10 ms
Position	int16	20 ms
Speed	uint16	20 ms
Encoder_Errors	Enum	20 ms

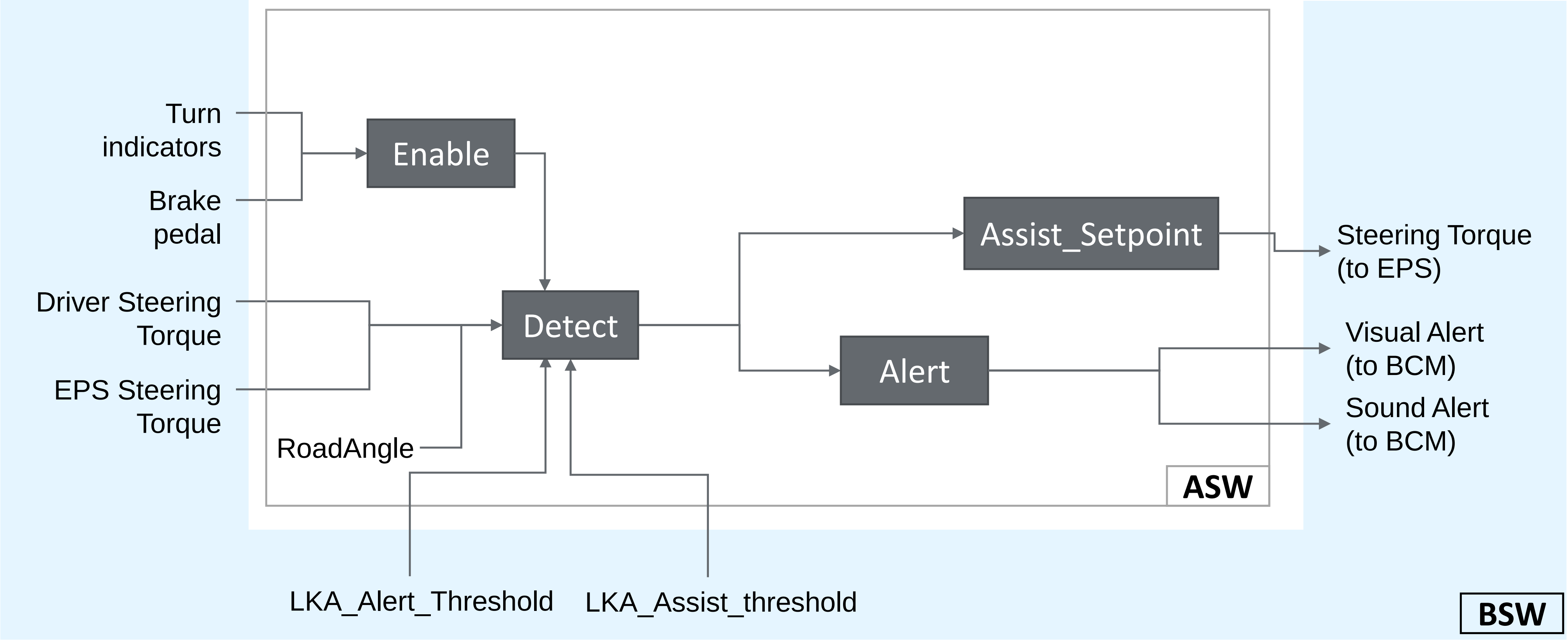
Dynamic architecture design





6-7 Software architectural design LKA example

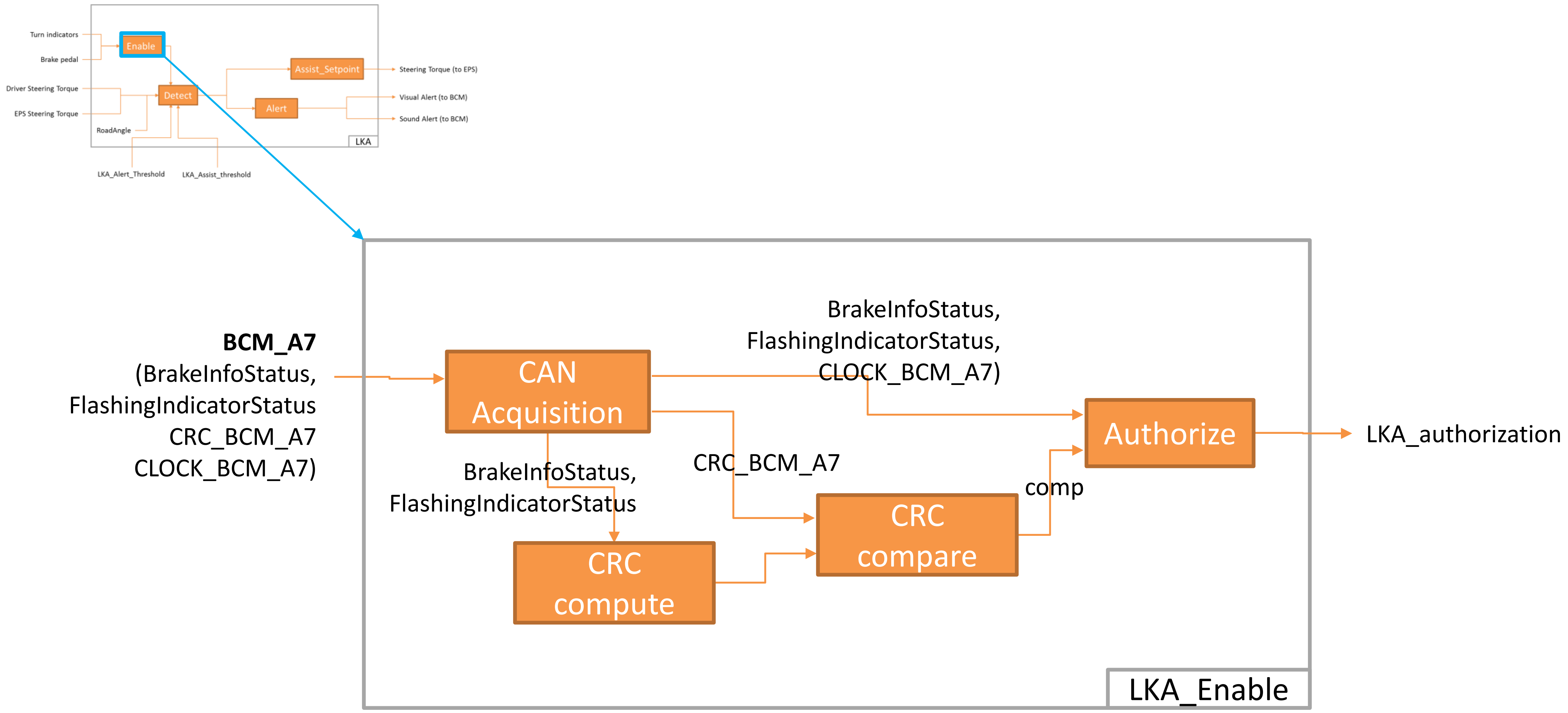
RENAULT-NISSAN-MITSUBISHI RESTRICTED B



6-7 Software architectural design
LKA example: LKA_Enable module

SWR Req ID	SWR Text	SWA Req ID	SWA Text	Allocation
SR2.1.1	The software shall acquire BrakeInfoStatus from CAN every 20 ms	SAR2.1.1.1	The LKA_Enable module shall acquire BrakeInfoStatus from CAN every 20 ms	LKA_Enable
SSR2.3.1	The software shall monitor Clock and CRC signals for BrakeInfoStatus	SSAR2.3.1.1	The LKA_Enable module shall acquire Clock (Clock_BCM_A7) and CRC (CRC_BCM_A7) signals from BrakeInfoStatus CAN frame BCM_A7.	LKA_Enable
SSR2.4.1	The software shall set lka_authorization to false if Clock or CRC for BrakeInfoStatus is erroneous	SSAR2.4.1.1	The LKA_Enable module shall set lka_authorization to false when Clock_BCM_A7 is stuck.	LKA_Enable
		SSAR2.4.1.2	The LKA_Enable module shall compute the CRC of received BCM_A7 frame. If it differs from CRC_BCM_A7, LKA_Enable module shall set lka_authorization to false.	LKA_Enable
SSR2.4.2	The software shall set lka_authorization to false if BrakeInfoStatus is unavailable	SSAR2.4.2.1	The LKA_Enable module shall set lka_authorization to false when BrakeInfoStatus value is “unavailable”.	LKA_Enable
SR1.2.1 SR2.2.1	The software shall set lka_authorization to false when turn signal value is “left” or “right” The software shall set lka_authorization to false when brake pedal signal value is “pedal pressed”	SSAR1.2.1.1	The LKA_Enable module shall output lka_authorization. Init value : False	LKA_Enable
		SSAR1.2.1.2	The LKA_Enable module shall set lka_authorization value to false when turn signal is “left” or “right” or brake pedal is “pedal pressed”; and to true otherwise.	LKA_Enable
-	-	SAR0.1	The LKA_Enable module shall be executed every 20ms.	LKA_Enable

6-7 Software architectural design LKA example



Turn indicators

Brake pedal

Driver Steering Torque

EPS Steering Torque

RoadAngle

Enable

Detect

Assist_Setpoint

Alert

LKA

Steering Torque (to EPS)

Visual Alert (to BCM)

Sound Alert (to BCM)

LKA_Alert_Threshold

LKA_Assist_threshold

BCM_A7
(BrakeInfoStatus,
FlashingIndicatorStatus
CRC_BCM_A7
CLOCK_BCM_A7)

CAN Acquisition

BrakeInfoStatus,
FlashingIndicatorStatus,
CLOCK_BCM_A7)

CRC_BCM_A7

BrakeInfoStatus,
FlashingIndicatorStatus

CRC compute

CRC compare

comp

Authorize

LKA_authorization

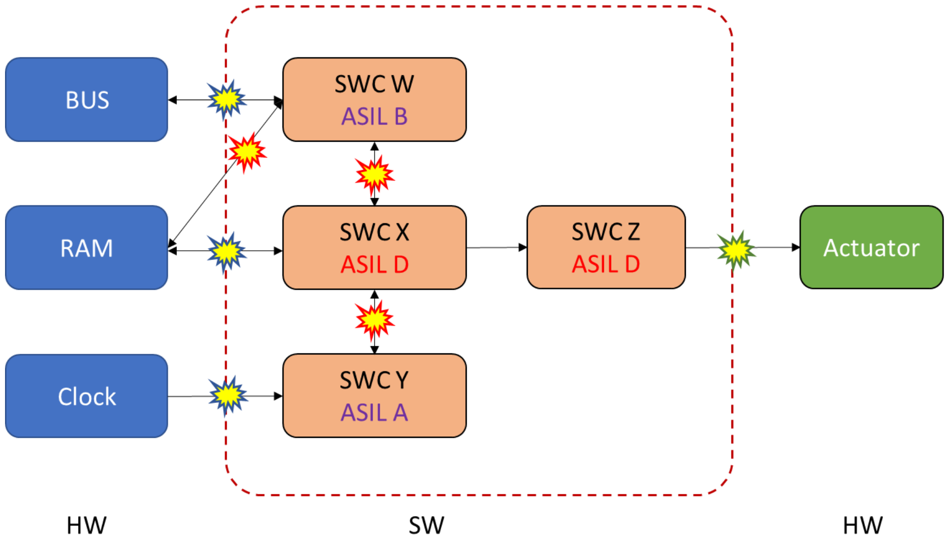
LKA_Enable

Safety software analysis

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Purpose of safety-oriented analyses (1/2)

- **Software safety analyses have two objectives:**
  - Design a safe product by using Software safety mechanisms covering Hardware weaknesses
  - Design a safe software not introducing fault leading to Safety goals violation



3 types of failures to check

Inputs from HW to SW
➔ Safety mechanisms defined by the HW, implemented by the SW

Dependent Failure Analysis

Outputs from SW to actuator
➔ Check possible SG violation based on guidewords
(eg: « NO » can be due to a function never called)

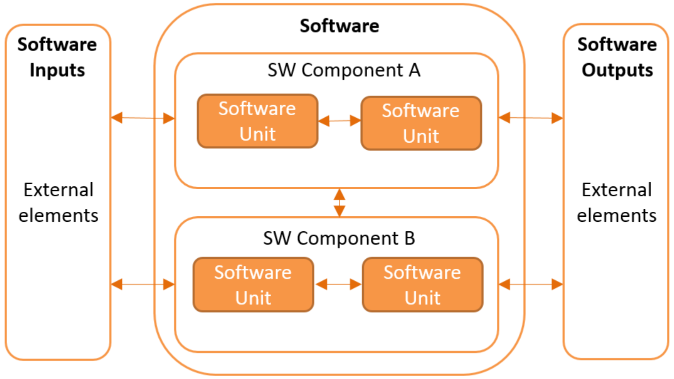
SWC = Software Component

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Principle of software safety analyses

Principle

- Dysfunctional analysis: propagate faults/failures in the software to identify potential occurrence of USWEs (Unwanted SoftWare Events)



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Purpose & principle: Synthesis

Safety analyses allow to:

- Ensure the non violation of Safety Goals
- Check the sufficiency of Safety mechanisms currently implemented
- If needed, identify new Safety mechanisms to be implemented, or Safety measures
- In the frame of Safety analyses, implementation of Safety mechanisms is preferable

Safety mechanisms

- **Safety mechanisms for error detection** (ISO 26262-6:2018 7.4.12)
  - **Range checks** of input and output data;
  - **Plausibility check** (e.g. using a reference model of the desired behaviour, assertion checks, or comparing signals from different sources);
  - **Detection of data errors** (e.g. error detecting codes and multiple data storage);
  - **Monitoring of program execution** by an external element such as an ASIC or another software element performing a watchdog function. Monitoring can be logical or temporal monitoring or both;
  - **Temporal monitoring of program execution**;
  - **Diverse redundancy in the design**;
  - **Access violation control mechanisms** implemented in software or hardware concerned with granting or denying access to safety-related shared resources
  - …

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Redundancy types

Redundancy can be on:

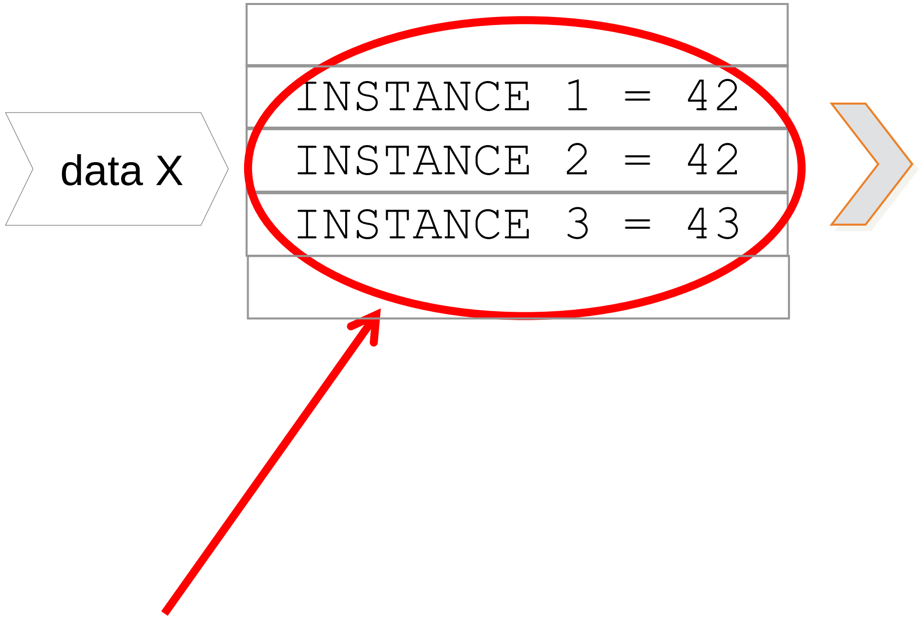
- HW
  - Sensors/actuators
  - ECU/IC
- Development process
  - Specification diversity
  - Independent development teams
  - Different development tools
- SW
  - Data redundancy
  - Temporal redundancy

Strength	Weakness
Prevent multiple HW failure	Integration cost Production cost
Prevent: design fault, implementation fault, tool fault introduction	Human Resources availability Development cost Tool cost
Prevent HW fault, Allow recovery	CCF on the ‘voter’ Need inputs independency CPU load

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Error detection solutions – Redundancy (1/6)

_ Data redundancy



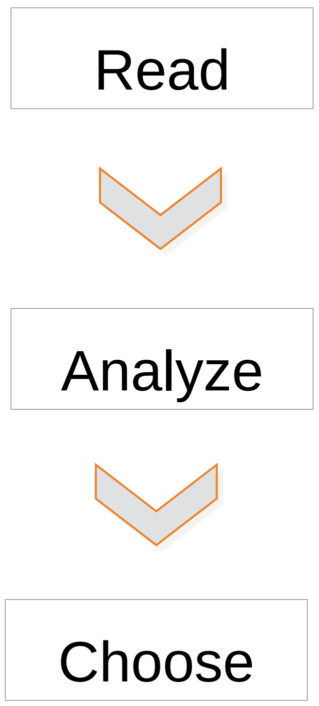
data X

INSTANCE 1 = 42

INSTANCE 2 = 42

INSTANCE 3 = 43

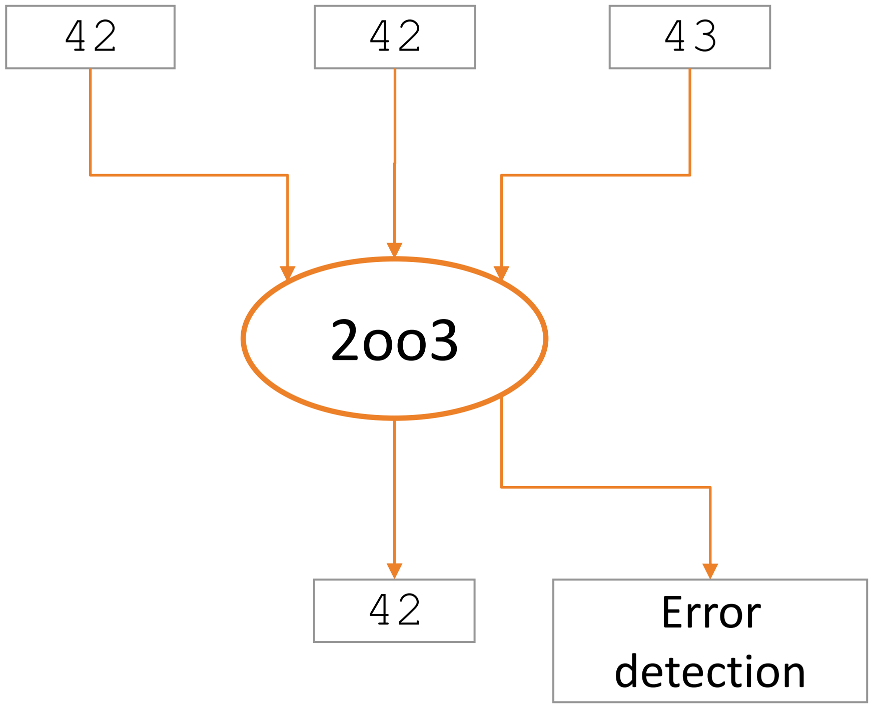
Data storage redundancy



Read

Analyze

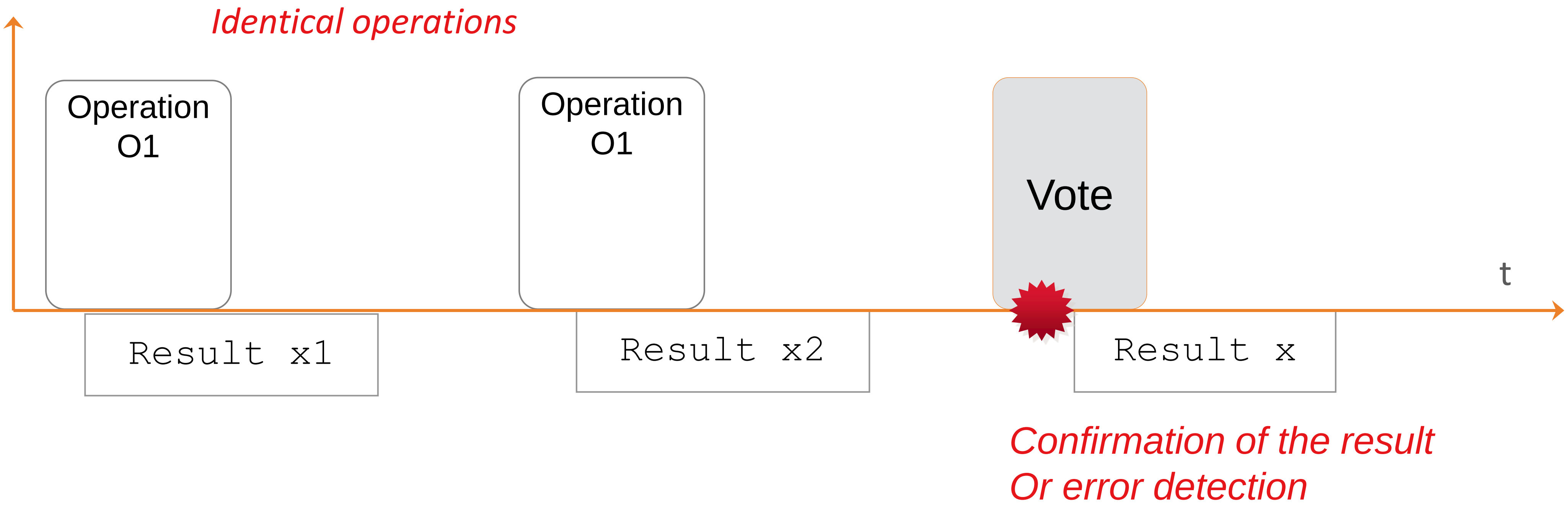
Choose



Error detection solutions – Redundancy (2/6)

Temporal redundancy

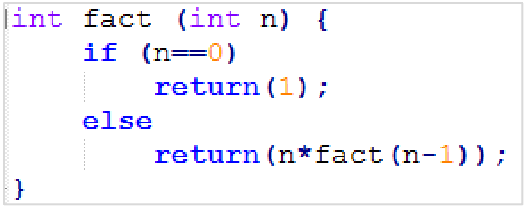
- Use the same algorithm at 2 different clock ticks to get the same result



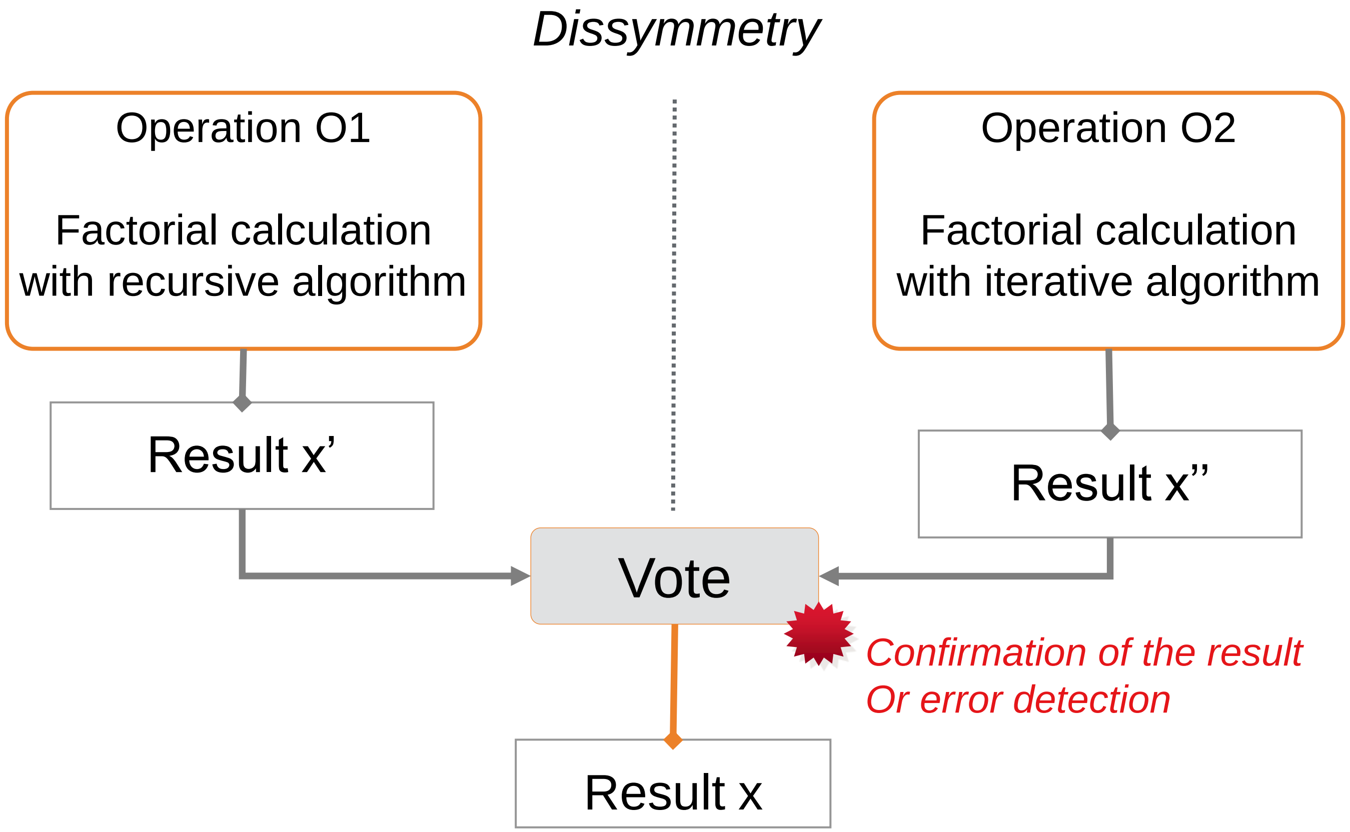
Error detection solutions – Redundancy (3/6)

Diverse redundancy in the SW design

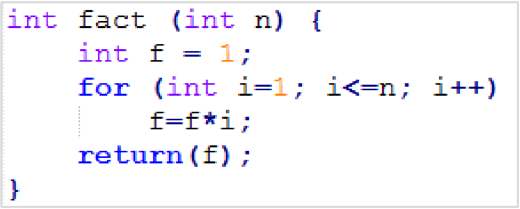
- Use 2 separate methods to get the same result



```
int fact (int n) {
    if (n==0)
        return(1);
    else
        return(n*fact(n-1));
}
```



Dissymmetry
Operation O1
Factorial calculation with recursive algorithm
Result x'
Operation O2
Factorial calculation with iterative algorithm
Result x''
Vote
Result x
Confirmation of the result
Or error detection

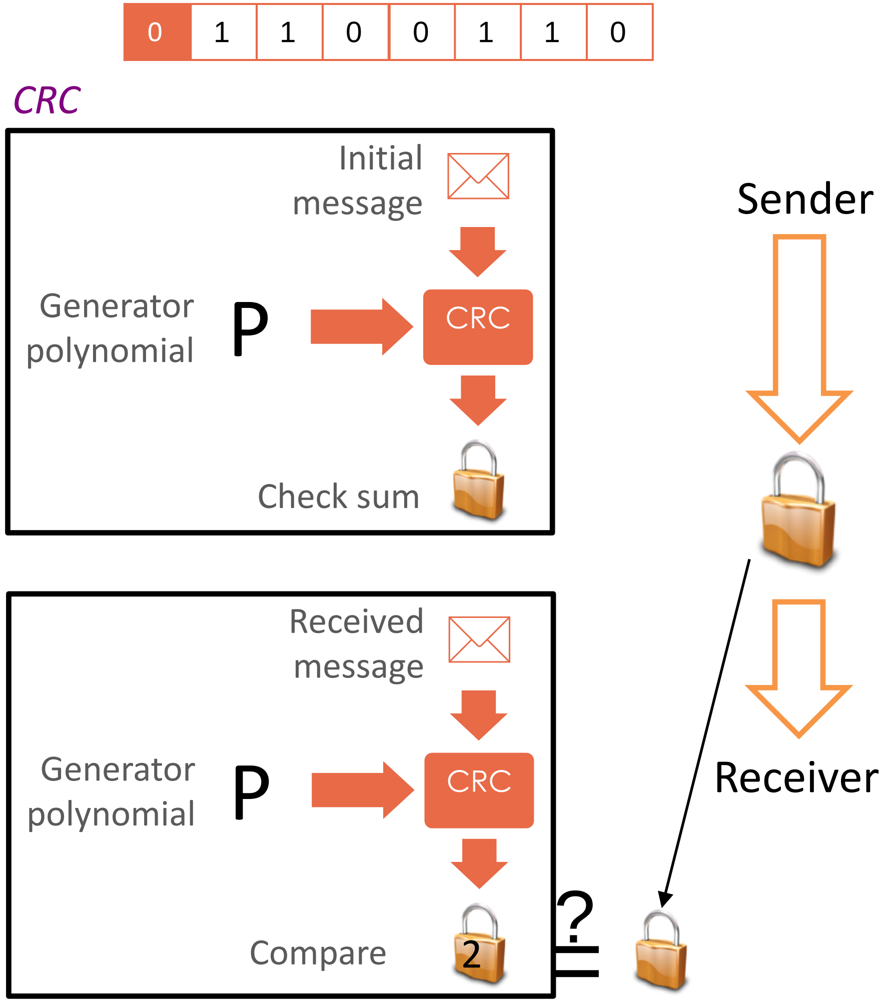


```
int fact (int n) {
    int f = 1;
    for (int i=1; i<=n; i++)
        f=f*i;
    return(f);
}
```

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Error detection solutions – Detection of data errors

- Error detecting codes (not to be confused with error correcting codes)
- Examples
  - Parity bit
  - Checksum
  - Hamming code: detecting and correction code
  - CRC (Cycle Redundancy Check): very reliable
- Prerequisites
  - Generator polynomial



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Error detection solutions – Range/Plausibility checks

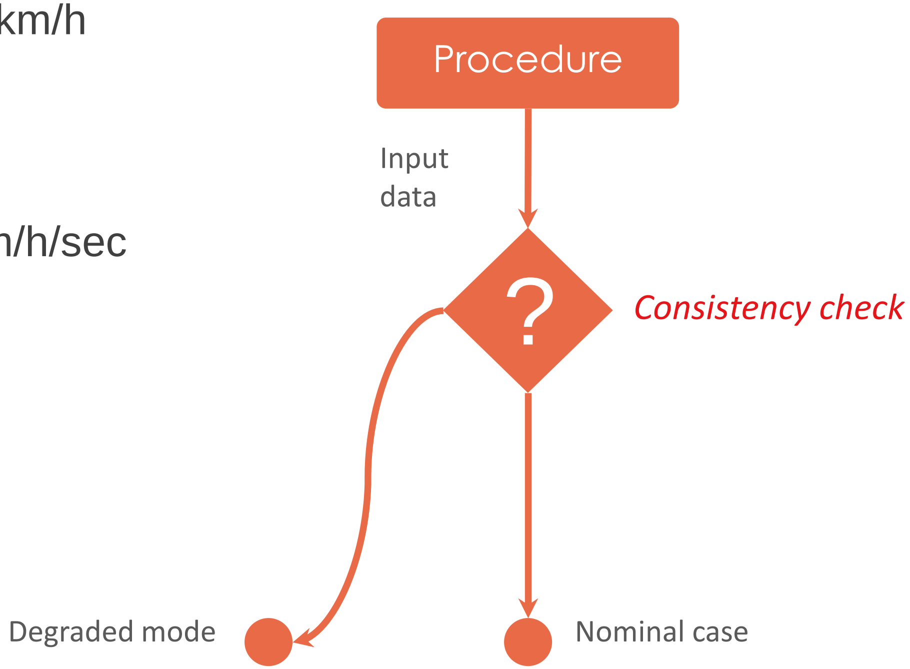
Consistency checks

- Boundary checks

  If TrainSpeed < 0km/h OR TrainSpeed > 350 km/h

- Dynamic checks

  If (TrainSpeed – lastTrainSpeed)/period > 2km/h/sec



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Error detection solutions – Temporal monitoring (1/3)

Safety mechanisms:

_ Task deadline monitoring

- Check tasks execution time against specified maximum value. This can be done by **watchdog**
- The scheduler is the system process that is in charge of handling processes scheduling
- A **scheduling algorithm** is a strategy used to schedule processes
- This algorithm relies on specific properties of the processes/system

  Periodic or aperiodic processes;

  Deadline system

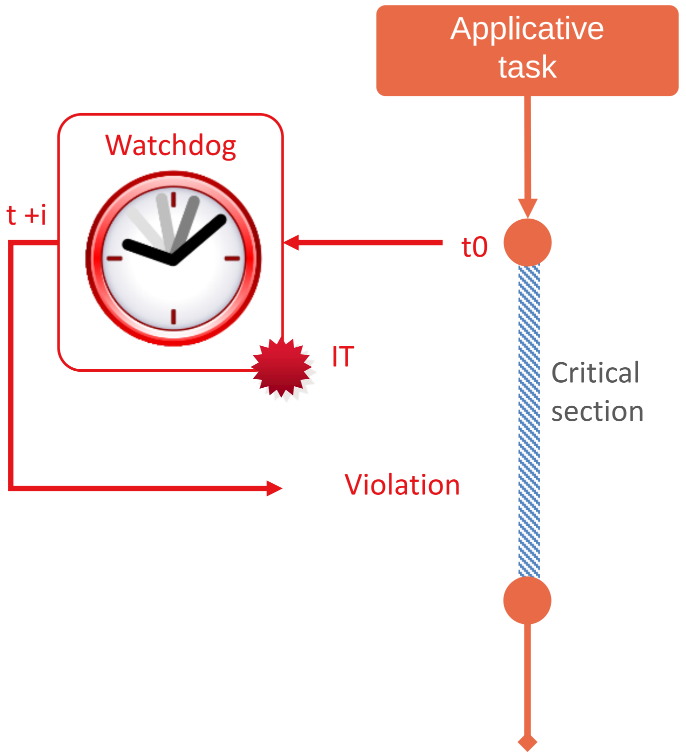
  Fixed-priority

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Error detection solutions – Temporal monitoring (2/3)

- **Watchdog (SW or HW device with a timer)**
  - Counter that is periodically incremented. Has to be reset before it reaches a certain value
  - HW
    Triggers a NMI (Non-Maskable Interrupt) if it is not reset before a certain delay

    Independence with microcontroller
  - SW
    Triggers an ECU reset when a software counter reaches a certain value



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA (1/2)

3.1. Fault Tree Analysis (FTA)

Definition

Graphic model of the various parallel and sequential combinations of faults that will result in the occurrence of USWE

Deductive analysis method: starts from known effects and seek possible causes

Top down / Qualitative approach

FTA asks « Which component can be responsible of this effect ? »

Hypothesis

Existence of Hardware and Software failures

Principle

Identify relationship of events by building a tree composed of different gates

Identify dependent failures, responsible components for USWE

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

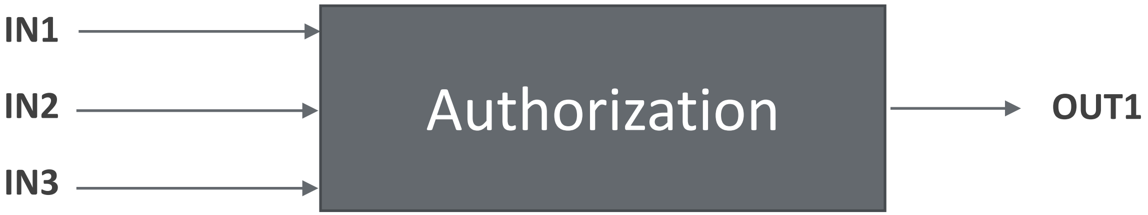
- **Authorization:**

  Input acquisition

  Consistency check

  Result comparison

  Send of authorization



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

- 3.1. Fault Tree Analysis (FTA)
  - Step 3.1.1 – Identification of SW output related to USWEs

ID	Unwanted Software Event (USWE)	ASIL inherited at software level	Related Software output
SWFE01	Authorization is given wrongly	ASIL B	OUT1

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

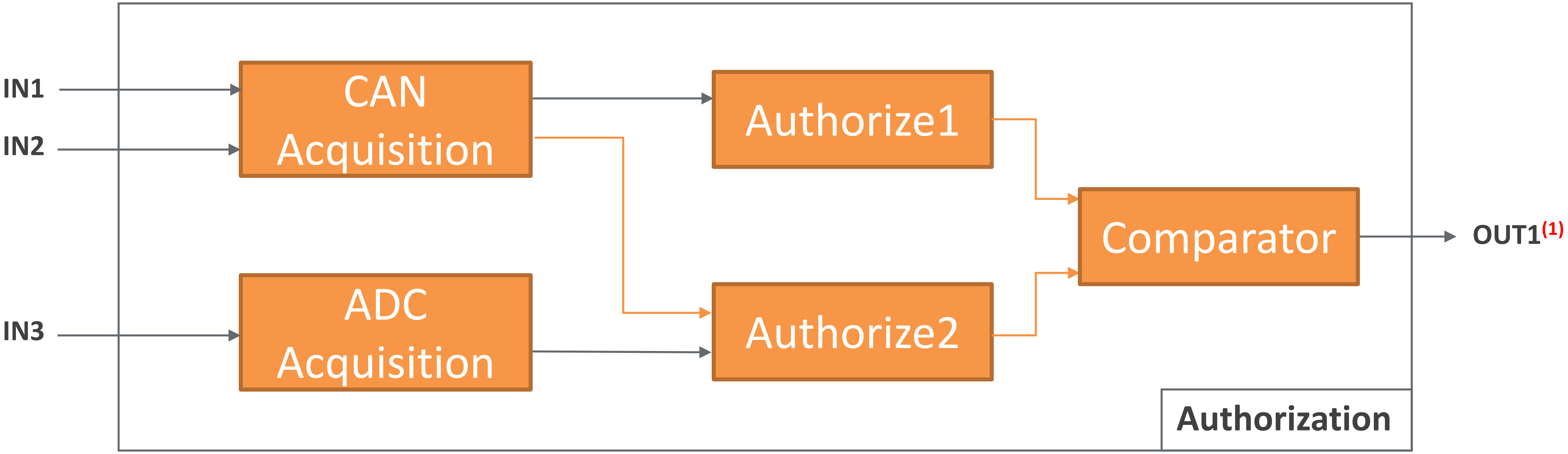
Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

Step 3.1.2 – Study of the model

Identification of the link between SW output and SW input

Example: SW Output related to SWFE01 **OUT1**(1)

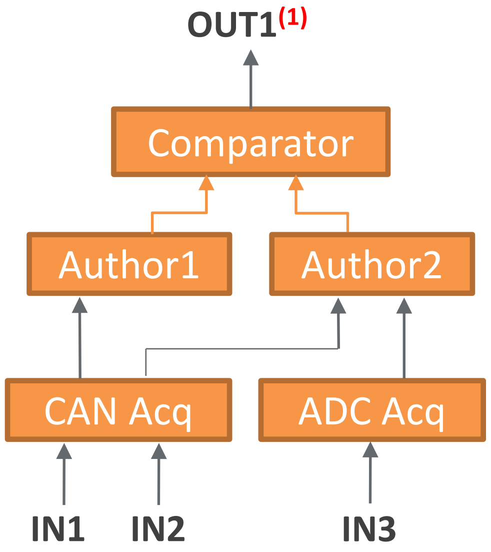


RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

- Step 3.1.3 – Application of failure modes & fault tree building



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

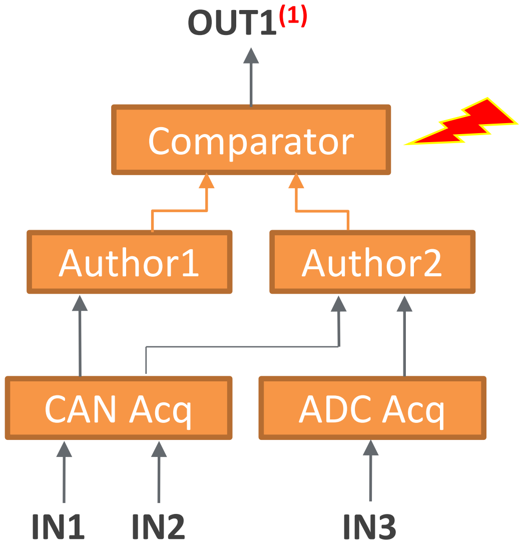
Exercise FTA

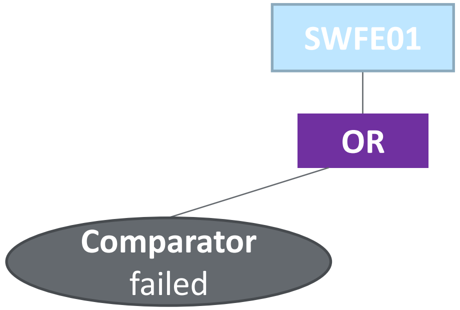
RENAULT-NISSAN-MITSUBISHI RESTRICTED B

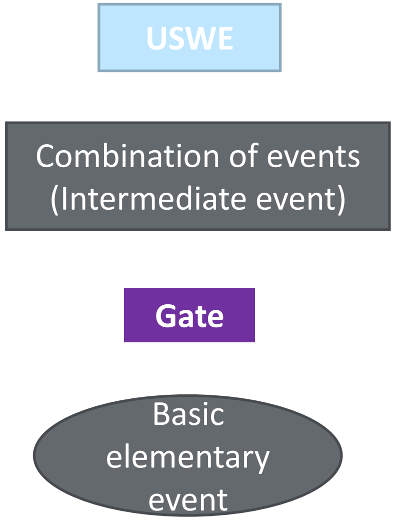
Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

Step 3.1.3 – Application of failure modes & fault tree building





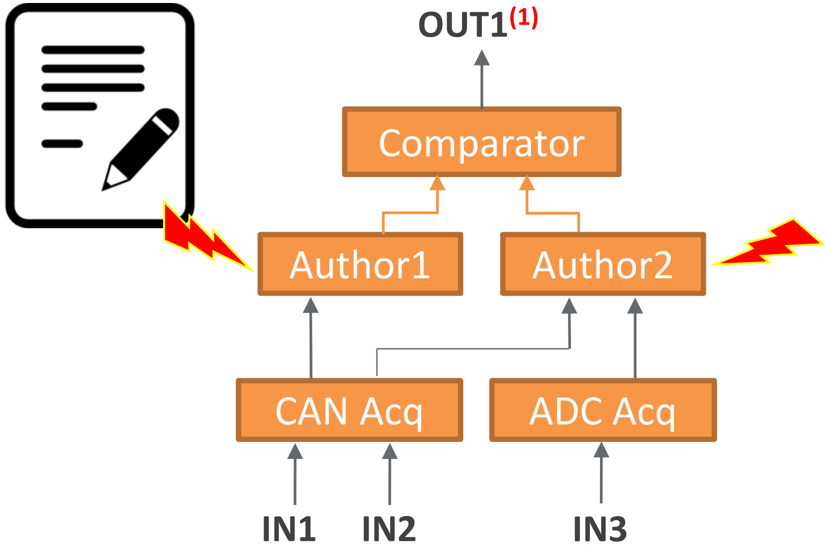


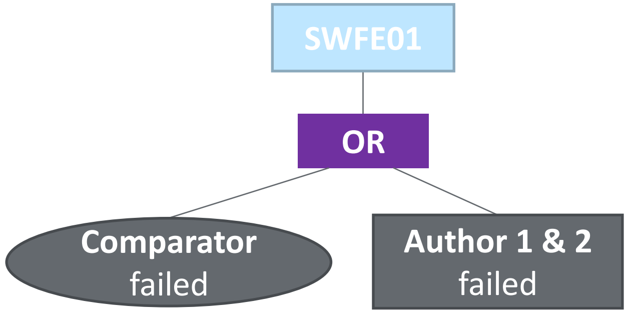
RENAULT-NISSAN-MITSUBISHI RESTRICTED B

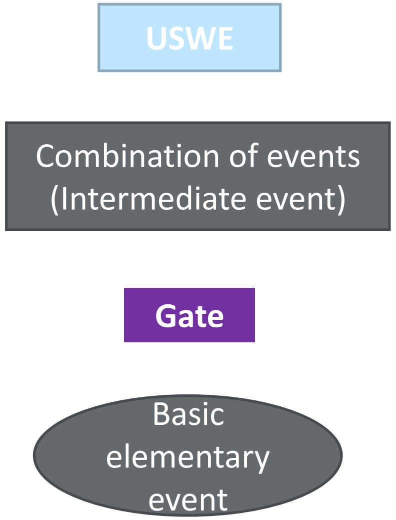
Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

- Step 3.1.3 – Application of failure modes & fault tree building



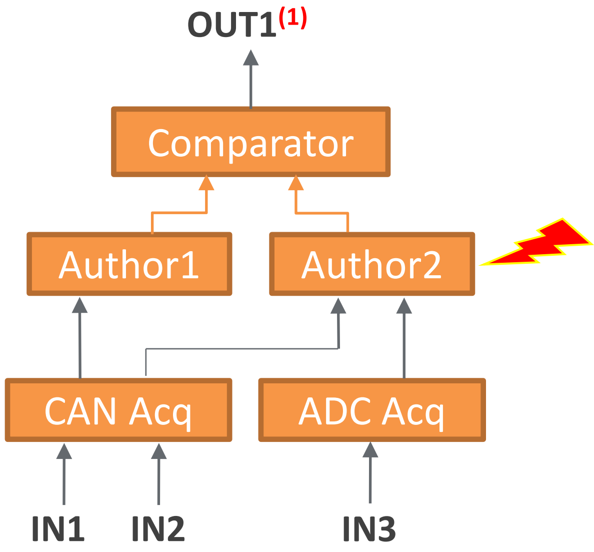


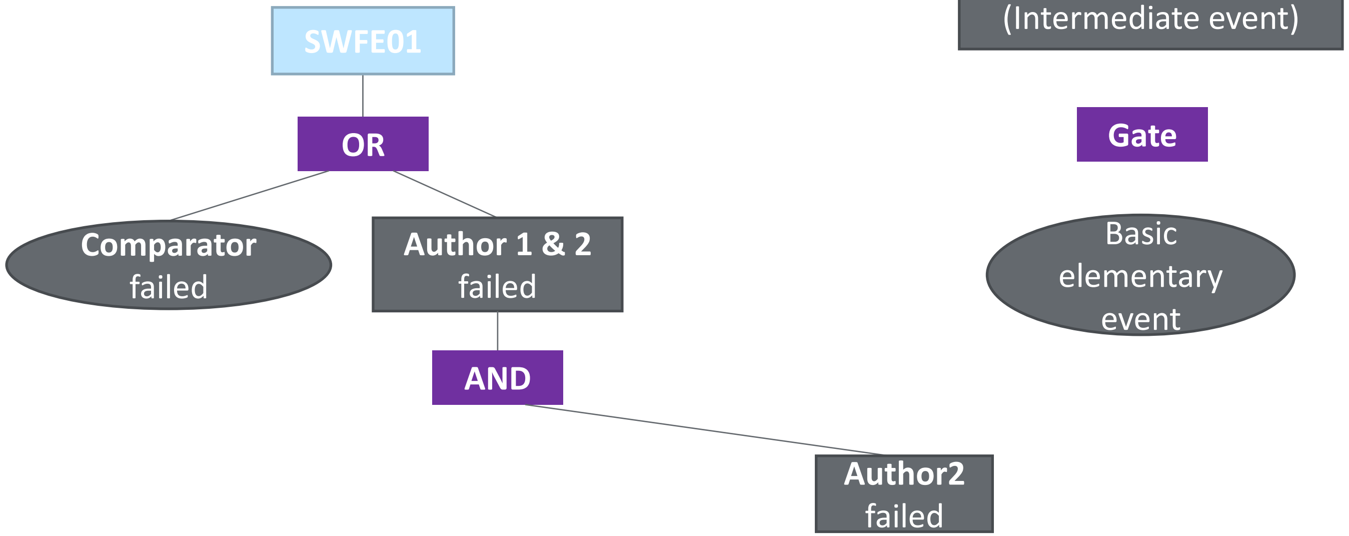


RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

- 3.1. Fault Tree Analysis (FTA)
  - Step 3.1.3 – Application of failure modes & fault tree building

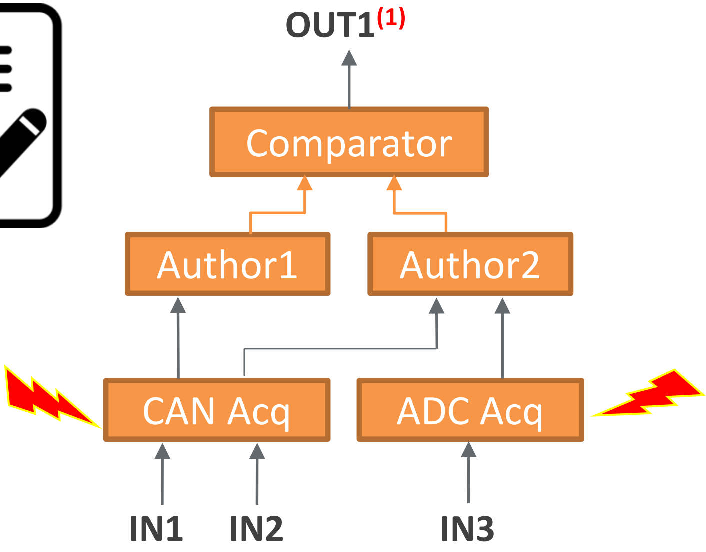


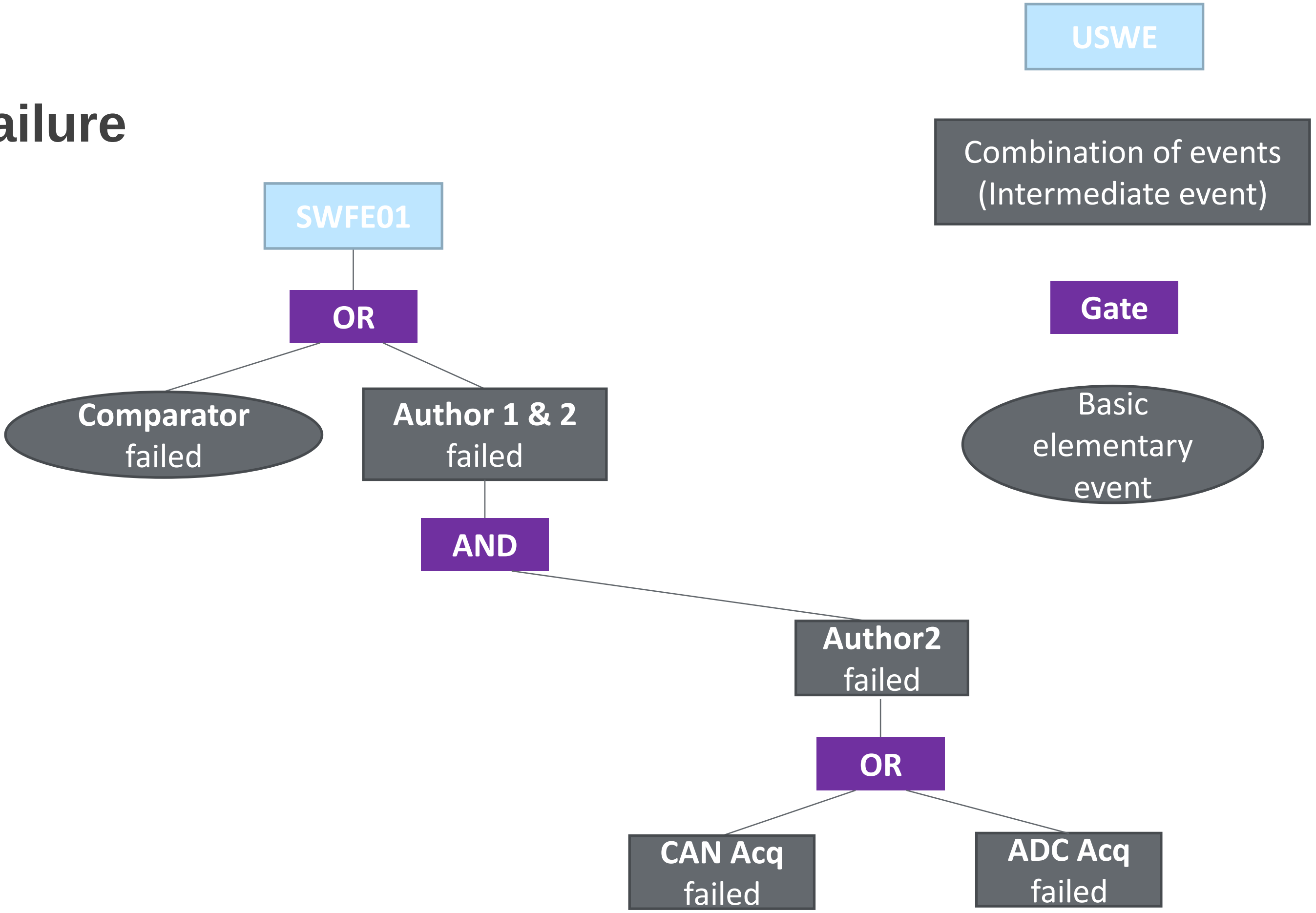


RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

- **3.1.** Fault Tree Analysis (FTA)
  - **Step 3.1.3 – Application of failure modes & fault tree building**

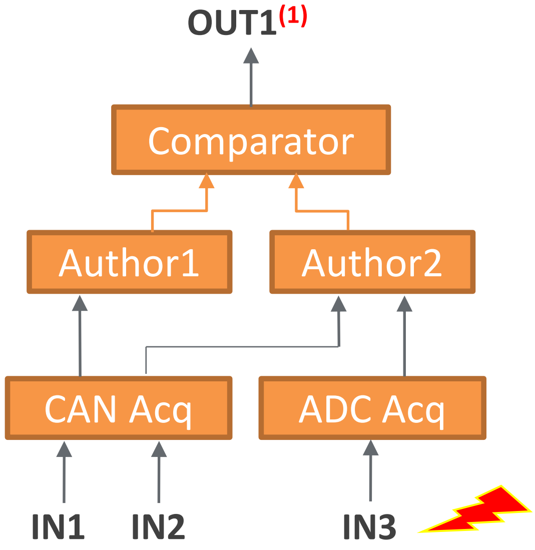


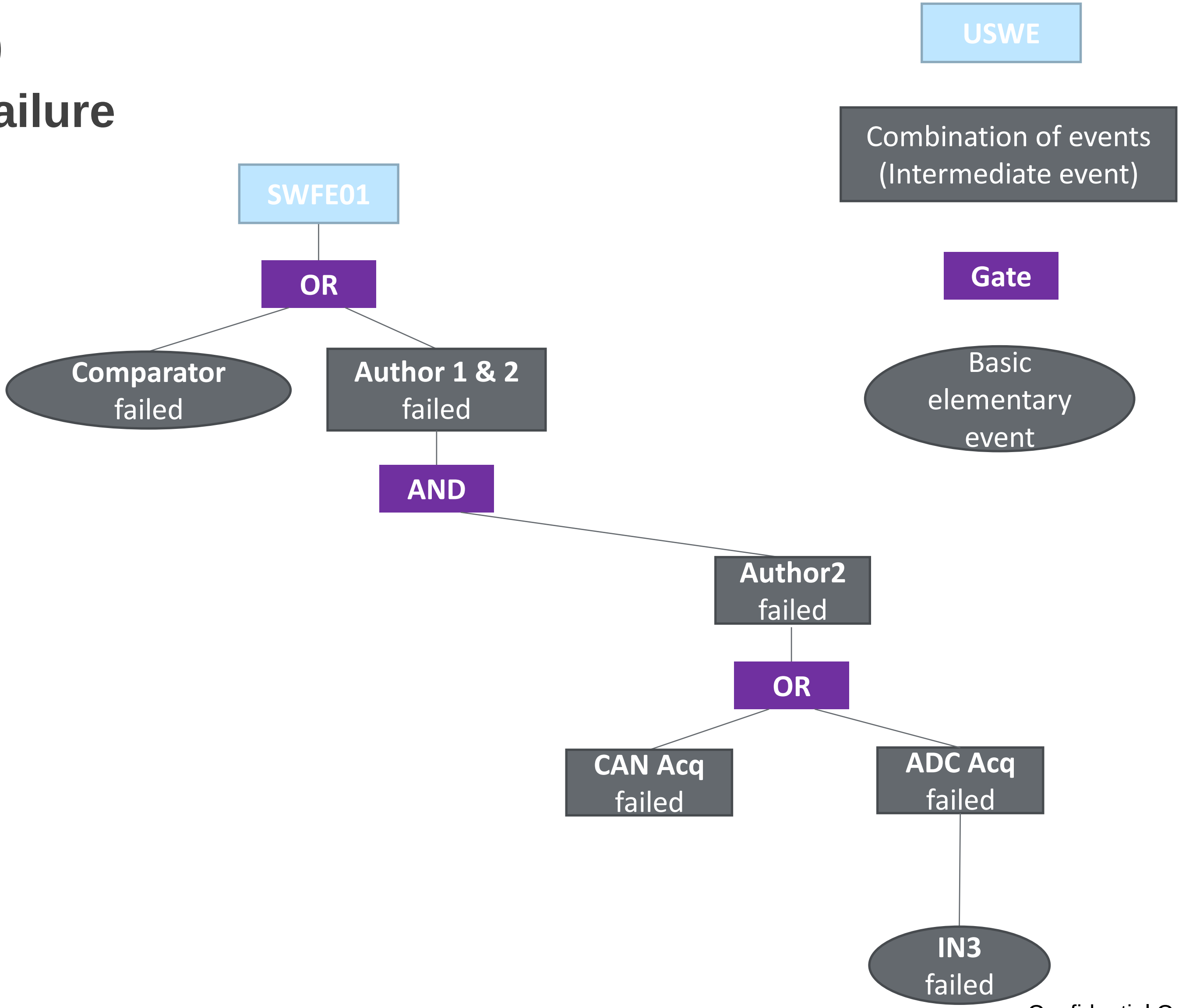


RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

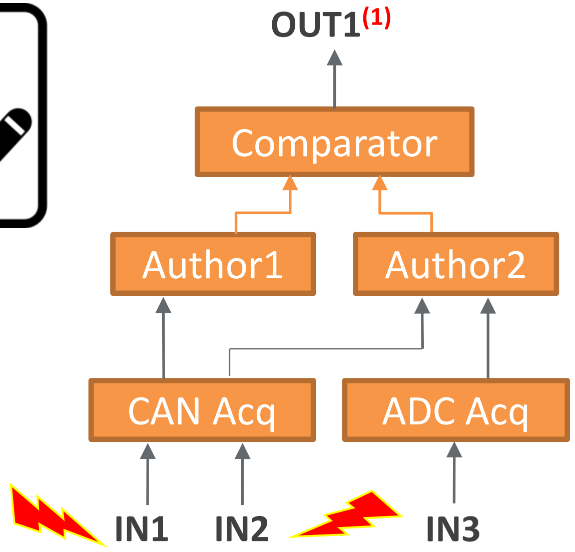
- **3.1.** Fault Tree Analysis (FTA)
  - **Step 3.1.3 – Application of failure modes & fault tree building**

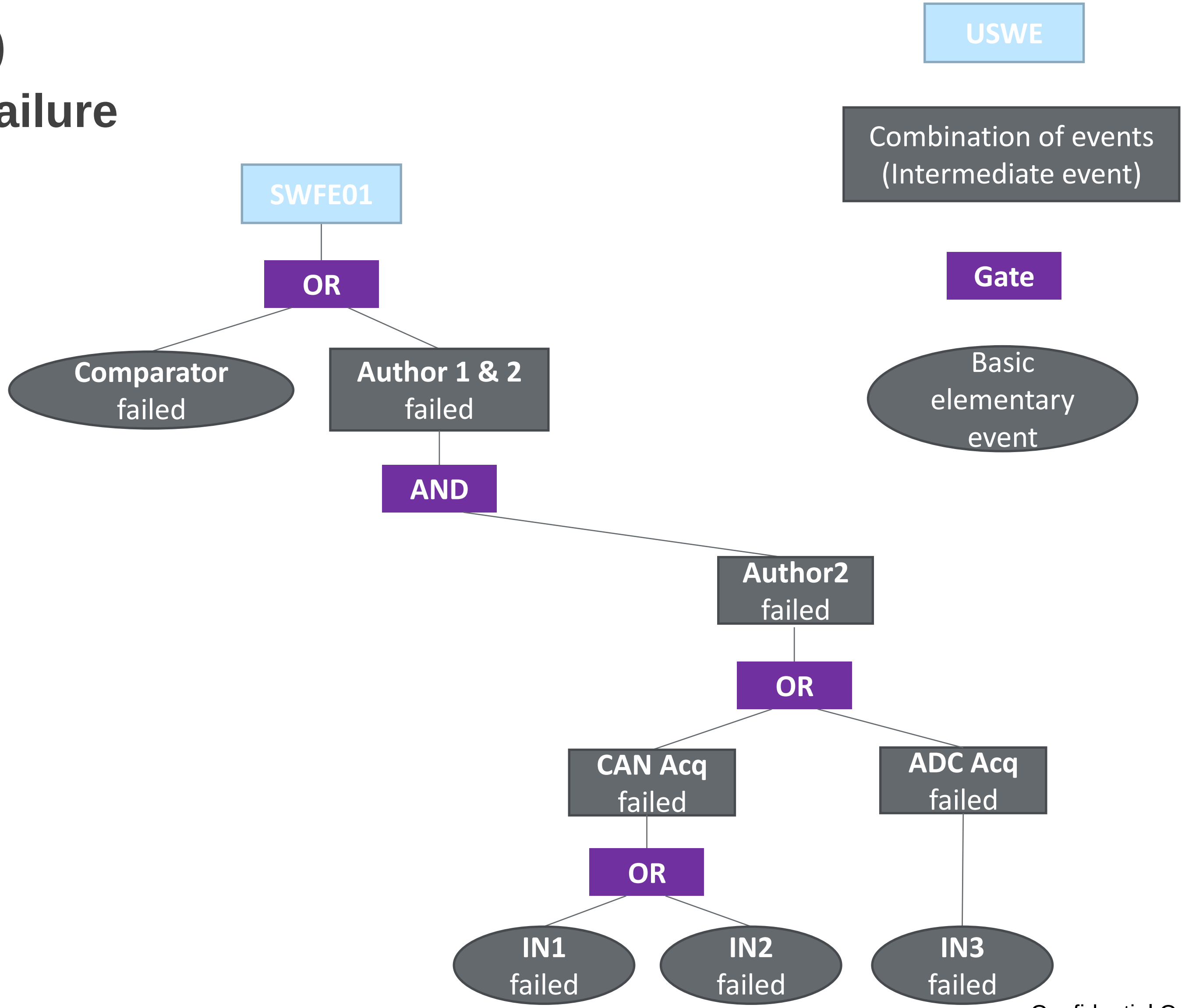




Safety analyses – FTA – Example

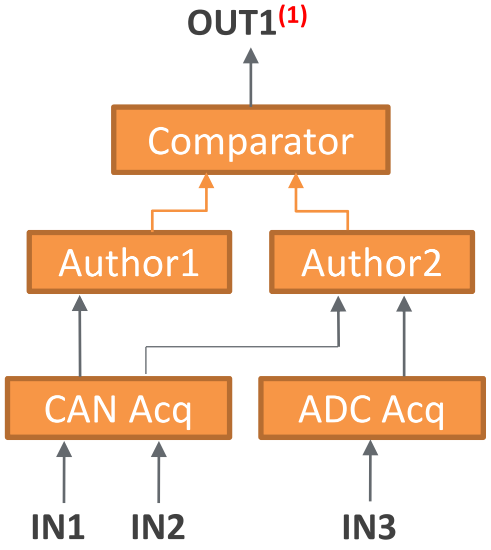
- **3.1.** Fault Tree Analysis (FTA)
  - **Step 3.1.3 – Application of failure modes & fault tree building**

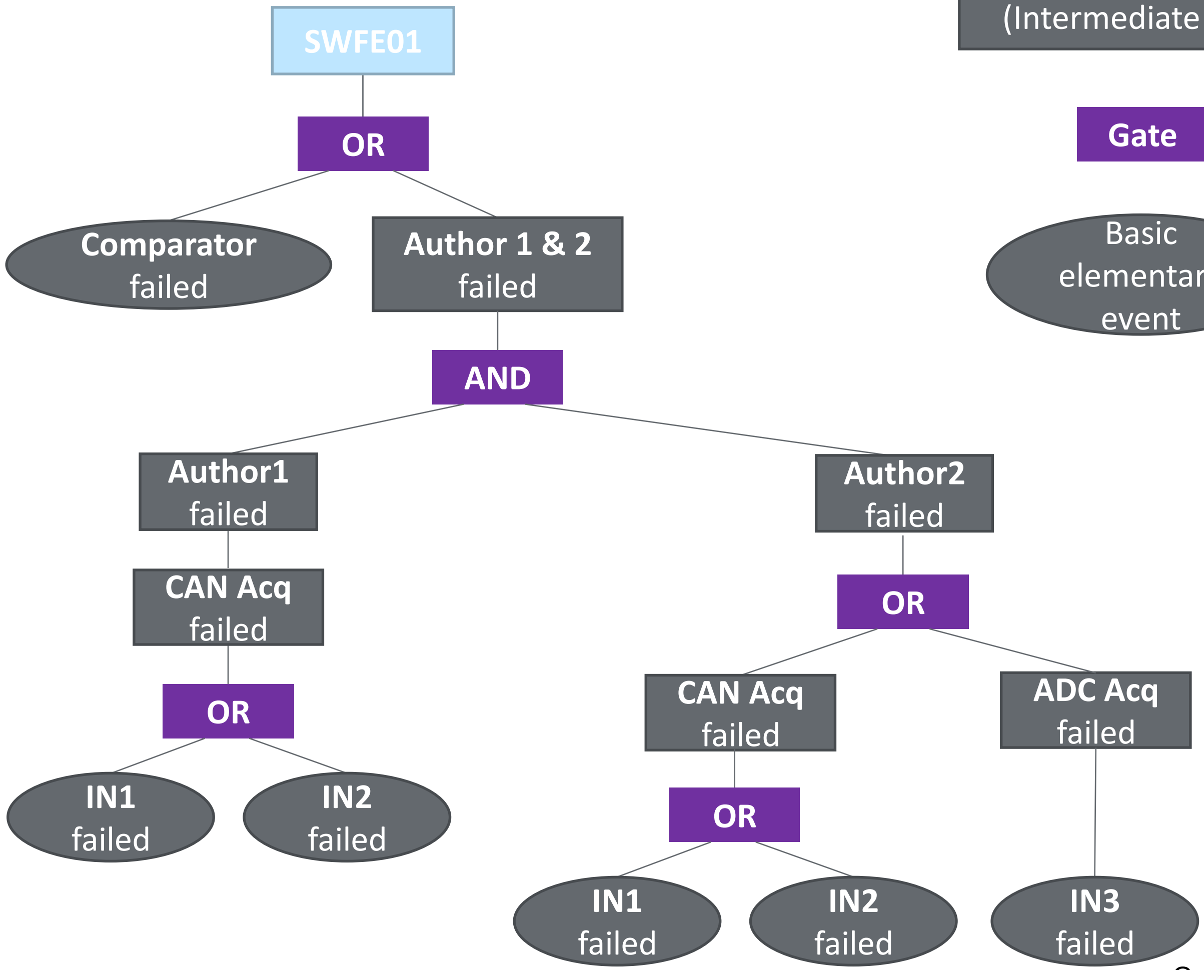




Safety analyses – FTA – Example

- **3.1.** Fault Tree Analysis (FTA)
  - **Step 3.1.3 – Application of failure modes & fault tree building**





USWE

Combination of events (Intermediate event)

Gate

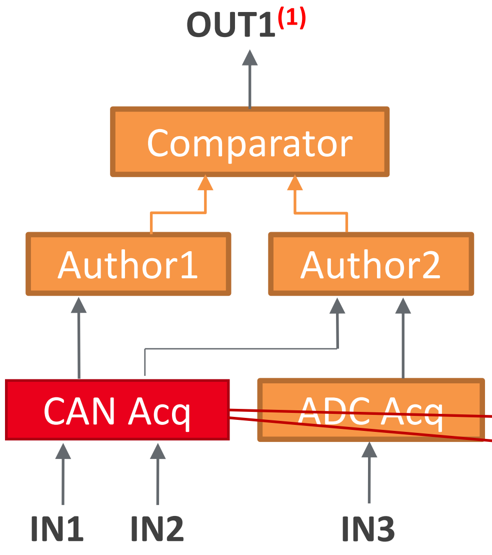
Basic elementary event

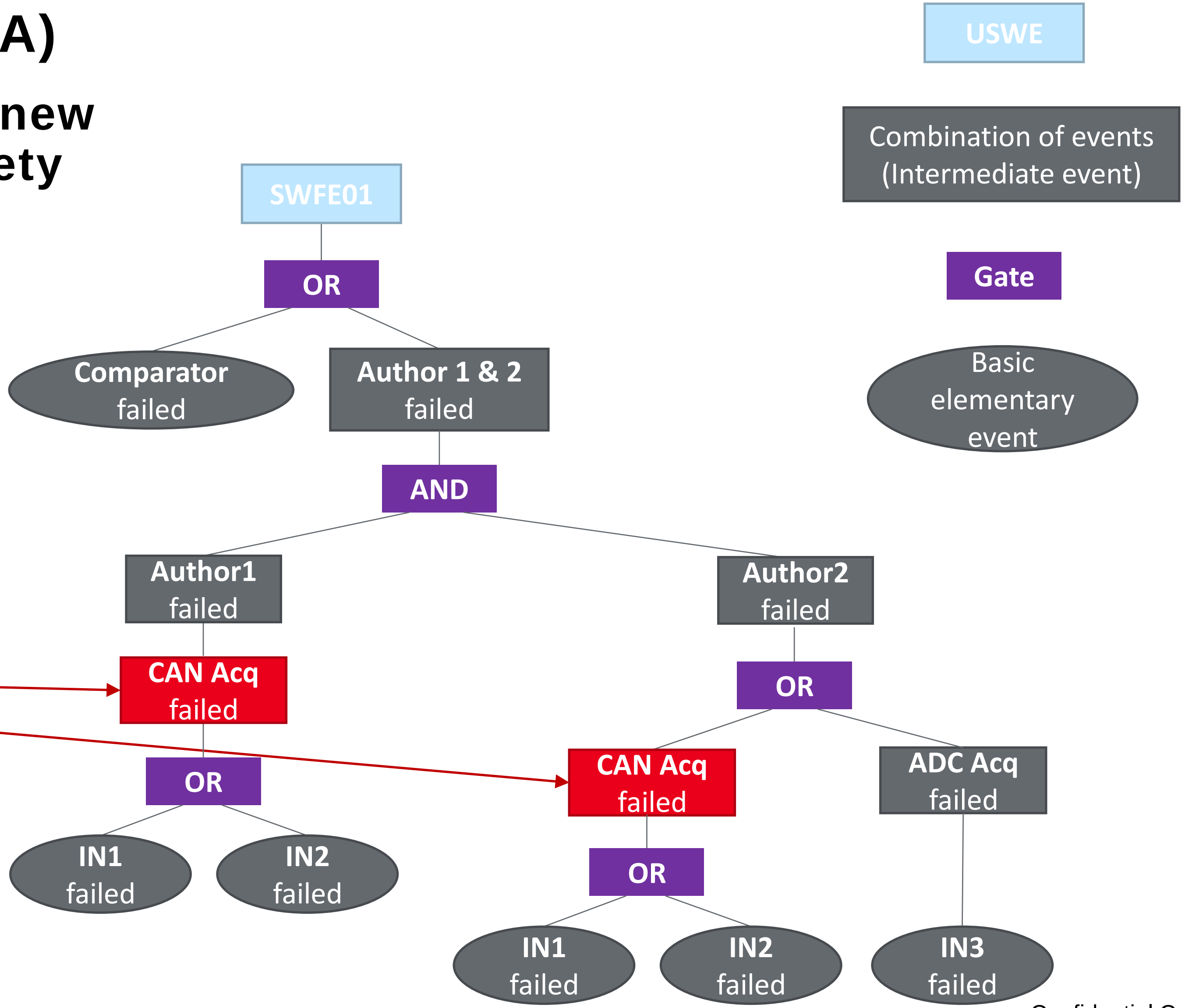
RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

3.1. Fault Tree Analysis (FTA)

Step 3.1.4 – Identification of new Safety requirements and Safety mechanisms

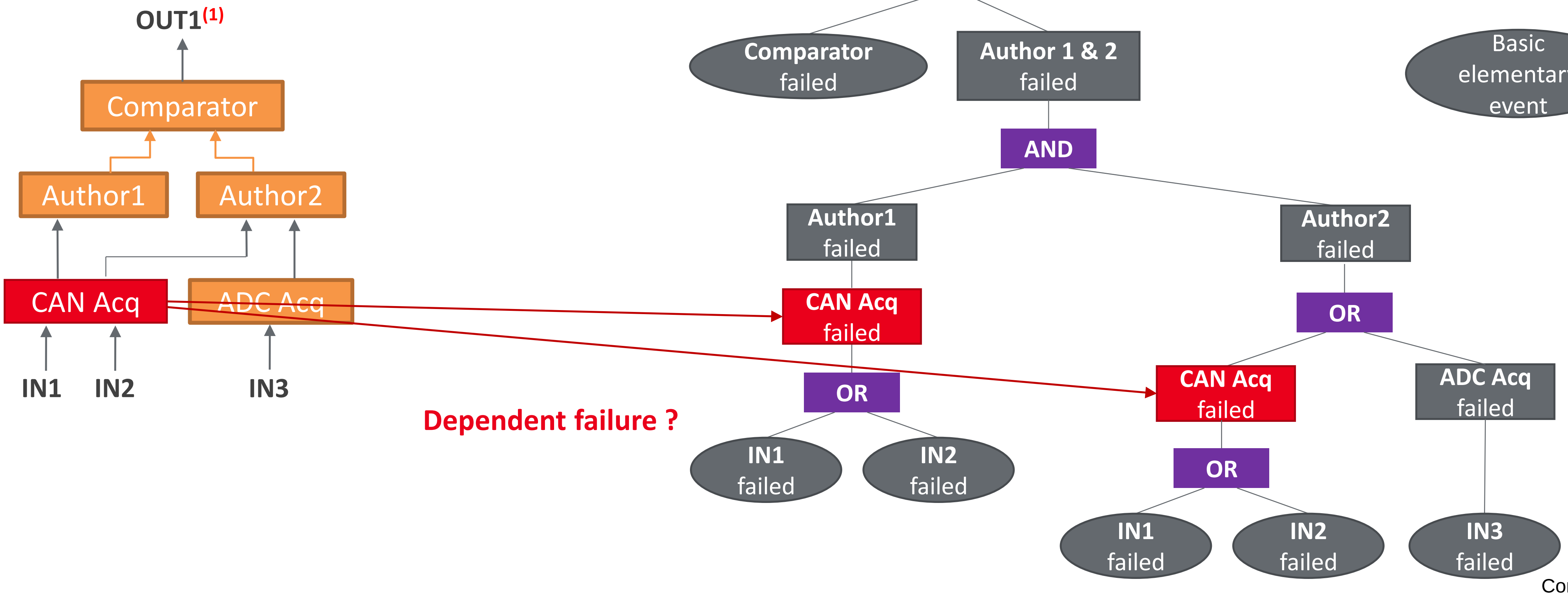




Safety analyses – FTA – Example

▶ 3.1. Fault Tree Analysis (FTA)

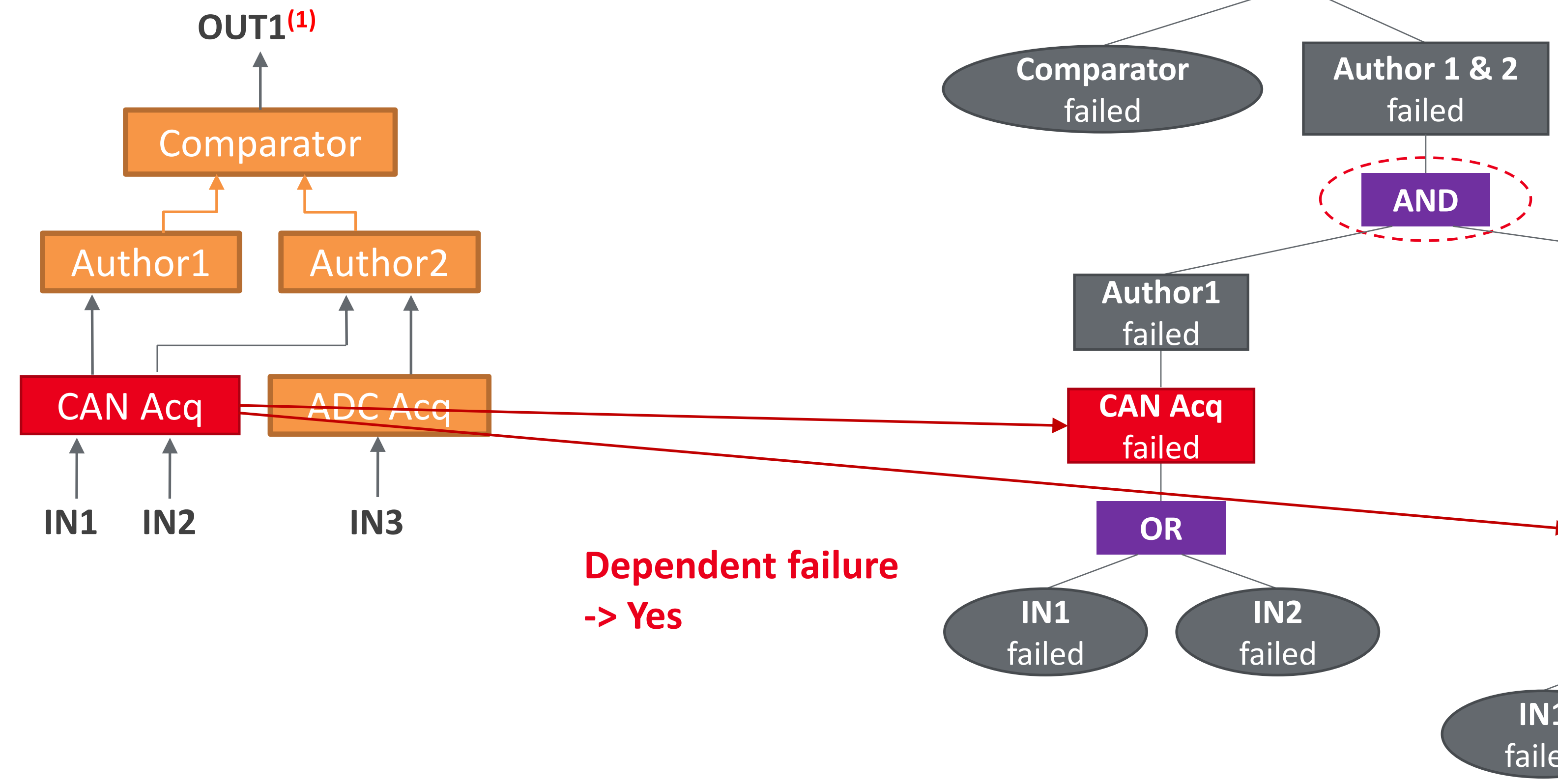
Step 3.1.4 – Identification of new Safety requirements and Safety mechanisms



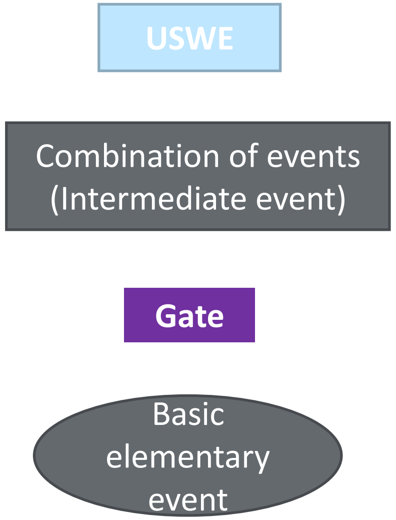
Safety analyses – FTA – Example

▶ 3.1. Fault Tree Analysis (FTA)

Step 3.1.4 – Identification of new Safety requirements and Safety mechanisms



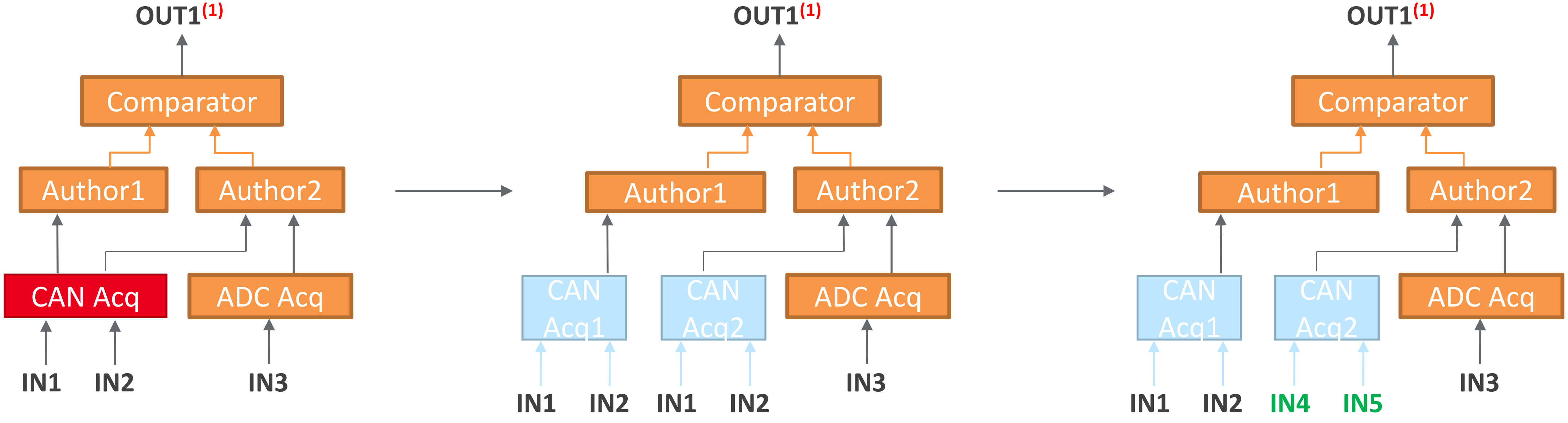
Dependent failure -> Yes



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FTA – Example

- **3.1.** Fault Tree Analysis (FTA)
  - **Step 3.1.4 – Identification of new Safety requirements and Safety mechanisms**



- Redundancy on **CAN Acquisition**
- Dependent failure on **IN1 & IN2**

- Redundancy on **CAN Acquisition**
- + Redundancy on **Inputs**

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety analyses – FMEA (1/4)

Step 3.2. Failure Modes & Effects Analysis (FMEA)

Definition

Analytical method used at the step of Software development

Inductive method: starts from known causes and identify possible effects

Bottom-up / Single failure / Worst case approach

FMEA asks « What is the effect if this component operates incorrectly ? »

Hypothesis

Existence of Hardware and Software failures

Principle

Propagate failures through the software in order to identify potential occurrence of USWEs

Safety analyses – FMEA (2/4)

Step 3.2. Failure Modes & Effects Analysis (FMEA)

- Example of failure modes

Failure mode	Meaning
Software Component/Unit not executed	All the Software Component/Unit outputs are erroneous because the Software Component/Unit is not executed Example: Software Component/Unit is not executed, all of the Software Component/Unit outputs are erroneous (random or initial values) It can be applied to SW Component/Unit level
Timing failure on run of Software Component/Unit (too late/early)	All Software Component/Unit outputs are erroneous because the Software Component/Unit is not executed at the correct time (too early or too late). Example: 1) Software Component/Unit logic is executed too early, all of the Software Component/Unit outputs are erroneous (data updated too early with out-of-date inputs) 2) Software Component/Unit logic is executed too late, all of the Software Component/Unit outputs are erroneous (data updated too late: outputs are out-of-date) It can be applied to SW Component/Unit level

Safety analyses – FMEA (3/4)

Step 3.2. Failure Modes & Effects Analysis (FMEA)

- Example of failure modes

Failure mode	Meaning
Software Component/Unit logic failure	All the Software Component/Unit outputs are erroneous because the Software Component/Unit internal logic is wrong Example: Software Component/Unit logic is failed, all of the Software Component/Unit outputs are erroneous (valid values but logic come from an other context) It can be applied to SW Component/Unit level
Erroneous value of a Software Component/Unit Data	One Data of a Software Component/Unit is erroneous. The Data value can be: - More/Less/Wrong enumerator - In range/Out of range - Permanently/Untimely Example: 1) Data is more, in SW range, permanently cause of a wrong unit (local one instead of international one) 2) Data is less, out of range, untimely cause of hardware disruption It can be applied to SW inputs or SW Component/Unit outputs

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

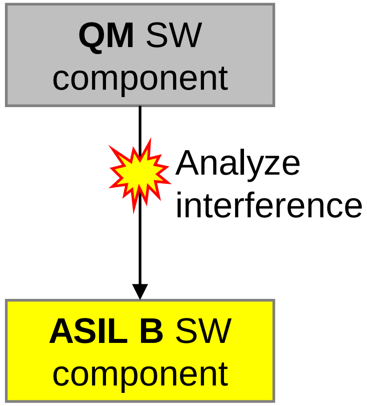
Safety-oriented analyses – FFI (1/4)

3.5. Freedom From Interference analysis (FFI)

Definition

Absence of cascading failures between two or more elements of different ASIL that could lead to the violation of a safety requirement

Particular case of DFA, to be applied when SW coexistence of different ASIL SW component is existing



RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI (2/4)

3.5. Freedom From Interference analysis (FFI)

Principle

Identify and analyze interferences between SW elements of different ASIL, to mitigate cascading failure

Identify sharing interfaces with any other component having different ASIL

Identify the type of potential risk

- *Spatial: Access of shared resources (safety data), memory partitions…*
- *Temporal: Interruption policy, scheduling policy…*
- *Communication: Boundary and dynamic consistency check…*

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI (4/4)

3.5. Freedom From Interference analysis (FFI)

Main steps

3.5.1. Identify interaction between components with different ASIL

3.5.2. Based on Safety analyses, identify potential dependent failure

3.5.3. Determine plausibility of failure

3.5.4. Identify:

Operating situations/modes applicable to dependent failure

Cause of the failure

Impact of the failure

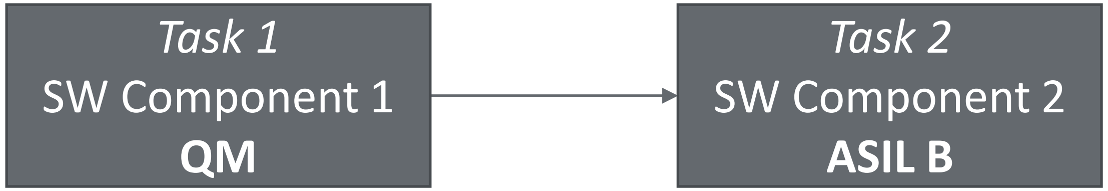
3.5.5. Suggest methods of resolution (safety mechanism or measure)

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI – Example

3.5. Freedom From Interference analysis (FFI)

- To consider the example for FFI application:



Hypotheses:

Spatial: SWC1 and SWC2 use the same memory space

Temporal: No timing management, SWC2 will be executed when SWC1 execution is ended

Communication: SWC1 sends data to SWC2 (no protection implemented)

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI – Example

3.5. Freedom From Interference analysis (FFI)

Step 3.5.2 – Identification of potential dependent failure

Spatial interaction, temporal interaction and communication

Potential Dependent Failure	Safety Analyses Reference	Plausibility of failure	Operating Situations\Modes for failure	Cause of failure	Impact of failure	Suggested Resolution
Spatial interaction	-					
Temporal interaction	-					
Communication	-					

This table is integrated in DFA/FFI template (sheet “Dependent Failure Analysis”)

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI – Example

3.5. Freedom From Interference analysis (FFI)

Step 3.5.3 – Plausibility of failure

Spatial interaction, temporal interaction and communication

Potential Dependent Failure	Safety Analyses Reference	Plausibility of failure	Operating Situations\Modes for failure	Cause of failure	Impact of failure	Suggested Resolution
Spatial interaction	-	Same memory space between SWC1/2				
Temporal interaction	-	No timing management				
Communication	-	Communication between QM and ASIL SWC				

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI – Example

3.5. Freedom From Interference analysis (FFI)

Step 3.5.4 – Operating situations, cause of failure and impact

Spatial interaction, temporal interaction and communication

Potential Dependent Failure	Safety Analyses Reference	Plausibility of failure	Operating Situations\Modes for failure	Cause of failure	Impact of failure	Suggested Resolution
Spatial interaction	-	Same memory space between SWC1/2	HW failure occurs on memory, leading to SWC1 failure. Worst case: Failure propagated in SWC2	No spatial independence	SWC2 failure	
Temporal interaction	-	No timing management	Task1 is delayed, leading to Task2 delay. Violation of timing constraints	No temporal independence	SWC2 delayed	
Communication	-	Communication between QM and ASIL SWC	Data transmitted to SWC2 from SWC1 is erroneous, leading to SWC2 failure	Data produced by SWC1 is consumed by SWC2	SWC2 failure	

RENAULT-NISSAN-MITSUBISHI RESTRICTED B

Safety-oriented analyses – FFI – Example

3.5. Freedom From Interference analysis (FFI)

Step 3.5.5 – Suggested resolution

Spatial interaction, temporal interaction and communication

Potential Dependent Failure	Safety Analyses Reference	Plausibility of failure	Operating Situations\Modes for failure	Cause of failure	Impact of failure	Suggested Resolution
Spatial interaction	-	Same memory space between SWC1/2	HW failure occurs on memory, leading to SWC1 failure. Worst case: Failure propagated in SWC2	No spatial independence	SWC2 failure	Safety mechanism: To implement MMU, allowing each SWC to be executed in its own virtual memory space, supported by hardware protection
Temporal interaction	-	No timing management	Task1 is delayed, leading to Task2 delay. Violation of timing constraints	No temporal independence	SWC2 delayed	Safety mechanism: To implement deterministic scheduling method, taking into account priority, FTTI and including Watchdog (slides 34 to 36)
Communication	-	Communication between QM and ASIL SWC	Data transmitted to SWC2 from SWC1 is erroneous, leading to SWC2 failure	Data produced by SWC1 is consumed by SWC2	SWC2 failure	Safety mechanism: To implement consistency checks (slide 33) Safety measure: To perform FMEA between SWC1 and SWC2 to confirm this approach

Thank you for your attention

QUESTIONS?