RESILTECH

“A methodology to ensure safety (certification) of complex software in safety critical automotive systems"

Francesco.Brancati@ResilTech.com

Scope of the Talk

This talk presents an approach to

Safety Analysis

Dependent Failure Analysis

according to the automotive standard ISO26262 for complex software with focus on the following features

- Embedded SW
- Library/component based
  - Suitable for SEooC (Safety Element out of Context) integration
- Multi-criticality software systems

RESILTECH

1. Short Company Introduction
2. SW Safety Analysis and DFA in Automotive
3. ResilTech Methodology
4. Feedback from application and future directions

RESILTECH

1. **Short Company Introduction**
2. SW Safety Analysis and DFA in Automotive
3. ResilTech Methodology
4. Feedback from application and future directions

ResilTech s.r.l.

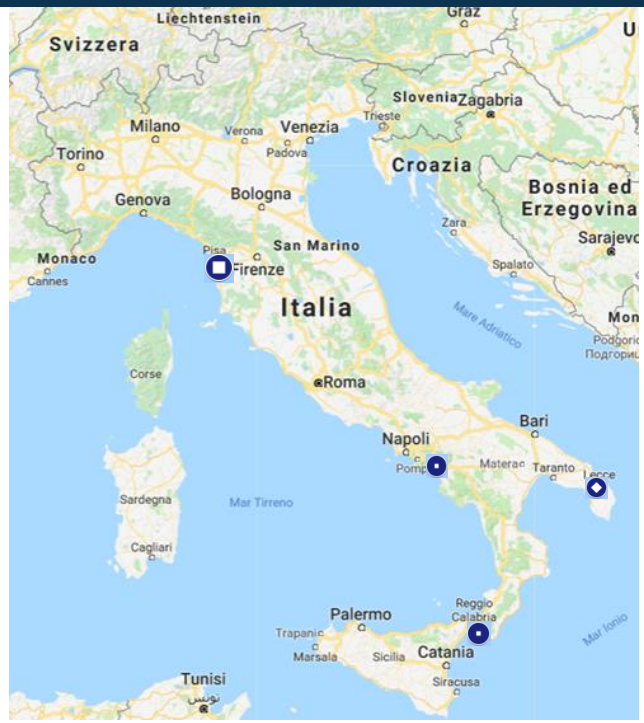
Company founded in late 2007 by

- **specialists in the industrial field** of Verification and Validation (V&V) of critical systems and
- **university researchers** expert in Resilient Computing
- 2012-2016 **Spin-OFF** of the University of Florence

UNIVERSITÀ DEGLI STUDI FIRENZE
SPIN-OFF APPROVATO

HeadQuarter

Piazza Nilde Iotti, 25
56025 - Pontedera (PI), Italy



Mission

To provide engineering consulting and design services to companies and public bodies mainly for, but not limited to, the field of resilient systems and infrastructures

Member of CISQ Federation
RINA
ISO 9001:2008
Certified Quality System

ISO 9001:2015 certified

Branch Office 1
Via Colonnello Archimede
Costadura 2C, 73100 - Lecce, Italy

Branch Office 2
Via La Boccetta 7,
89134 – Reggio Calabria, Italy

Branch Office 3
Salerno, just opened in Dec. 2018

Core services

Resilient Systems Design

Architecting and Implementation of Dependable Systems

Verification & Validation & Safety

Full V&V&S Cycles activities according to latest standards of SW intensive system

Support to Certification Bodies

Cooperation with National & International Certification Agencies

Cyber security

Security solution design and assessment

Advanced Training

On Safety Standards, system modeling, Life Cycle Cost Analysis, Verification and Validation

Creating Innovation

Strong Reseach Attitude:

Ongoing Projects:

SISTER - POR Toscana 2014
STORM - H2020-DRS11-2015
PROTECT ID– PON – MISE 2016
Net2DG– H2020-LCE-2017
YACHT4.0– POR Toscana 2017
Good4you– Innonetwork (Puglia)

Starting Projects:

MAIA– PON-MIUR-2018
ADVANCE– H2020-RISE-2018

Main research topics

- **Cost Effective V&V Methodologies and tools**
- **Integration of AI components in Safety Critical System**
- Online Failure/Intrusion prediction and Detection
- Monitoring and Analysis (ML& AI)
- Continuous Transparent Biometric authentication
- **Safety Platforms SW/system for Emdedded System**
- **Intelligent and smart monitoring of SoCs**
- Methods for Resilient time distribution

Patents:

METHOD AND APPARATUS FOR A RESILIENT SIGNALING OF TIME
Italian Office N. 102015000072477

Standardization Activities in Safety

ISO TC22/ SC32/WG8
for ISO26262 ("Road vehicles - Functional safety")
for ISO21448 SOTIF ("Safety of the Intended Functionality")

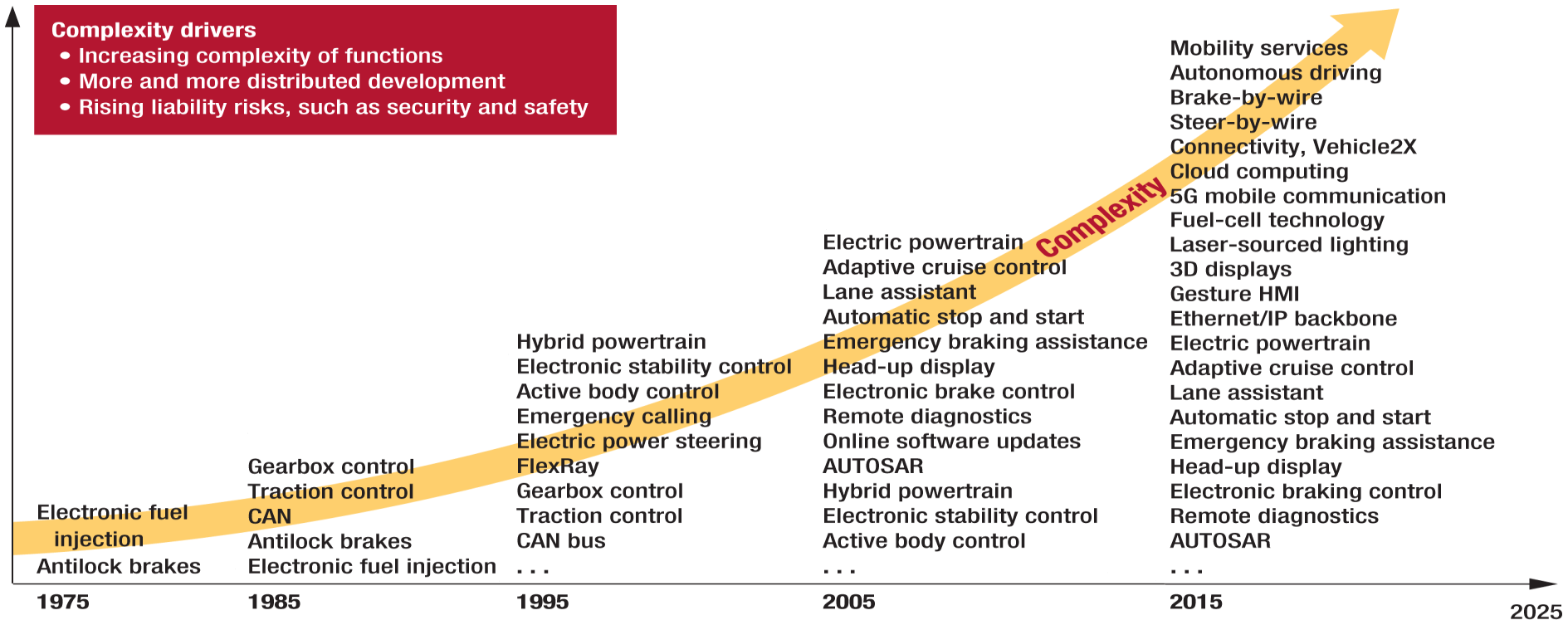
OpenGL SC 2.0 is a safety critical subset of the Open Graphics Library for safety critical markets

- streamlined APIs can significantly reduce certification costs
- includes avionics and automotive displays
- OpenGL SC 2.0 Full Specification: April 2016.
  - https://www.khronos.org/registry/OpenGL/specs/sc/sc_spec_2.0.pdf

RESILTECH

1. Short Company Introduction
2. **SW Safety Analysis and DFA in Automotive**
3. ResilTech Methodology
4. Feedback from application and future directions

Automotive market trend



- "migration" of technology (and SW) from non safety relevant application.
- increasing need of having components with some degree of built-in error-detection capabilities
  - To ease the integration and acceptance of SW non developed with full compliance to safety lifecycle (e.g. library porting from consumer application).

Normative requirements - intro

ISO 26262
Road Vehicles - Fuctional Safety

ISO26262 supports such industrial need asking to enhance the safety architecture of the SW even at component level (**SEooC concept**) when this applies.

Part 6, **7.4.10** till **7.4.13**

Additional info on part 9, sec 7,8 and annexC, but not sw specific

- **Safety Analysis** (SA)
- **Dependent Failure Analysis** (DFA)

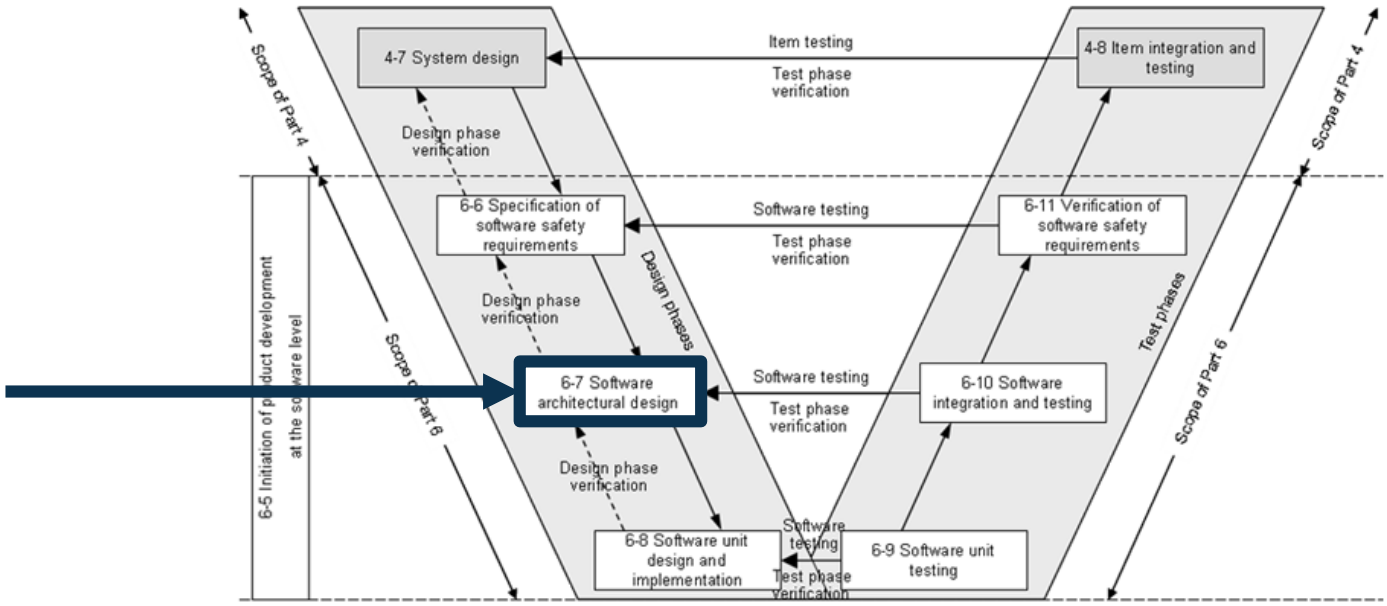
main goals:

1. to **support the safety concept verification** when is based on the independence/diversity of software functions/components
2. to **verify the coexistence criteria** among the software components
3. to **support the specification of safety mechanism** at software architecture level, in order to mitigate SW failure identified in the analysis

Main techniques: **SW-FMEA** (3) and **DFA** (1-2)

Such requirements were already present in Edition 1 (2011), but lack of experience in application push the committee to provide a **full informative annex (E) to guide industry** in the second edition (2018)

Normative requirements - lifecycle



Main aim within the lifecycle is:

to support the **specification of safety mechanism at software architecture level**.

Output of the Activity:

- **modified architecture** to accommodate error detection and error recovery mechanisms (and proper reactions of the SW in line with original safety concept).

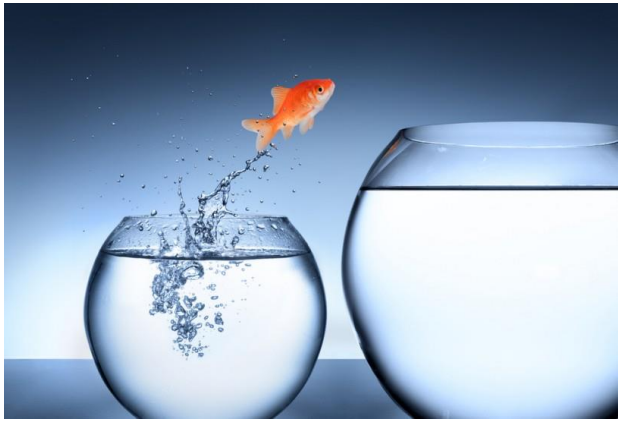
And/or…

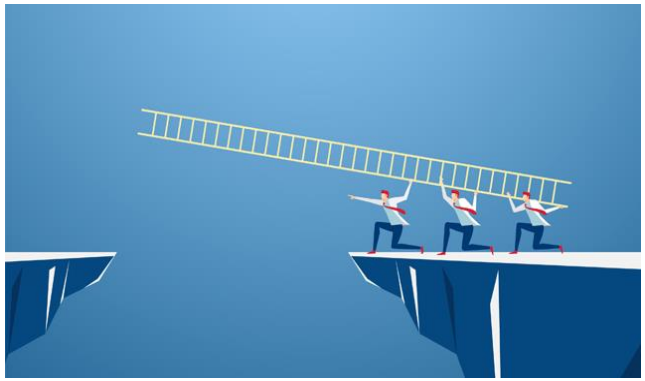
- Evidence that existing **architecture is completely or partially fine as it is**.
- Additional **Assumptions of Use** for system level

Challenges and Opportunities

Challenges:

1. The inclusion of mechanism as deadline or control flow monitoring in SW architectures is not new in the safety industry, but this is mostly done based on experience **without a complete formal modelling of the architecture and of the SW faults.**
2. This inclusion is **often done when dealing with the entire system architecture** while it may be beneficial also if applied to parts of it (e.g. OS+middleware or complex libraries).
3. The new annex in **ISO26262** provides some guidance (example-based) but still **delegates the definition of a clear methodology** in line with the aim of an informative text.





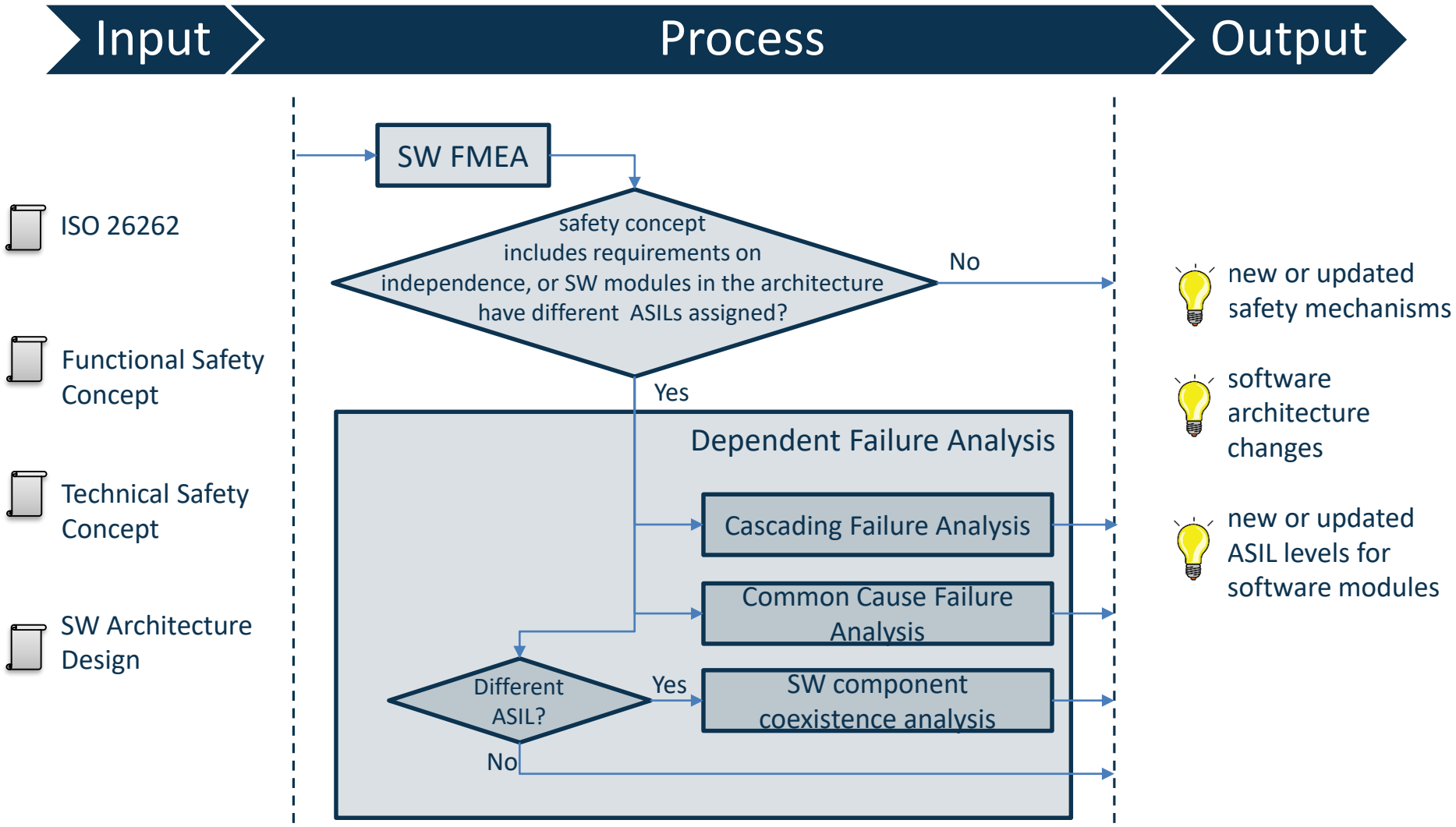
Opportunities:

1. Proposal of a **clear methodology** to perform such activities.
2. This is **important particularly for SW** as fault modelling and FMEA approaches are more understood and applied in the industry at HW and system level rather than SW.
3. In addition an important aspect, generally not fully considered when defining SW Safety Mechanisms, is to consider **how the effectiveness of V&V activities affect the "likelihood" of some SW faults**.
   - Here the point is trade-off **architectural changes versus fault-removal techniques**.

RESILTECH

1. Short Company Introduction
2. SW Safety Analysis and DFA in Automotive
3. **ResilTech Methodology**
4. Feedback from application and future directions

ResilTech Methodology – Overall View



ResilTech Methodology – Input 1/4

Input

Data from **ISO26262-6 Annex D-Freedom from interference between software elements**

ISO 26262

Functional Safety Concept

Technical Safety Concept

SW Architecture Design

Constitute the reference set of guidewords to define **failure modes** in Safety Analysis and DFA.

Timing and execution

- blocking of execution
- deadlocks
- Livelocks
- incorrect allocation of execution time
- incorrect synchronization between software elements.

Memory

- corruption of content
- inconsistent data (e.g. due to update during data fetch)
- stack overflow or underflow
- read or write access to memory allocated to another software element

Exchange of information

- repetition of information
- loss of information
- delay of information
- insertion of information
- masquerade or incorrect addressing of information
- incorrect sequence of information
- corruption of information
- asymmetric information sent from a sender to multiple receivers
- information from a sender received by only a subset of the receivers
- blocking access to a communication

ResilTech Methodology – Input 2/4

Input

ISO 26262

Functional Safety Concept

Technical Safety Concept

SW Architecture Design

Safety Mechanisms from **ISO26262-6 «Table 4 — Mechanisms for error detection at the software architectural level**

Constitute the reference set to select the “intended safety mechanisms” in SW FMEA and DFA. It can be refined, depending on the characteristics of the project.

Mechanisms		ASIL			
		A	B	C	D
1a	Range checks of input and output data	++	++	++	++
1b	Plausibility check	+	+	+	++
1c	Detection of data errors	+	++	++	++
1d	Monitoring of program execution	o	+	++	++
1e	Temporal monitoring of program execution	o	+	++	++
1f	Diverse redundancy in the design	o	o	+	++
1g	Access permission control mechanisms	+	+	++	++

ResilTech Methodology – Input 3/4

Input

ISO 26262

Functional Safety Concept

Technical Safety Concept

SW Architecture Design

Error Handling from **ISO26262-6 "Table 5 — Mechanisms for error handling at the software architectural level"**

Constitute the reference set to select the "intended safety mechanisms and related error handling" in SW FMEA and DFA. It can be refined, depending on the characteristics of the project.

Mechanisms		ASIL			
		A	B	C	D
1a	Static recovery mechanism	+	+	+	+
1b	Graceful degradation	+	+	++	++
1c	Homogenous redundancy in the design	+	+	+	++
1d	Diverse redundancy in the design	o	o	+	++
1e	Correcting codes for data	+	+	+	+
1f	Inhibit access permission violations	+	+	++	++

ResilTech Methodology – Input 4/4

Input

ISO 26262

Functional Safety Concept

Technical Safety Concept

SW Architecture Design

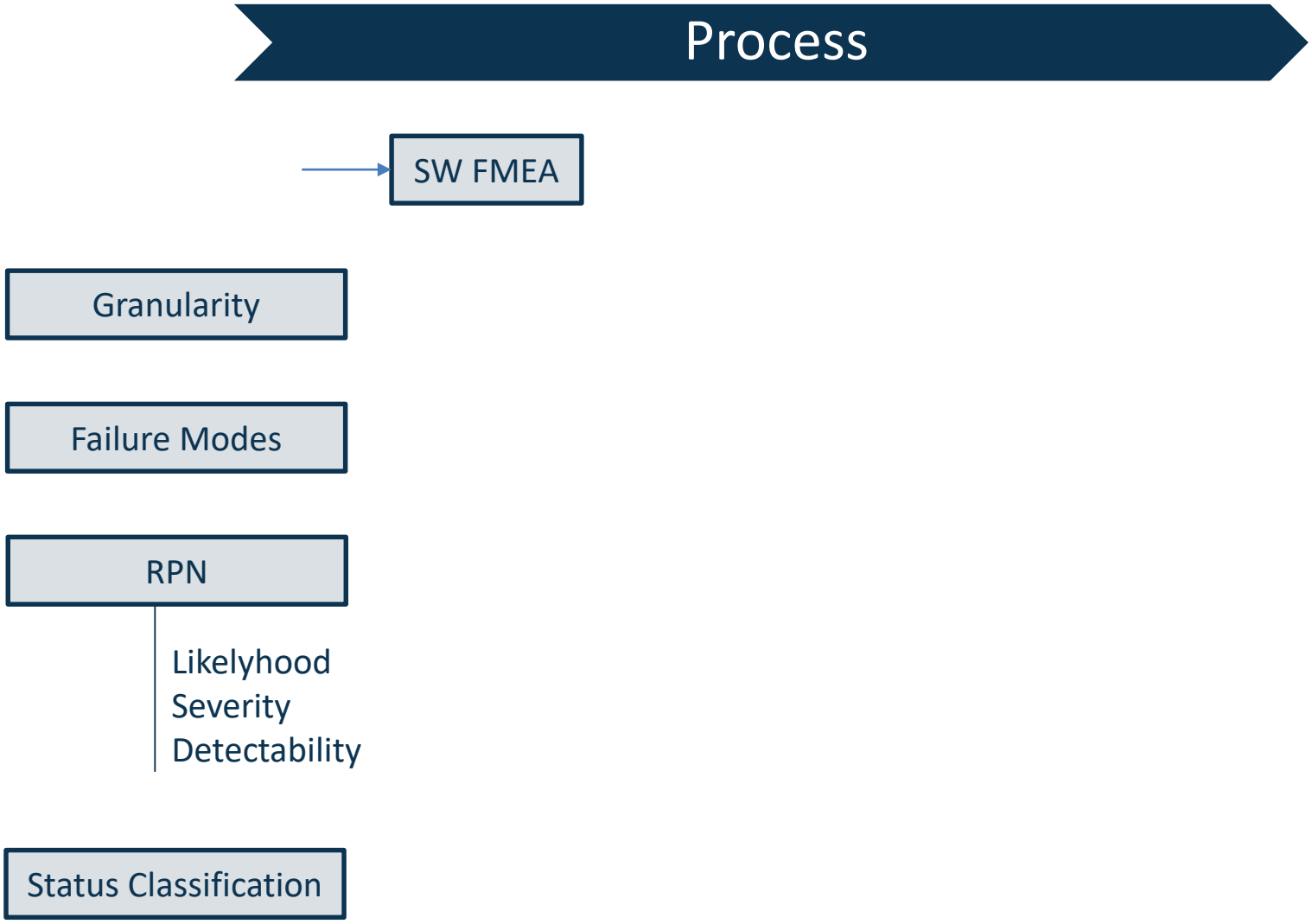
Functional safety concept (ISO26262-2): specification of the *functional safety requirements*, with associated information, their allocation to architectural *elements*, and their interaction necessary to achieve the *safety goals*

Technical safety concept (ISO26262-2): specification of the *technical safety requirements* and their *allocation* to *system elements* for implementation by the *system* design

Software Architecture Design:

- **(software) Architecture (ISO26262-2):** representation of the structure of the *item* or *systems* or *elements* that allows identification of building blocks, their boundaries and interfaces, and includes the allocation of requirements to hardware and software *elements*
- **Design (FP7 AMADEOS):** The process of defining an architecture, components, modules and interfaces of a system to satisfy specified requirement.

SW FMEA: Steps



SW FMEA: Granularity

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood
Severity
Detectability

Status Classification

- It depends on the target software, as general rule:
  - Mandatory:
    - system APIs and APIs of components at the high level design.
  - Recommended:
    - subcomponent (module) levels. The level of details reached for the SW FMEA analysis depends on the complexity of the component and the design principles.
    - At least:
      - internal resource usage
      - internal IPC
      - timing
      - local scheduling and priority

SW FMEA: Failure Modes

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood
Severity
Detectability

Status Classification

- Guidewords from ISO26262-6 Annex D
- component failure modes, which **can (should) be refined depending on the project:**
  - Fails to execute or halts
  - Executes incompletely or concludes abnormally
  - Output incorrect, missing or late (includes possibility of returning no error or wrong error codes)
  - Incorrect timing – too early, too late, slow, etc..
  - Incorrect internal state change
  - Incorrect internal IPC
  - Incorrect local scheduling and priority
  - Incorrect internal resource usage (virtual/physical resources, computational power)
  - Erroneous data management and data corruption
  - Wrong calibration data
  - **Others…**

SW FMEA: Likelyhood

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood

Severity

Detectability

Status Classification

- In case the target system is an evolution of an existing one:
  - History from bug reports
  - Code metrics as cyclomatic number, code smells detection (if applicable)
- Otherwise (no SW reuse), we can use design metrics as:
  - design complexity
  - configuration complexity (if applicable)
  - hardware, OS, libraries dependencies
  - Number of global values, of function/system status, dimension of data structures, shared resources
- These values could contribute to the definition of a concept of classes of likelihood of failures

SW FMEA: Likelyhood (example)

Process

SW FMEA

Granularity

Failure Modes

RPN

- Likelyhood
- Severity
- Detectability

Status Classification

Failure mode description for particular SW	SW component complexity parameters	Likelihood pre- V&V	V&V activity	V&V efficiency	Likelihood
Run time library exceeding time slot	Cyclomatic number>=Y & LOC >=X	High	WCET estimation	Low	High
Run time library exceeding time slot	Cyclomatic number<Y & LOC<X	Low	WCET estimation	Low	Low
Run time library exceeding time slot	Cyclomatic number>=Y & LOC >=X	High	Complete set of performance / timing test	High	Low

SW FMEA: Severity

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood

Severity

Detectability

Status Classification

- Severity of failures effects should be evaluated with respect to the design specification, the safety concept and the safety goal
- It is difficult to provide objective ways to measure severity.
- We Just distinguish in two classes: YES or NO
  - if the failure leads to the violation of a safety goal or a safety requirement it is classified: Severity= YES
  - otherwise it is classified: Severity= NO

SW FMEA: Detectability

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood
Severity
Detectability

Status Classification

The Failures detectability of mitigation items is the estimated ability of the selected countermeasures to detect and tolerate a given component failure

We propose three classes on such coverage:

- **High**
  - For example, a checksum coverage that is adequate for ASIL B.
- **Medium**
  - For example, a range check, which is not able to detect approximation errors.
- **Low**
  - **LOW shall be selected whenever evidence for High or Medium coverage cannot be provided**

SW FMEA: Risk Probability Number

Process

SW FMEA

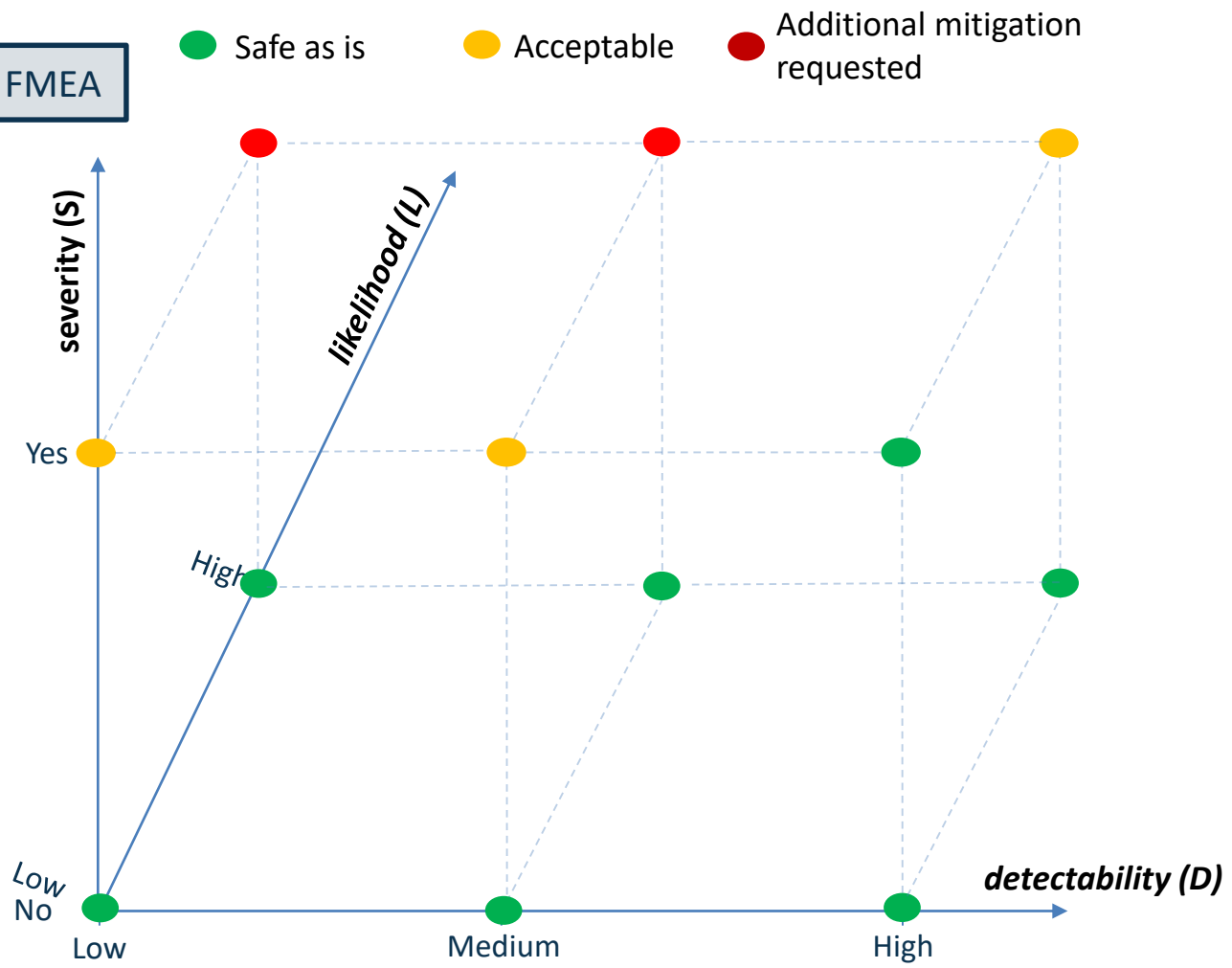
Granularity

Failure Modes

RPN

Likelyhood
Severity
Detectability

Status Classification



SW FMEA: Status Classification

Process

SW FMEA

Granularity

Failure Modes

RPN

Likelyhood
Severity
Detectability

Status Classification

***Open**: the failure is not yet managed.*

***Mitigated**: a safety mechanism was previously implemented in order to mitigate the failure.*

***Ignored**: the impact of the failure is, to an acceptable extent, a minor one. It is not necessary to mitigate it. (severity=NO).*

***Closed**: a new solution is introduced to mitigate the failure*

***Transferred**: the failure is not mitigated; its propagation will be mitigated at a later phase. This generally means that new assumptions of use, modification of architectural design, or additional V&V activities are introduced and matched to mitigate this failure.*

SW FMEA: resulting table (examples)

ID	SW Component ID	Component Failure Mode	Component Failure Description	Likelihood	Effect description	Severity class	Existing mitigation and impact	Detectability	RPN	Intended Mitigations	Status (with RPN post mitigation)
CPP-2	C/C++ Runtime library	Crash	No computations performed/application crash. It happens when the result has NULL input iterators.	High	Application crashes	Yes	Application fails to contact the safety monitor. Safety monitor reports to MCU	High	Acceptable	AoU - Applications must report their state to the Health Monitor	Transferred
CPP-4	C/C++ Runtime library	Error in implementation of exception handling	Possible issues : * Stack not correctly unwound * Exception not thrown, wrong exception thrown * Memory not available for exception handling	Low	Wrong execution flow, memory leaks	Yes	Full validation and code developed according to ISO 26262 part 6.	High	Acceptable		Mitigated
CPP-5	C/C++ Runtime library	Input not accepted	Dyanmic memory allocation fails. This can happen e.g. if dynamic memory fails within the RT	High	Computation not performed and error code returned	Yes	Error code returned to the application that can take corrective measures.	High	Acceptable	AoU - Applications must handle error status	Transferred
CPP-8	C/C++ Runtime library	IEEE exception	Executes incompletely and returns error code	High	Computation not performed and error code returned	Yes	Error code returned to the application that can take corrective measures.	High	Acceptable		Mitigated

ResilTech Methodology – Overall View

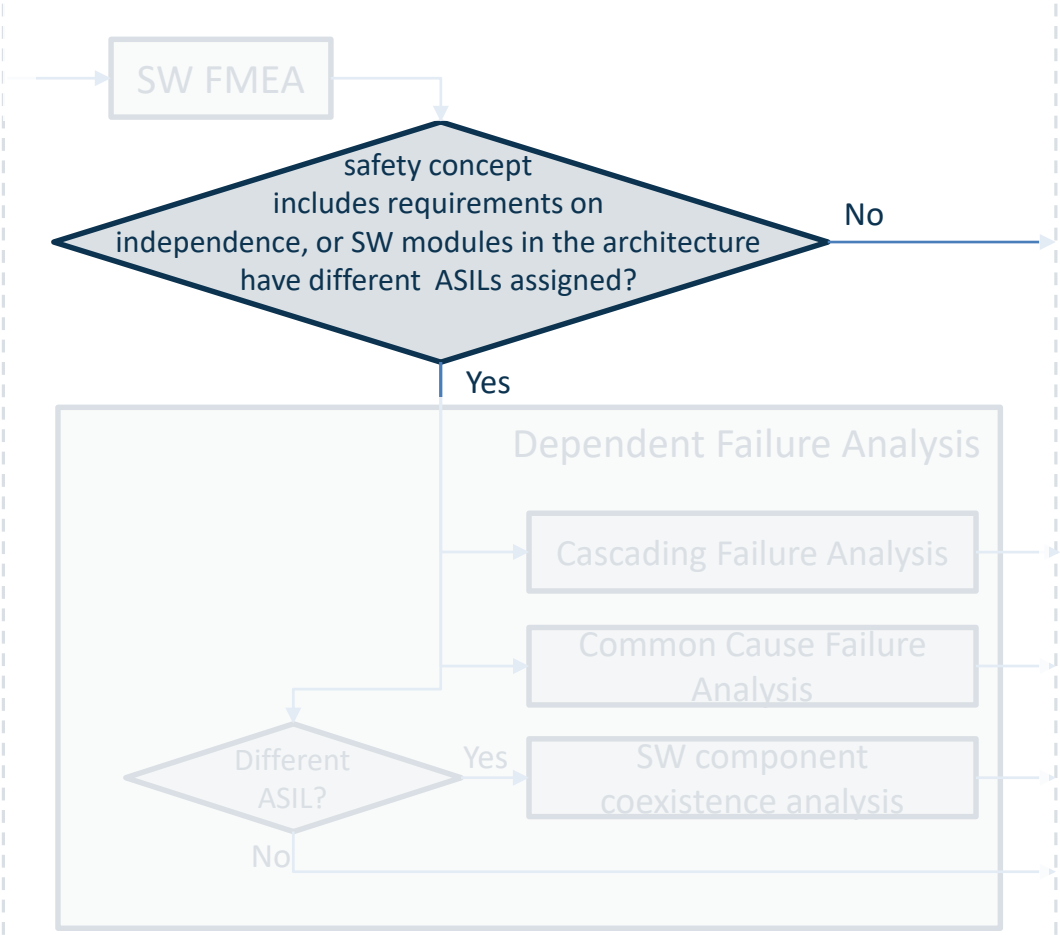
Input | Process | Output

ISO 26262

Functional Safety Concept

Technical Safety Concept

SW Architecture Design



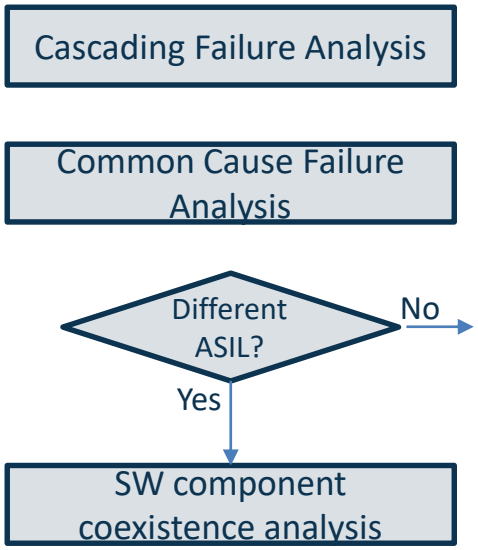
new or updated safety mechanisms

software architecture changes

new or updated ASIL levels for software modules

SW Dependent Failure Analysis: Introduction

Process



The goal is

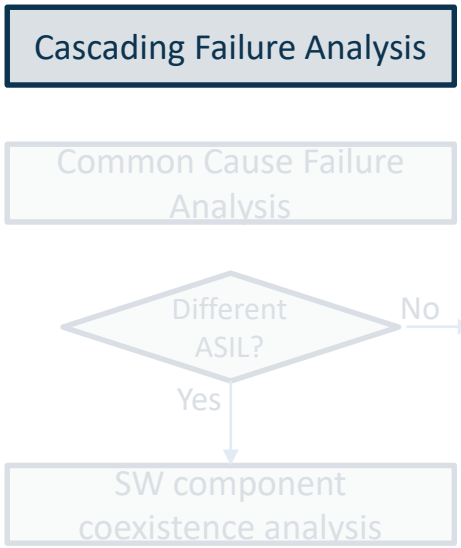
- to identify and analyze the possible **common cause and cascading failures** between supposedly independent software elements,
- to assess their risk of violating a safety goal (or derived safety requirements)
- to define new safety measures to mitigate such risk if necessary.

Steps:

- Software component independence analysis
  - cascading failures analysis
  - common cause failures analysis
- In case of sub-elements with different ASILs
  - Software component coexistence analysis

SW DFA: Cascading Failure Analysis

Process



It refers **exclusively to software** and it is organized in the following steps:

Step 1. A checklist to define failures that may propagate through a failure chain is identified. Each element is numbered with an ID.

Step 2. Identify couples of SW components to be checked for independence based on the requirements of independence (e.g., parallel elaboration with diverse algorithm). Each set is numbered with an ID.

Step 3. A guideword-based analysis is applied to each set, to understand the impact of such failures from a system-level point of view.

SW DFA: Cascading Failure Analysis: checklist

Process

Cascading Failure Analysis

Common Cause Failure Analysis

Different ASIL? No

Yes

SW component coexistence analysis

Step 1: checklist (to be refined and tailored for each project)

– Some Examples:

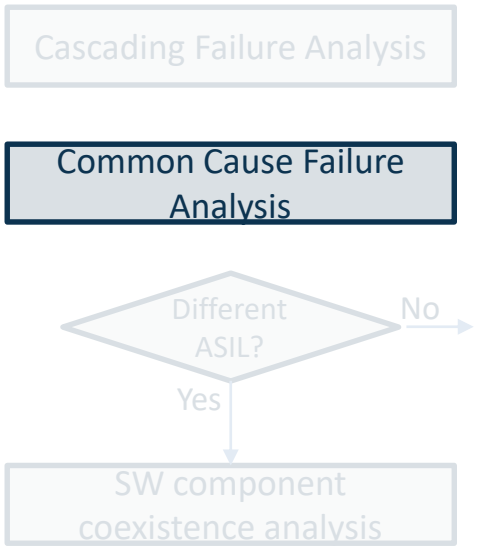
ID	Category	Element	Interpretation
Timing_1	Timing and execution	Block of Execution	Does a block of execution of the SW component impact the destination SW components?
Timing_3	Timing and execution	Deadlocks	Are there potential situations where the SW component experiments deadlocks? (e.g. locking mutexes, waiting for the return value of a function, etc.)
Timing_5	Timing and execution	Execution Time	Is the SW component taking too much time to execute? Or is it too fast? Or starts at a wrong instant?
Memory_1	Memory	Corruption of content	Check the possible propagation of corrupted data from the source to the destination SW components.
Memory_3	Memory	stack overflow/underflow	Check for possible stack overflow/underflow during memory usage
Information_1	Exchange of Information	Information repetition	Does a potential information repetition create a cascading failure in the destination SW components?
Information_2	Exchange of Information	Loss of data	Does a potential loss of data create a cascading failure in the destination SW components?
Information_6	Exchange of Information	Incorrect sequence	Does a potential incorrect sequence of information create a cascading failure in the destination SW components?
Information_7	Exchange of Information	Corruption	Does a potential corruption of information create a cascading failure in the destination SW components?

SW DFA: Cascading Failure Analysis: resulting table (example)

ID	ID of SW components under analysis	Cascading failure checklist	Failure Description	Likelihood	System Failure Mode (effect)	Severity Class	Existing mitigation and impact	Detectability	RPN	Intended Mitigations	Status (with RPN post mitigation)
CCF-5	(1 application using C/C++ Runtime, with certain requiremetns on independence)	Timing	A completion or failure indication is received too late - A job hasn't completed on time	HIGH	Both applications cannot proceed	YES	watchdog timer expired to indicate to the applications that the job hasn't completed	HIGH	Acceptable		Closed

SW DFA: Common Cause Failure Analysis

Process



It refers **exclusively to software** and it is organized in the following steps:

Step 1. A set of guidewords for events or root causes that may be cause of common failures of software elements is identified. Each keyword is numbered with an ID.

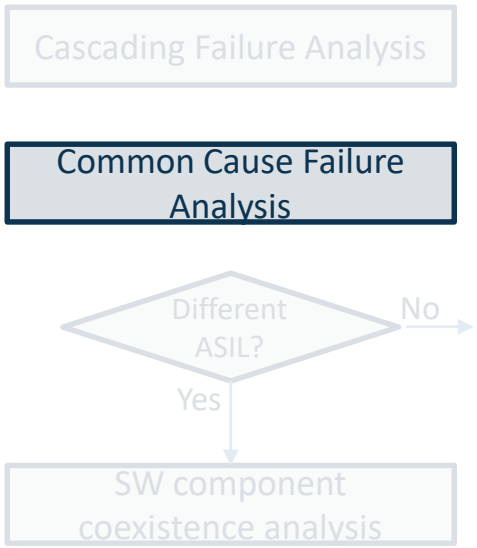
Step 2. Identify couples of SW components

- Typically, these are elements that:
  - Realize safety-critical functionalities through software diversity.
  - Are replicated software, running on the same hardware.
  - Implement redundant functionalities:
  - This item includes the redundancy of a safety mechanisms with respect to a target element.

Step 3. A guidewords-based analysis is applied to each set, to understand the impact of such failures from a system-level point of view.

SW DFA: Cascading Failure Analysis

Process



Step 1: checklist (to be refined and tailored for each project)

– Some Examples:

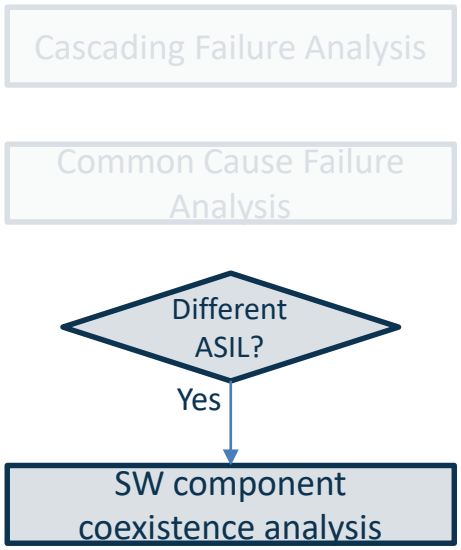
ID	Domain	Event or Root Cause	Interpretation
E1	Spatial	Misbehaviour of a shared resource or service	Any software element can act as a shared resource or service (remember that these are resource or service external to the elements sets that will be identified in Step 2). However, from experience, we recommend attention to: software libraries, drivers, services, files, algorithms, virtual communication channels, IPC mechanisms, signals, calibration data, data.
E2	Spatial	Unavailability of a shared resource or service	
E3	Temporal	Slow shared resource or service	

SW DFA: Common Cause Analysis: resulting table (example)

ID	ID of couple of components under analysis	Common cause failure checklist	Failure Description	Likelihood	System Failure Mode (effect)	Severity Class	Existing mitigation and impact	Detectability	RPN	Intended Mitigations	Status (with RPN post mitigation)
CCF-5	(2 applications using C/C++ Runtime, with certain requiremetns on independence)	Temporal	A completion or failure indication is received too late - A job hasn't completed on time	HIGH	Both applications cannot proceed	YES	watchdog timer expired to indicate to the applications that the job hasn't completed	HIGH	Acceptable		Closed

SW DFA: SW Component Coexistence Analysis

Process



- The goal is to check that lower ASIL component failures do not impact on higher ASIL components
  - It should be expected that the pair of components have been already investigated by the cascading failures analysis
  - In this case, the checklist for cascading failure is re-used
- Output may be a new ASIL level for the SW component. In fact, status may have values:
  - **No impact**: failure of the lower-ASIL or QM component have no effect on the higher ASIL element
  - **New ASIL**: failure of the lower-ASIL or QM component propagates to the higher ASIL element , and a new evaluation of assigned ASIL is required
  - **Architecture review**: assigned ASILs are not changed but architecture is reviewed.

SW DFA: Common Cause Analysis: resulting table (example)

ID	ID of SW components under analysis	Cascading failure checklist	Component failure description	Likelihood	System Failure Mode (Effects)	Severity class	Detectability	RPN	New ASIL	Status
COA-1	Compute runtime (ASIL B), Application (ASIL D)	Timing	A completion or failure indication is received too late - A job hasn't completed on time	HIGH	Application cannot proceed	YES	External watchdog timer expired to indicate to the applications that the job hasn't completed	HIGH	ASIL D	New ASIL

RESILTECH

1. Short Company Introduction
2. SW Safety Analysis and DFA in Automotive
3. ResilTech Methodology
4. **Feedback from application and future directions**

Feedback from application 1/2

- Positive
  - Having a clear method to follow
    - ***To verify the completeness of the Safety Requirements and Mechanisms and also Assumption of Use in particular in case of SEOOC***
    - ***Standardize requirements for supplier: most important for long supply-chain as in automotive***
  - Having a guideline on selective application of Safety Mechanisms (run-time)
  - Good acceptance from Quality Departments

Feedback from application 2/2

- Negative
  - Effectiveness of analysis highly depend on detailed SW architecture design
    - Typically not available when it should
  - Once a potential safety impact is found it is not always straightforward to motivate usage of on-line error detection and mitigation techniques versus process oriented solutions (e.g. "improve" SW testing)

Future directions

- Developing low complexity modelling facilities
- Running model execution to evaluate “severity” prior to SW development
- Connection with Fault Injection campaign to validate “detectability” post development
- Formalized / semiformalized SW architecture would allow to be input for semi-automatic analysis
  - Despite a number of tools and methodologies available in last decades adoption from industry is still far from becoming a common practice

RESILTECH

Questions and (hopefully) Answers