life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni

Functional Safety Expert

STMicroelectronics

Lesson #1

General introduction on safety including HARA

Summary:

- **General concepts: risk, safety functions, Hzard and Risk Analysis, risk reduction**
- **Safety standards**
- **Faults, failures, HRF vs systematic**

life.augmented

Why Functional Safety is important in everyday life

Protects human life and health by preventing accidents and hazardous failures

Reduces risk of property damage and environmental harm caused by system malfunctions

Ensures reliable operation of critical systems in vehicles, medical devices, and industrial equipment

Builds user trust in technology through consistent and safe performance

Minimizes downtime and costs associated with failures and recalls

Enables innovation by providing a safety framework for new technologies and automation

Some useful definitions

Hazardous event/hazard: an event which intrinsically have the capability to cause a harm (physical injury/death of people, or damage to things)

Risk: is associated to a hazard, and it combines the possibility of occurrence of harm and the resulting consequences (how much severe the harm is)

Tolerable/acceptable risk: risk which can be reasonably considered is accepted in a certain context or situation. It clearly depends on the current system of values adopted in the society

Safety: the absence of risks that are considered unacceptable.

E/E/PE acronym: Electrical/electronic/programmable electronic

What is Functional safety (IEC61508 definition)

It is a part of the overall safety (Overall safety >> functional safety; contribute to achieve tolerable risk)

Depends on the correct functioning of the control system (in our case, the E/EE/PE)

It may depend on additional measures capable to reduce the risk

About “risk” concept

The risk concept is usually based on three parameters:

Severity: how many people will be involved, and how (injuries/death…)

Exposure: how often we’re running the given risk

Controllability: is there a chance for involved people to control in some way the effect of the system failure

The three parameters are combined in a kind of cross-matrix to derive resulting risk.

life.augmented

The airbag “paradox”

One impressive example of risk classification is given by the well-know car airbag. In airbags we have two separate safety functions: a) Firing: to fire the airbag when it is needed (car accident) and b) Safing: do not fire the airbag when it is not needed (no car accident).

Parameter	Firing failure (airbag not fired when car has an accident)	Safing failure (airbag fire when no car accident)	Comparison
Severity	Medium: only the driver will be affected	High: loosing the control of your car (consequence of unexpected firing) you can involve pedestrians or other car’s drivers	Safing > Firing
Exposure	Low! When actually you have an accident so one/two time in your life, hopefully ☺	High! Any time you drive your car, even in the parking	Safing >> Firing
Controllability	None	Low or none	Safing = Firing

Conclusion: risk associated to Safing failure is higher than Firing case. That was not obvious too.

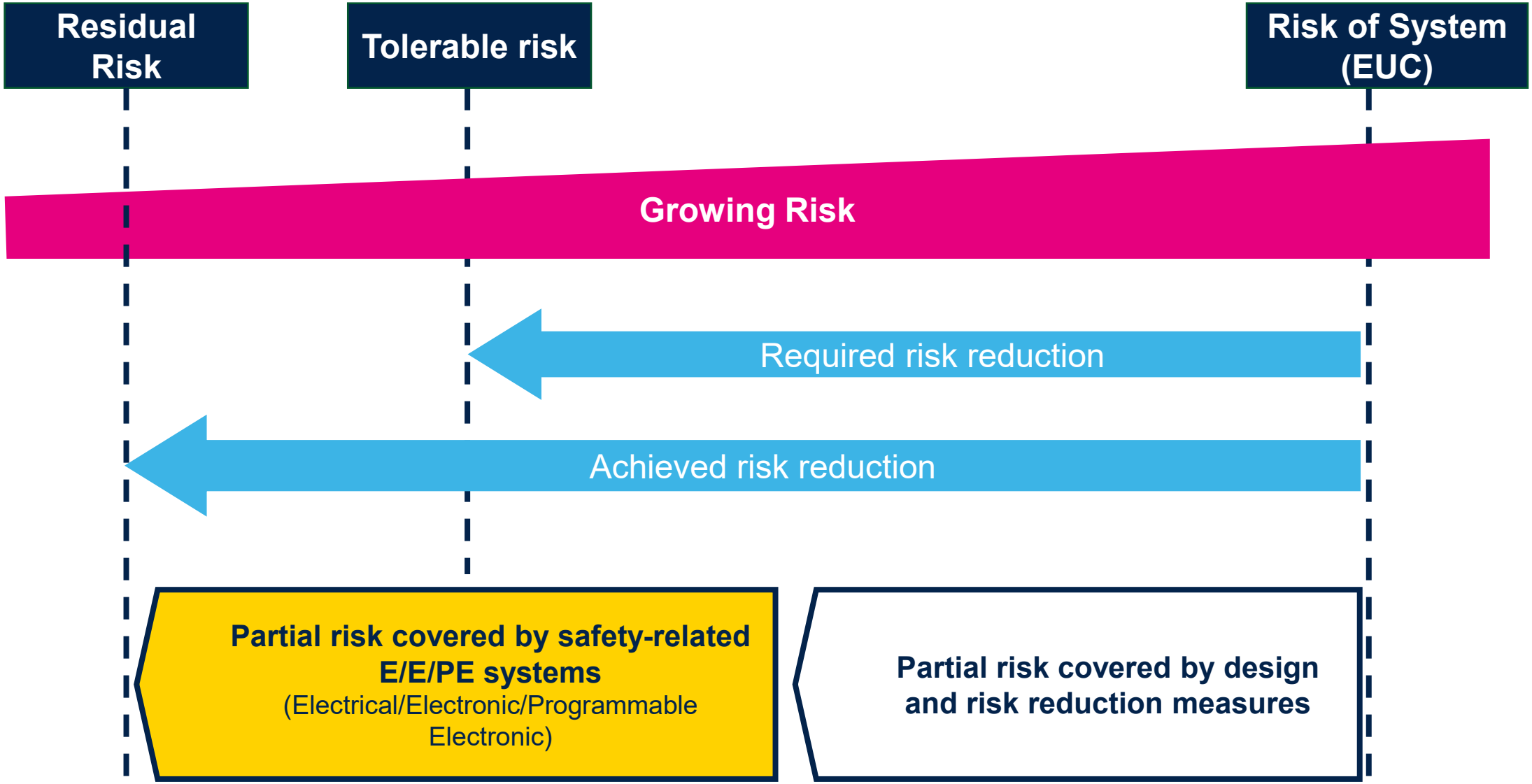
life.augmented

About “risk” concept

- We are not speaking about “risk elimination” (impossible!) but “risk mitigation” so avoiding unacceptable risks

- The concept of “tolerable risk” is in continuous evolution:
  - Once a new technology became widely diffused, people expectation for absence of correlated risk tends to increase (see for instance civil aviation)
  - Market/sector dependency, correlated to general public perception (e.g. space missions are reasonably believed to be more dangerous then commercial flights) and also legal approach (see for instance car market where the probability of high-cost class actions pushes for increased safety)

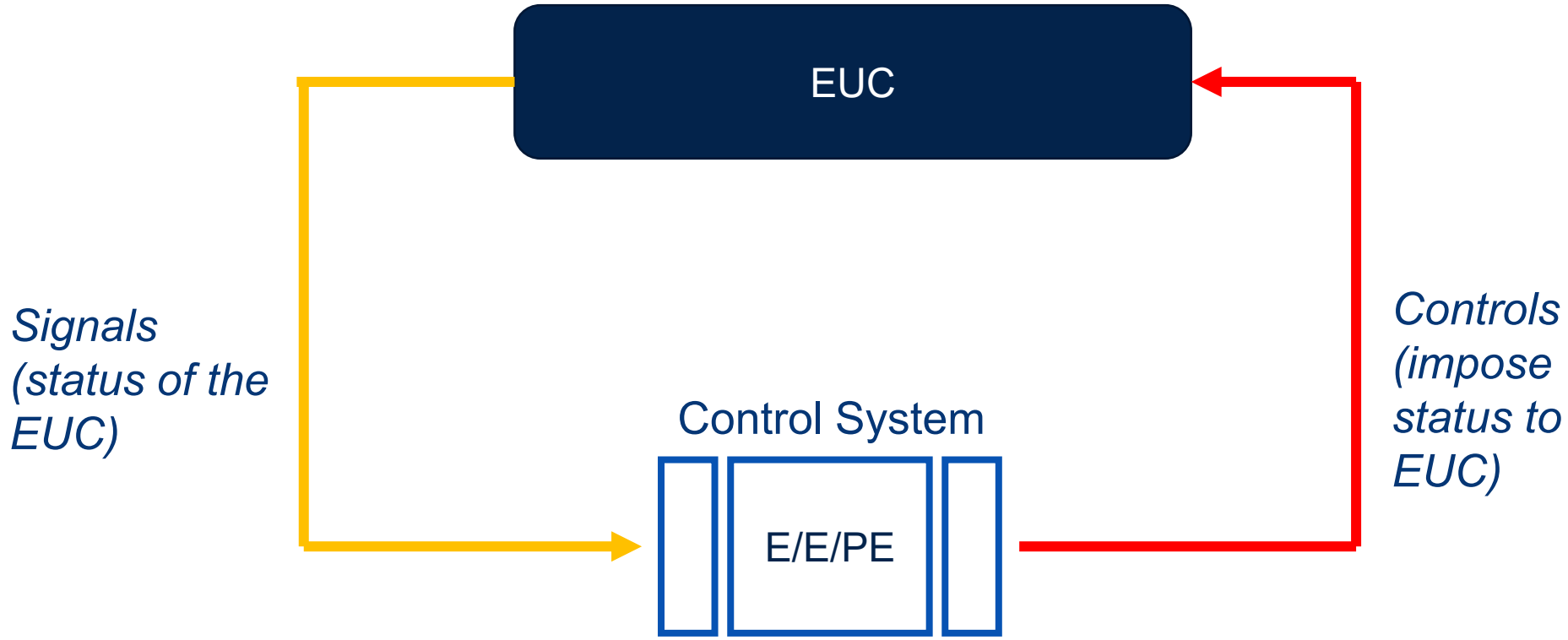
Risk Reduction



life.augmented

The EUC/control system dualism

IEC61508 is based on the concept that the final system (“the plant”) can be described as dual structure: controlled system (EUC, Equipment Under Control) / control system (i.e. kind of feedback control scheme).



The protagonist: the safety function

Safety function: function implemented by an E/EE/PE safety-related system (potentially also with the participation of additional measures to reduce the risk), that is

- intended to achieve or maintain the safety (so, no unacceptable risks) for the EUC,
- defined in dependency to a specific hazardous event

Examples of safety functions:

- *Functions required to be executed as positive action to avoid hazard (e.g. stopping the motor of a robot arm)*
- *Functions preventing actions being taken (e.g. preventing to open a door when a train is moving)*

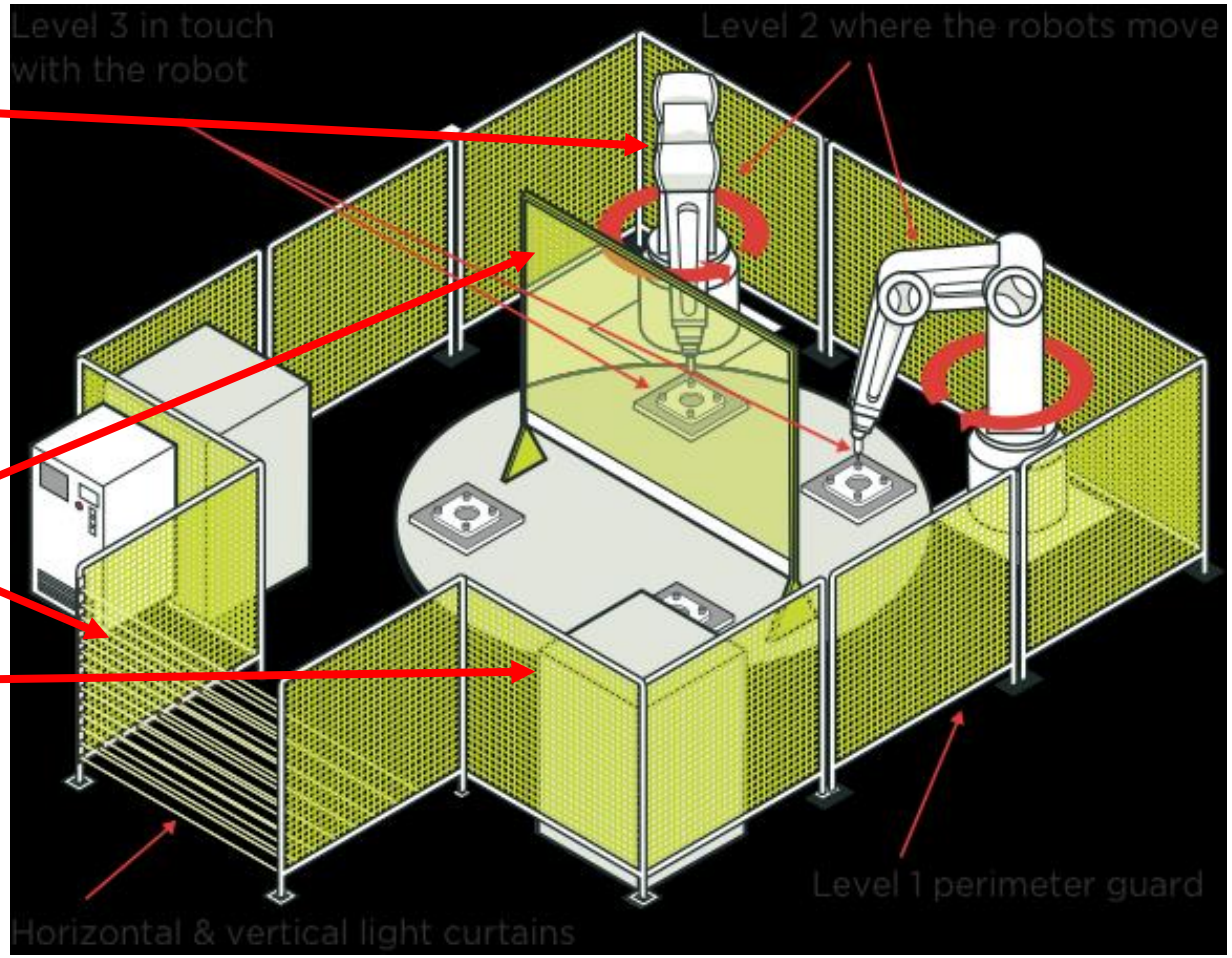
Mapping example – robot arm cage

part of the overall safety

EUC and the EUC control system

that depends on the correct functioning of the E/E/PE safety-related systems

risk reduction measures.



life.augmented

Hazard and Risk Analysis

Hazard and Risk Analysis is the process aimed to derive the target “safety integrity level” (i.e. “how much safety we need to add to the system”) according to the evaluation of several factors like exposure to the hazard, severity of the consequences, controllability and/or possibility to avoid the hazard.

Safety standards provides only guidance and not mandatory procedure. E.g., risk graph approach.

HARA example – ISO26262

Severity (S)	Exposure (E)	Controllability (C)	ASIL Outcome
S3	E4	C3	ASIL D
S3	E4	C2	ASIL C
S3	E4	C1	ASIL B
S3	E3	C3	ASIL C
S3	E3	C2	ASIL B
S3	E3	C1	ASIL A
S2	E4	C3	ASIL C
S2	E4	C2	ASIL B
S2	E4	C1	ASIL A
S1	Any	Any	QM

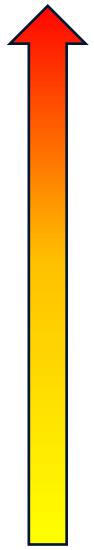
Severity (S) — Examples range from S0 (no injuries) to S3 (life-threatening injuries or fatality).

Exposure (E) — Ranges from E0 (incredible) to E4 (high probability).

Controllability (C) — Ranges from C0 (controllable in general) to C3 (difficult to control)

How to measure safety: Safety Integrity Levels

Safety integrity: it is a way to measure of the probability that a E/E/PE safety-related system performs in correct way a specified safety functions under, under specific conditions and time. Accordingly, there are Safety Integrity Levels (SIL)



SIL 4 (Nuclear plant, railways)

SIL 3 (Motor control, heavy industry)

SIL 2 (Industrial robot, small machinery, medical)

SIL 1 (E-bike, small appliances)

Similar “levels” appears in other safety standards (ASIL A->D in ISO26262, PL a->e in IEC13849, etc…)

life.augmented

Safety standards ecosystem

IEC 61508-4

Edition 2.0 2010-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

IEC61508 is the meta-standard

Each “legacy” safety standard defined his proprietary:

- Scope of application (refined)
- Risk evaluation criteria (tuned on application)
- Safety integrity levels

General concepts and definitions are generally inherited from IEC61508.

life.augmented

Safety standards: Prescriptive vs. Risk-Based Approach

Feature	Risk-Based	Prescriptive
Approach	Risk-based Safety lifecycle	Fixed, based on detailed requirements
Application	Broad industrial safety-related systems	Specific products lines
Risk Assessment	Central and mandatory	Minimal or none
Safety Lifecycle	Defined and enforced	Often not defined
Verification	Formal verification & validation	Prescribed tests
Flexibility	High	Low
Certification Focus	Functional safety & Safety Integrity Level achievement	Product compliance

Risk-based: IEC 61508 and all its derivatives, ISO 13849

Prescriptive: IEC 60730/60335, UL 1998

life.augmented

Faults and Failures

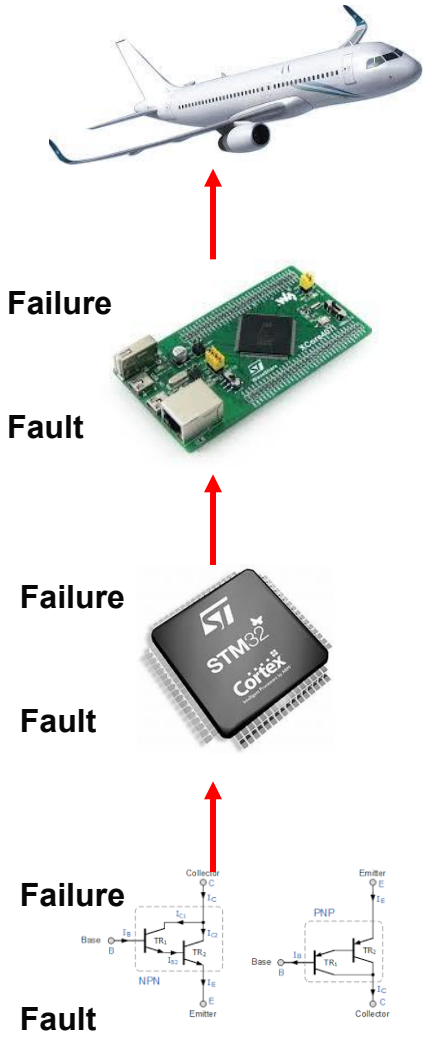
Fault: abnormal condition that may cause a reduction/loss of capability in a functional unit to execute a specific required function

Failure: termination of the ability of a functional unit to provide a required function/operation in any different way than as required

Definition is generic so include any kind of fault and failures.

Faults are the cause of failures.

Fault ⟶ Failure



Failures propagate from lower to upper layers in “modular” systems.

Why we focus on failures

Faults are always the root cause. BUT if they do not cause a failure, no risk is associated.

Faults “emerge”, are observed only through the caused failures.

A single fault can be source for multiple failures.

Safety **depends on the correct functioning of the E/E/PE safety-related systems**

and to possible other risk reduction measures.

We focus on failure as they are the termination of the system capability to execute a function, and therefore potentially to prevent that safety is achieved

Harm is caused by a wrong or missing functioning of the system – therefore, caused by failure(s) and not by fault(s).

The Functional Safety crossroad

random hardware failure: failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware

systematic failure: failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors

WARNING: this key concept is source of continuous misleading

life.augmented

Two kind of failures: RHF vs Systematic

Random Hardware Failures: the system is physically damaged (in permanent or transient way) leading to the failure of the expected functionality

- Based on probability analysis (unpredictable)
- Mainly quantitative (numbers!)
- Countermeasures based on fault detection and control

Systematic Failures: the system is wrongly designed and so under certain combination of external/internal conditions, or inputs, it will deviate from expected functionality (“bugs”)

- Based on process/method analysis (deterministic)
- Mainly qualitative (guidance)
- Countermeasures based on fault avoidance (process quality)

What is (NOT) safety

Functional Safety is NOT security

Safety: deals with failures on devices/software (system not working due to a fault)

Security: deals with intentional breaches/hacks on devices/hardware (system information and/or control are violated by an external attacker)

(Note: usually functional safety take into account as potential hazard a reasonably foreseeable misuse by the end user, and not voluntarily misuse).

Functional Safety is NOT reliability

Safety deals with the capability of a system to detect abnormal situations and to reach a specific safe state. Reliability deals on lifetime duration of systems perfectly working (all nonworking systems are considered the same – no mitigation/detection concepts exists, just redundancy if the case)

English wording: no misunderstandings (?)

… we will extensively use English wording in this training because, unfortunately, other European language are a bit misleading:

Sécurité
Sicurezza

Safety: deals with failures on devices/software (system not working due to a fault)

Security: deals with breaches/hacks on devices/hardware (system information is violated by external attacker)

Furthermore, despite IEC61508 has been published in English and French, English wording are well-known among industrial and safety community, so this last one allows major audience/understanding for documents..

… but still among English wording some misalignment still exist…

life.augmented

Hardware Random Failures theory

Summary:

- **Faults and failures (random hardware)**
- **Failure rate**
- **Faults classification**
- **Safety metrics: absolute and relative**

Hardware random failures: FAULT MODELS

For semiconductors components, two main falt model exists:

- Permanent faults
- Transient faults

Permanent faults

- The fault is irreversible (e.g. transistor broke, memory cell definitely open or short)
- Sources: aging, temperature stress

Transient faults

- The fault is not permanent (it can be a bit flip in a register, or a glitch in logic)
- Bit flips (aka known as “soft errors) can be corrected (or erased e.g. refresh for a FF)
- Sources: EMI, package radiation (alpha particles) , Sun radiation (neutron particles)

Permanent faults

Safety standards like IEC 61508 or ISO 26262 detail the minimum set of permanent faults to be considered (mainly as a “guidance”_:

- Stuck at 0/1
- Open circuit
- Short circuit
- Bridging
- High impendence
- Drift
- Oscillation

Note: they can be deen as “failures” with underlying faults generating them – depends on the abstraction level selected!

Transient faults

Also for trandsient faults .safety standards like IEC 61508 or ISO 26262 detail the minimum set of faults to be considered (mainly as a “guidance”_:

- Bit flips on registers
- Bit flips on volatile memory cells (RAM)
- Glitches on busses/connections/inputs

How to establish faults/failures

Given a certain component or technology, questions arise on how to establish a reasonable list of potential faults and resulting failures. Some main patterns are possible:

- Usually, each safety standard lists the minimum set of faults/failures to be analyzed according to the target safety integrity level (the higher the integrity , the larger will be the set of faults/failures)
- Collateral documentation listing potential faults/failures, see for instance reference NASA document [R1]. (search for “failure mode” for discrete and analog components)
- State-of-the-art tools for safety analysis (e.g. FMEDA tools) usually provide hands-on list of faults/failures for each given type of component.

How to deal with random failures

Random failures must be mitigated. Two ways are possible

- Fault avoidance
- Fault detection

Fault avoidance

- The probability of failure is decreased, for instance by redundancy of two independent functions. Best approach for mission-critical application and to increase availability. Concept associated Hardware Fault Tolerance (HFT)

Fault detection

- The system includes new, additional diagnostic functions devoted to detect failures. They are usually called safety mechanisms
- Once a failure is detected, a correction can be done (e.g. see ECC), or (if not possible) the system is informed of the failure, and safety is achieved by other means (system is driven in safe state)

life.augmented

How to deal with random failures - examples

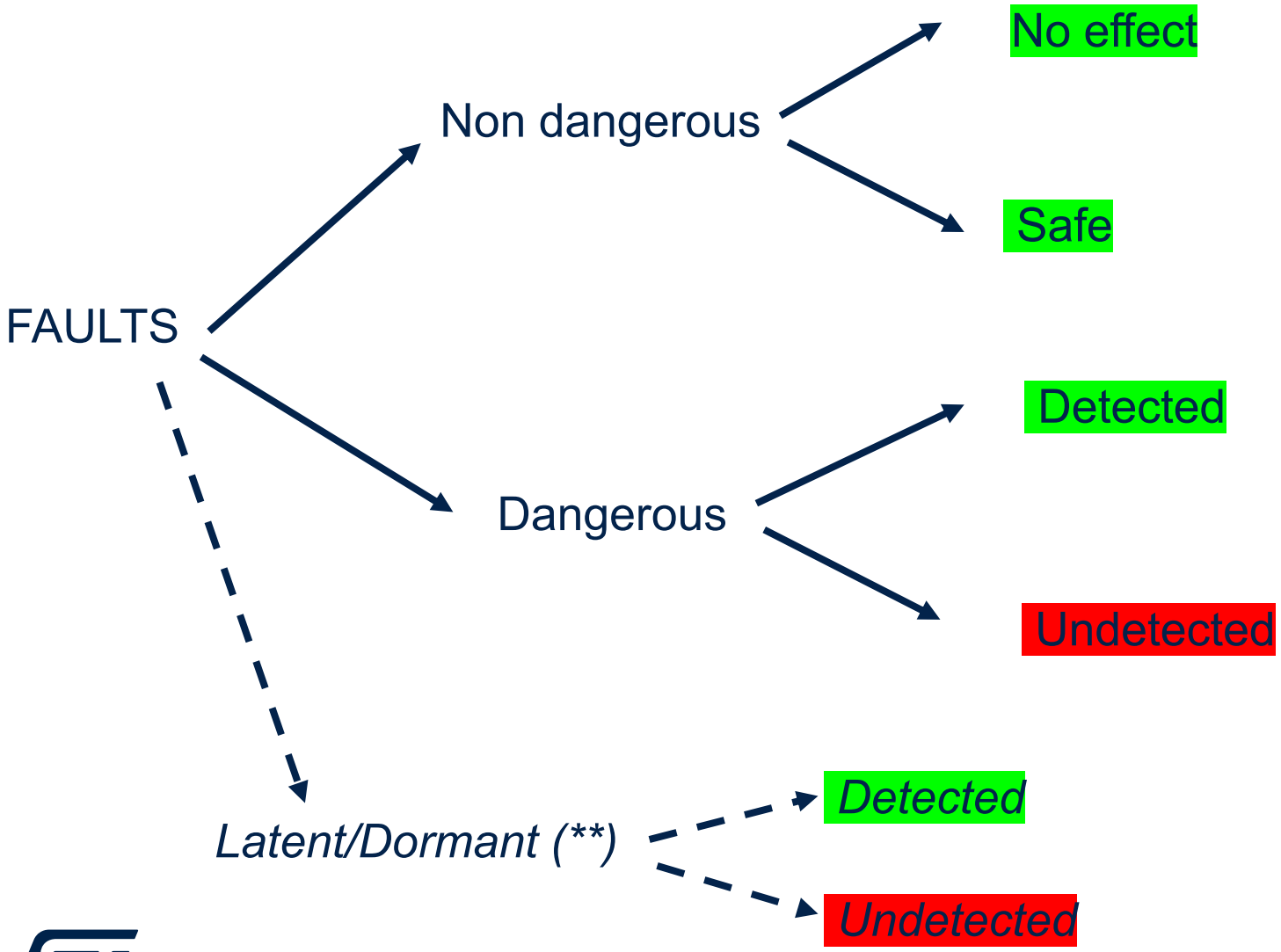
Fault avoidance example:

Register lock by key: Write access to configuration registers are protected by a key (software sequence of commands to be respected to unlock). Avoids faults related top unintended writes on the registers

Fault detection example:

Parity: a parity bit is added to each word (multiple schemes are possible), enabling single bit error discovery when data are read (computed parity bit doesn’t match with the stored one). Detects single bit flips faults.

Faults (*) classification



(*) Note: this terminology can be applied to failures as well

(**) defined only on some safety standard

Faults classification (definitions)

Faults/failures can be classified as:

No effect/no part/ non safety related: affecting hardware not involved in the implementation of the safety function

Safe: fault/failure driving (or helping to maintain) the system to a safe state (where safety is achieved)

Dangerous: able to interfere with the safety function

Detected: a fault/failure for which its presence is revealed by a safety mechanism(s)

Undetected: the vice-versa of previous

Latent/dormant: a fault which cannot directly interfere with the safety function, but which can do that in presence of another one.

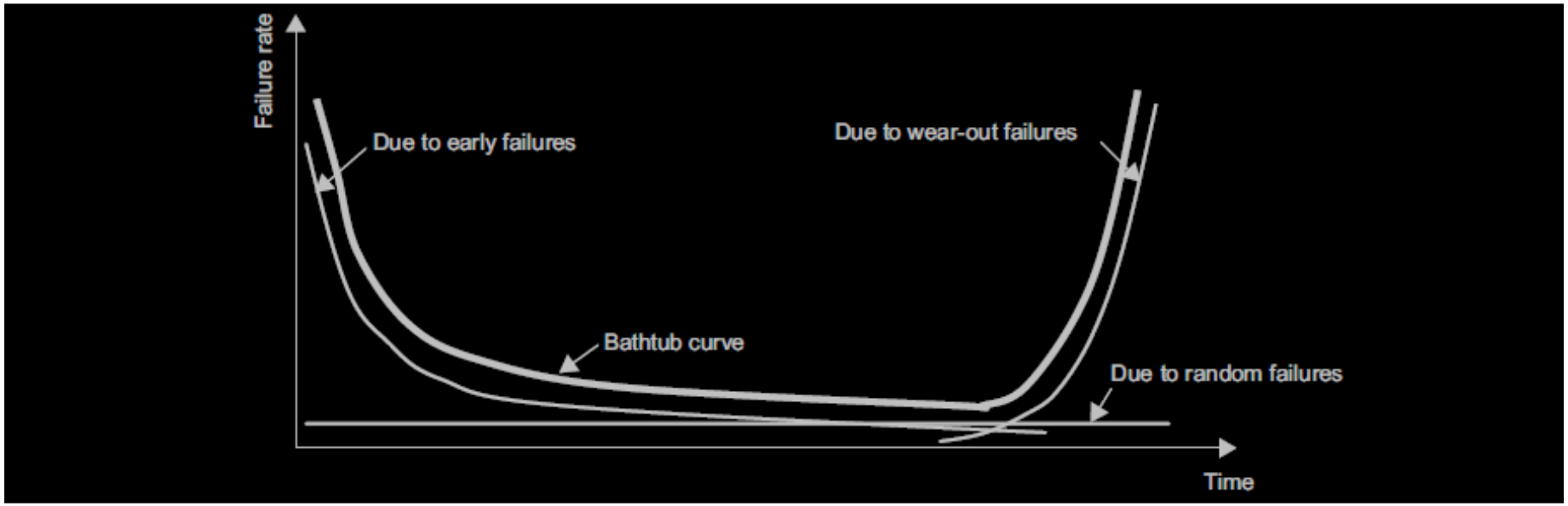
Note: general definitions, a mix between IEC 61508 and ISO 26262 definitions

Semiconductors failures

Early Failures: in the production process, semiconductor devices may contain defects due to the presence of tiny particles, as well as variations in manufacturing equipment and variations in dimensions. This fact is known as the initial defect density.

Random Failures: Failures resulting from production defects will attenuate with time.

Wear-out Failures: semiconductors fail when they reach the limits of their basic durability. This period is called the region of wear-out Wear-out failures differ with differences in the stresses applied to the device while it is being used.



life.augmented

Discrete components

Discrete components tend to show a constant failure rate (resistors, inductors, ceramic capacitors), while few specific exhibit a mild bathtub curve (electrolytic capacitors).

PCB connections and mechanical connectors show a constant failure rate with a wear-out increase phase, mainly depending on aging and mechanical stress cycles.

Relay often show a bathtub-like curve with noticeable wear-out phase due to aging and commutations cycles/electric load.

CONCLUSION: during the reasonable operating life of an electronic system, all failures can be considered to be in their constant value phase.

Failure rate

failure rate: reliability parameter (λ(t)) of a single components or system (entity) such that λ(t).dt is the probability of failure of this entity within [t, t+dt] under the assumption that it has not failed during [0, t].

Failure rate λ is therefore a probability divided by time (important!)

It is measured in FIT(s); 1 FIT is equivalent to 1 failure per 10E+9 hours

The failure rate of a series of components/systems is the sum of the failure rates of each of them. The failure rate of redundant (parallel) systems is generally non constant.

About base failure rates

“Base failure rate” for a given semiconductor component is the failure rate associated to an individual, specific subset of the component itself: for example, portion of the silicon area, transistor, memory bits etc

Base failure rate is a controversial argument as different methods (e.g accelerated tests and/or mathematical models) with different confidence levels exists in the industry. Main potential issue is to combine data coming from a different source in the system FMEDA

- For permanent failures, the most popular data sources are
  - IEC62380 (and its equivalent model in ISO26262-11)
  - SN29500-2:2010 Siemens norm (currently a bit outdated but still widely used)
- For transient failures, strong dependency on IC technology, package (LA vs ULA), altitude, shielding. Data are usually coming from irradiation tests or ITRS tables

Safety Metrics – absolute vs relative

Absolute metrics

λ expressed in FITs (1 failure over 1 billion hours)

They stroingly depends on assumptions on operating conditions *temperature, cycles)

They provide a measure of the probability of failure over time

Relative metrics

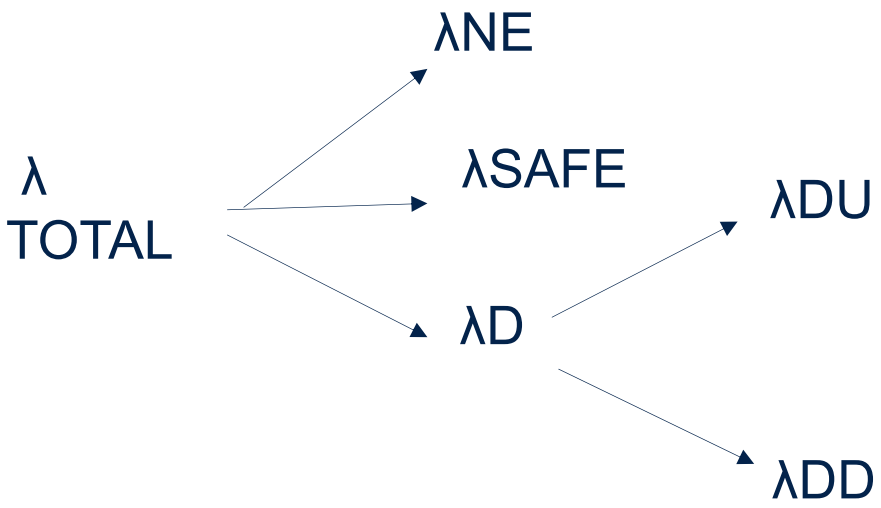
Expressed in percentages from 0% to 100%

They are the ration between homogeneous terms (λs) so they are only architectural dependent

They express a numerical evaluation on the overall combination of fault tolerance, safety mechanisms and mitigation measures.

About Safety Metrics – IEC 61508

Absolute metrics, λ expressed in FITs (1 failure over 1 billion hours)



Relative metrics
(percentages from 0% to 100%)

Diagnostic Coverage

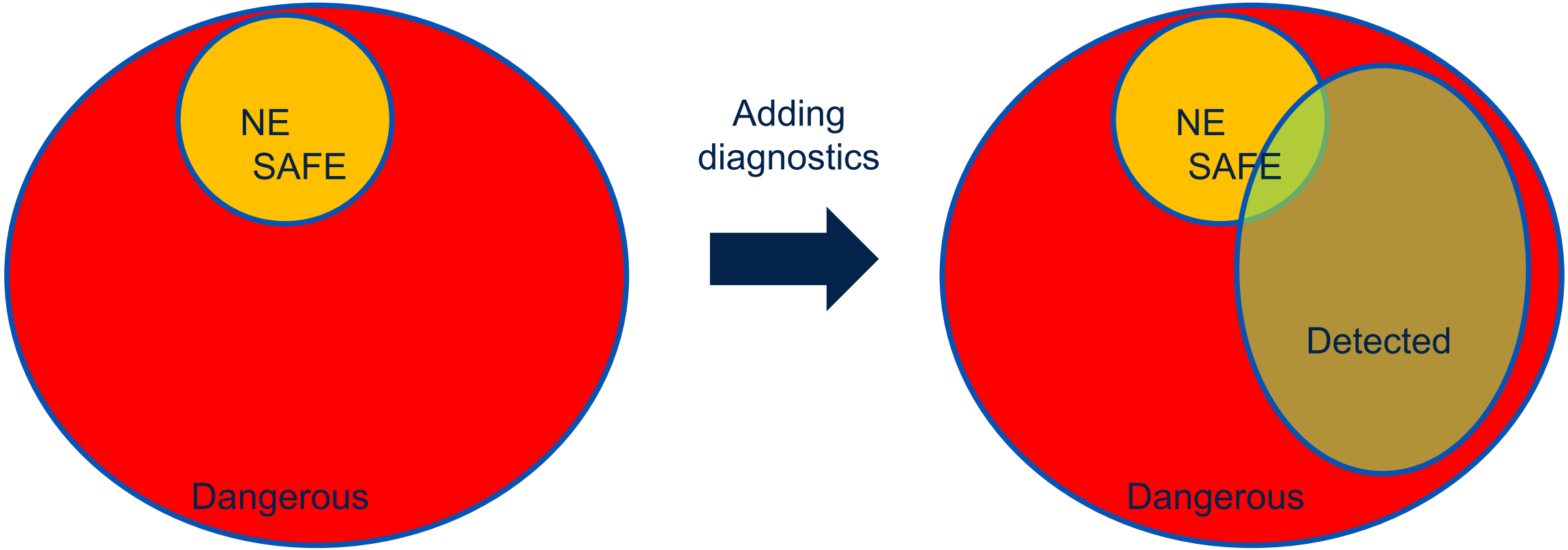
$$DC = \frac{\sum \lambda DD}{\sum \lambda D}$$

Safe Failure Fraction

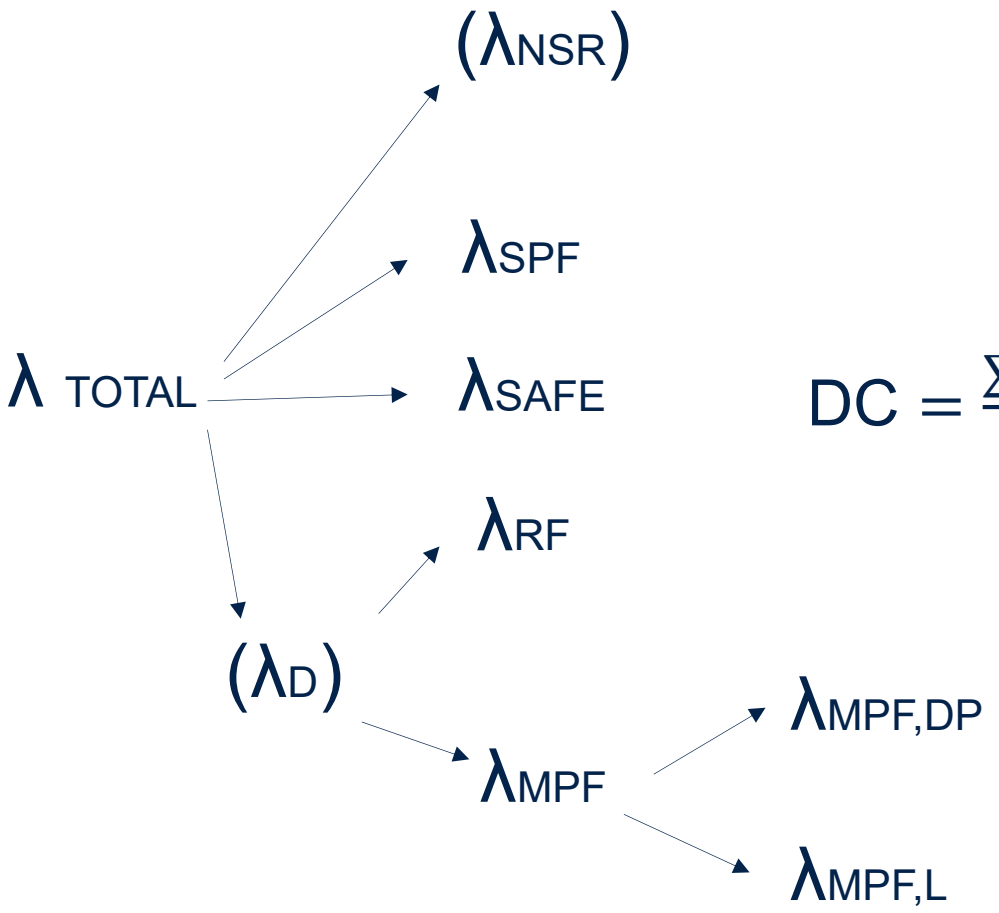
$$SFF = \frac{\sum \lambda DD + \sum \lambda SAFE}{\sum \lambda DD + \sum \lambda DU + \sum \lambda SAFE}$$

life.augmented

Faults detection



About Safety Metrics – ISO 26262



$$DC = \frac{\sum \lambda DD}{\sum \lambda D}$$

Single Point Fault metric

$$SPFm = \frac{\sum \lambda MPF + \sum \lambda SAFE}{\sum \lambda MPF + \sum \lambda SPF + \sum \lambda RF + \sum \lambda SAFE}$$

Latent Fault metric

$$LFm = \frac{\sum \lambda MPFdp + \sum \lambda SAFE}{\sum \lambda D - \sum \lambda SPF - \sum \lambda RF}$$

Note: ISO 26262 fault classification is quite specific because of its explicit classification for dual faults

life.augmented

Relative Safety metrics - targets

IEC 61508-2 relative metrics targets (*)

SFF	HFT = 0	HFT = 1	HFT = 2
< 60%	Not allowed	SIL 1	SIL 2
60% - 90%	SIL 1	SIL 2	SIL 3
90% - 99%	SIL 2	SIL 3	SIL 4
>99%	SIL 3	SIL 4	SIL 4

(*) For a Type B element

SFF is the reference

ISO 26262 relative metrics targets

SPFm	
90% - 97%	ASIL B
97% - 99%	ASIL C
>99%	ASIL D

LFm	ASIL
60% - 80%	ASIL B
80% - 99%	ASIL C
>90%	ASIL D

life.augmented

Absolute Safety metrics - targets

SIL	Average frequency of dangerous failure (for HD/CM)
SIL 1	1E3 FIT< PFH < 1E4 FITs
SIL 2	100 FIT< PFH < 1000 FITs
SIL 3	10 FIT< PFH < 100 FITs
SIL 4	1 FIT< PFH < 10 FITs

IEC 61508-2 relative metrics targets

ASIL	PMHF Probabilistic Metric for random Hardware Failures
ASIL B	PMFH < 100 FITs
ASIL C	PMFH < 100 FITs
ASIL D	PMFH < 10 FITs

ISO 26262 absolute metrics targets

Note: PHF/PMHF computation is linked to λDU/λRF values. Details will be provided in the lesson related to safety architectures

life.augmented

How many faults in the system?

In principle, 1 to N faults can affect in the same time the system. If faults are independent, the propability of multiple faults is low and anyway depends on the time.

Each safety standard provide explicit guidance to the minimum number of *simultaneous* faults to be considered in the analysis of the system:

IEC 61508 ask for single faults, nut ask to “consider” multipe, faults scenarios

ISO 26262 ask for single faults + a second fault just on the diagnostics (latent) N>2 is out of scope

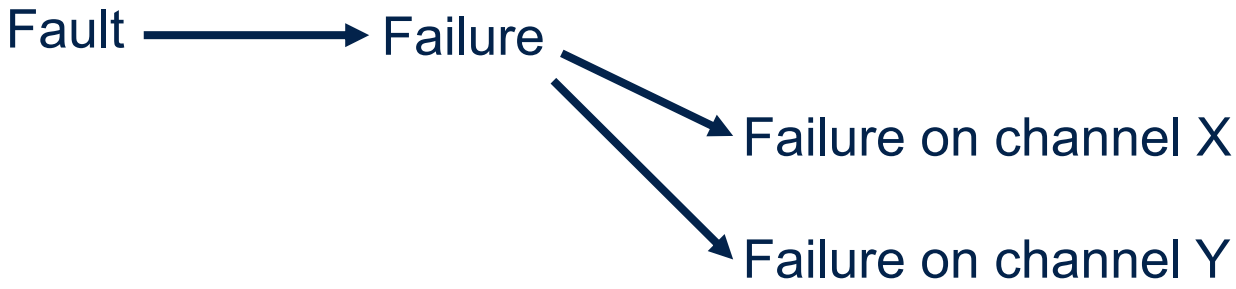
ISO 13849 requires only one fault except for specific PL/architexcture where HFT=1 is required

Non-independent failures

Dependent failure: .failures caused by non-independent events i.e. P(A and B) > P(A) × P(B).

Common cause failure: failure causing multiple concurrent failures in a multichannel system

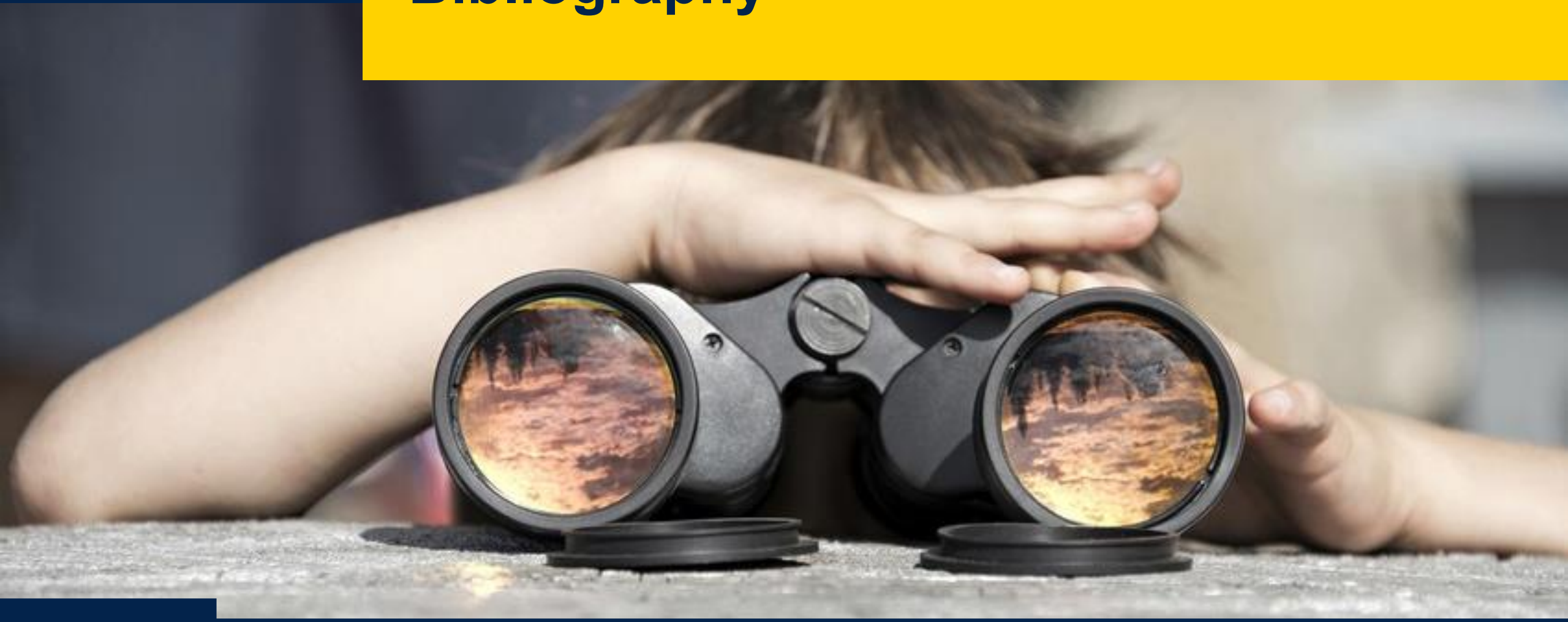
(example: failures of the common supply for a multichannel system)



Related problems

- They cannot be included in “standard” DC, λ computations
- They potentially undermine fault tolerance of the system

Bibliography



life.augmented

Reference documents

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

Thank you

life.augmented

life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni

Functional Safety Expert

STMicroelectronics

Lesson #3

Systematic Capability theory, including V model and preliminaries on SW and tools

Summary:

- Safety lifecycle
- V-model
- Notes on requirements
- Software development
- Tools assessment

life.augmented

How to deal with systematic failures

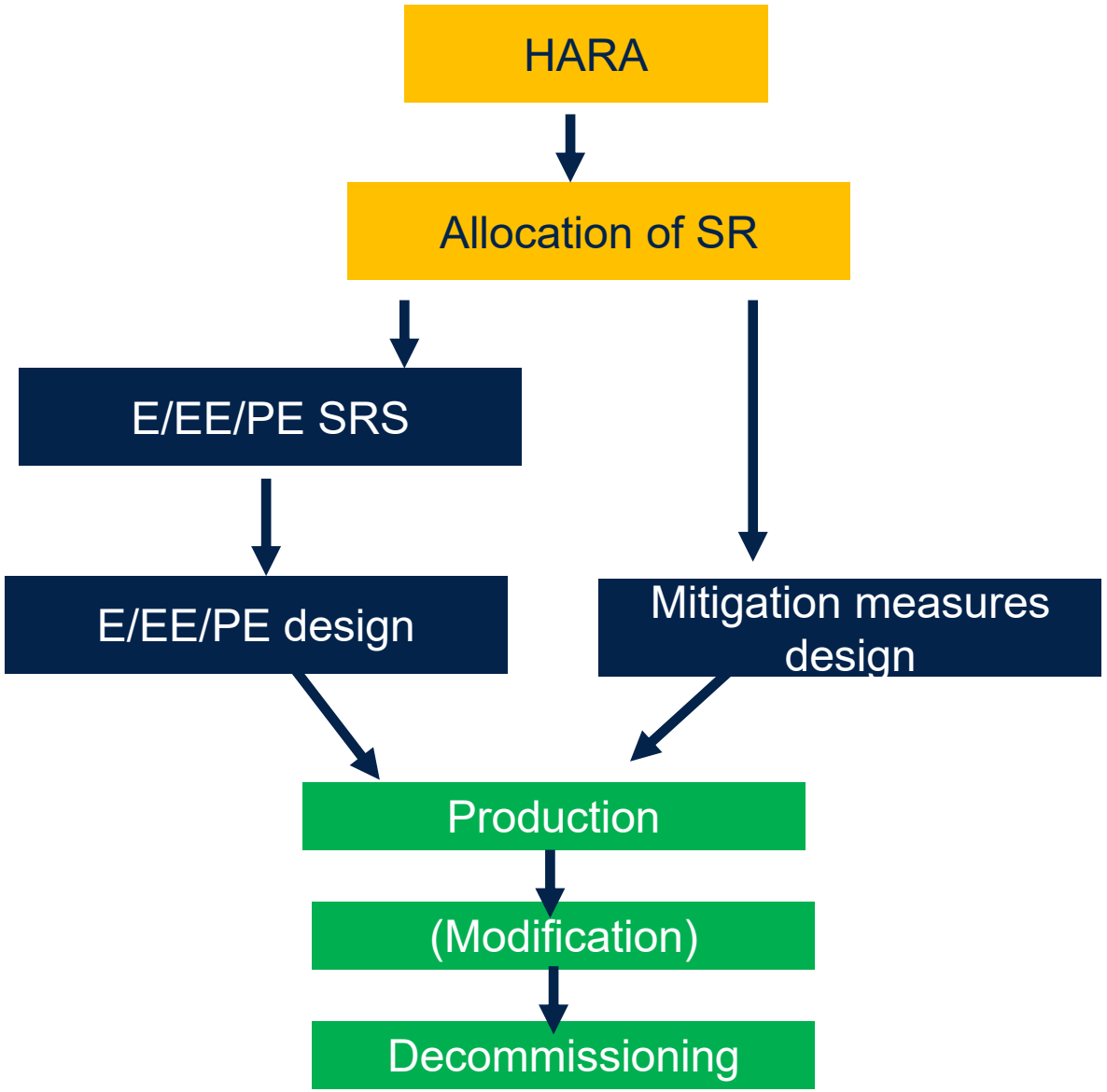
While most part of the safety standards requires to mitigate/consider systematic failures, related issue is more challenging for risk-based standards which set a specific safety integrity level also for systematic.

Two main patterns exist:

- Formal **safety lifecycle** (based on strict application of development rules tailored depending on the safety integrity level)
- **Proven in use argument**, based on evidence of stability/absence of systematic defects over time
- The two patterns can apply to embedded software and software tools as well.

Safety lifecycle

Each safety standard defines its specific safety lifecycle; the general structure is mainly the same:



Verification

Documentation

Safety Assessment

Verification & Validation

Verification is the process of confirming, through examination and objective evidence, that a product, system, or component meets its specified requirements. It answers the question: "Are we building the product right?:

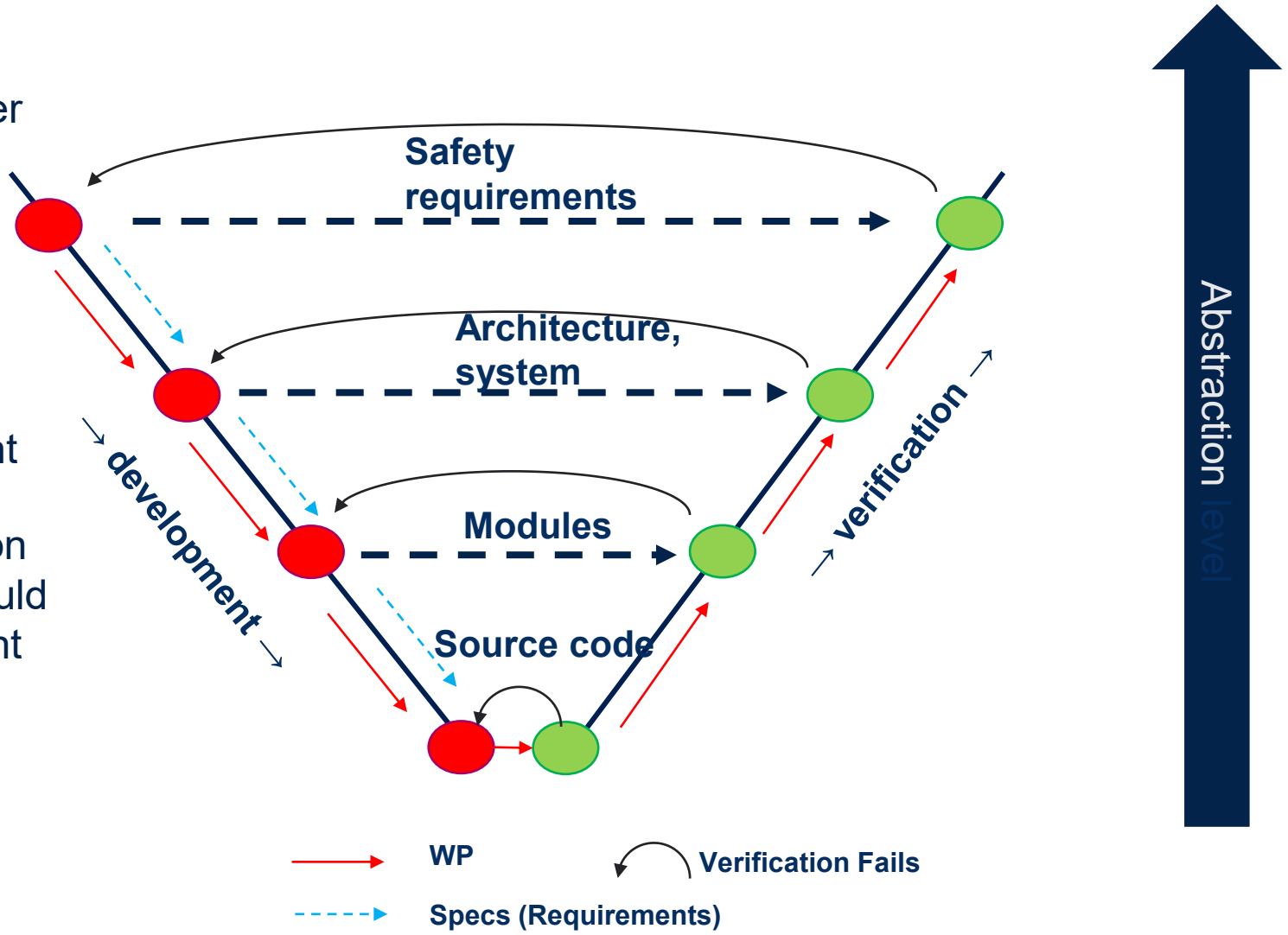
Validation is the process of confirming, through examination and objective evidence, that the product fulfills the requirements for its specific, intended use in the real world. It answers the question: "Are we building the right product?

Aspect	Verification	Validation
Purpose	Confirm requirements are correctly implemented	Confirm product meets user needs and intended use
Focus	Conformance to specifications	Fitness for purpose
Typical Activities	Reviews, inspections, unit/component testing	System-level testing, user acceptance testing

V-model characteristics (IEC 61508-3, ISO 26262-6)

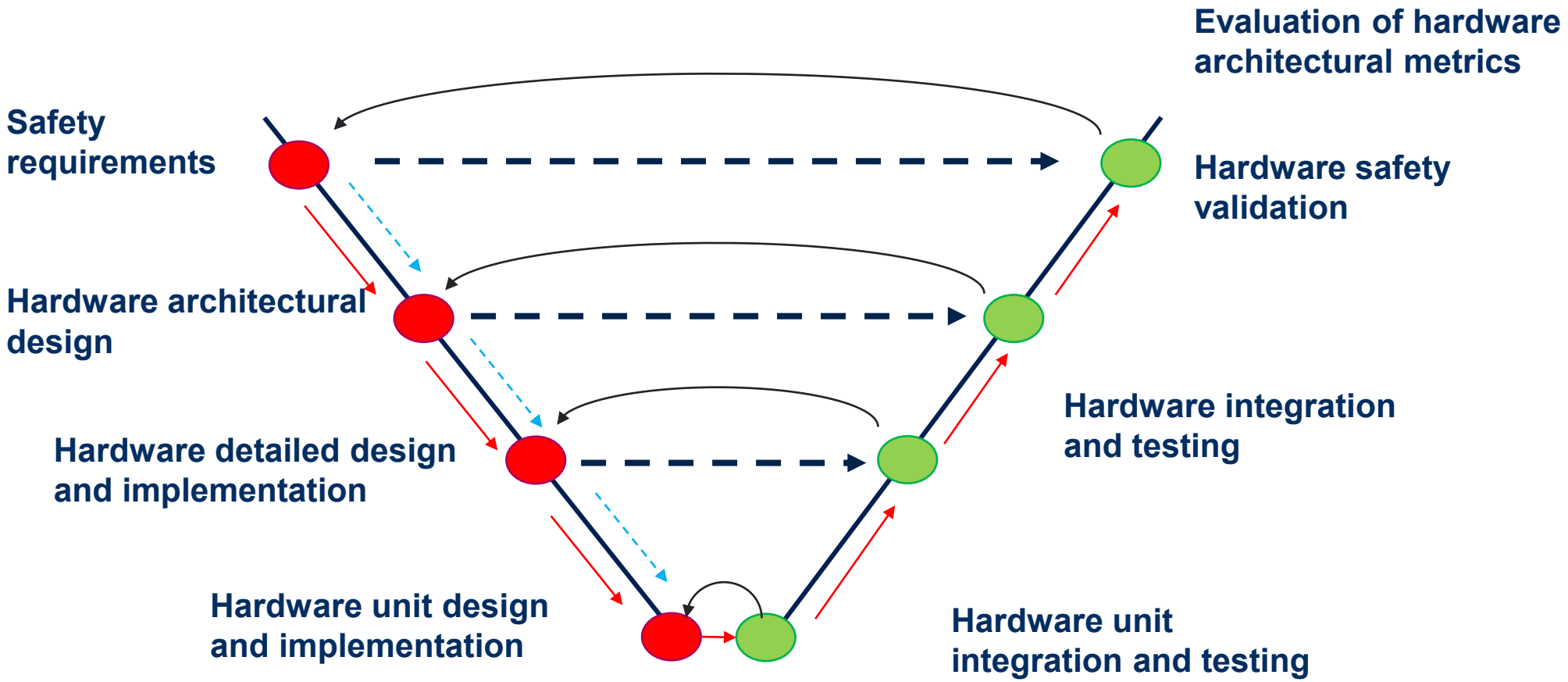
Advantages

- Top-down approach is forced
- Requirement-based client/server model between phases
- Inputs/outputs between phases (WP) well defined
- Testing specified at same abstraction level of development
- Non-compliance after verification is managed hierarchically (it could impacts the related development phases)
- Traceability bonded inside

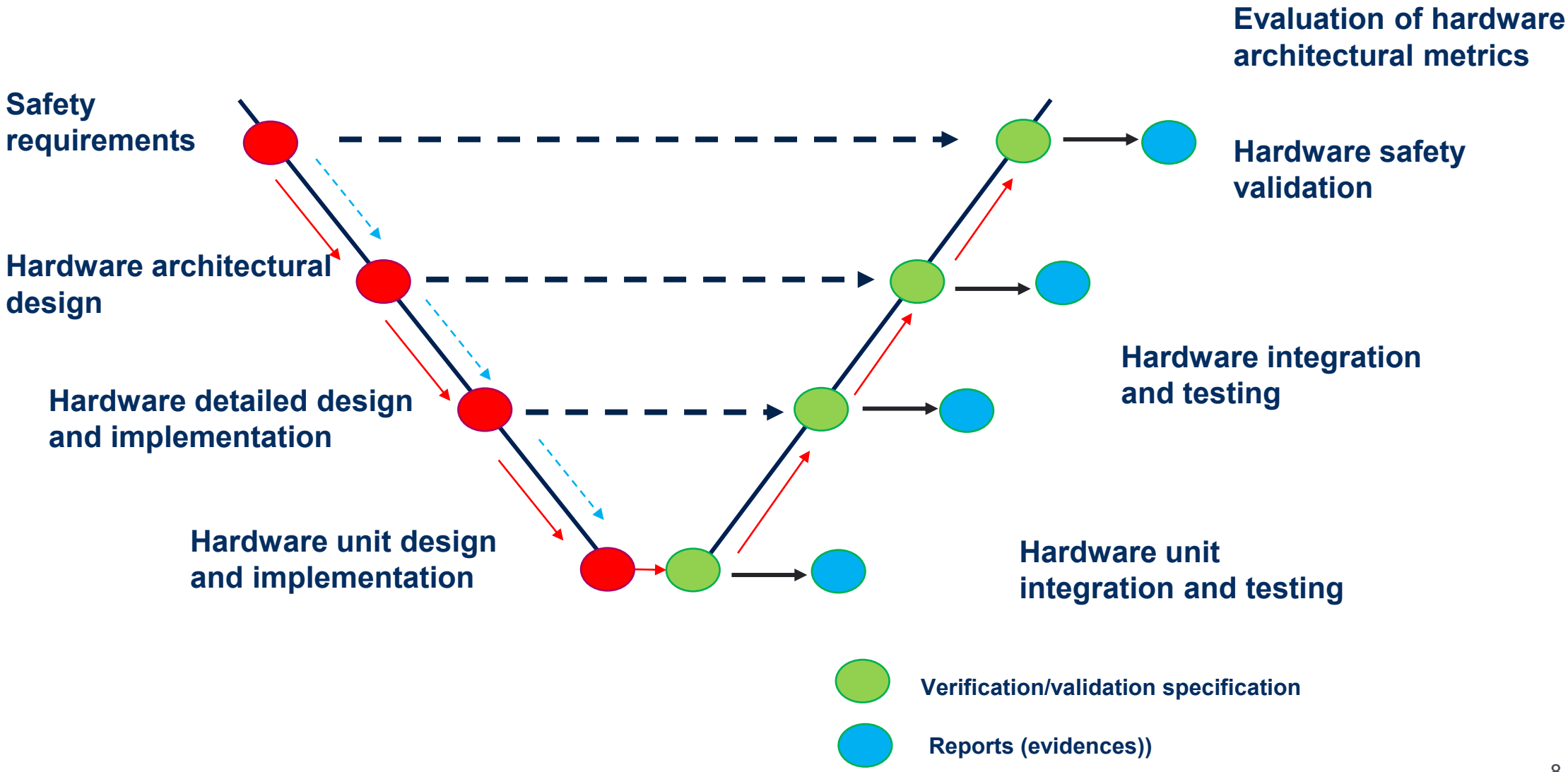


life.augmented

V-model for hardware development (ISO 26262-5) - phases



V-model for hardware development (ISO 26262-5) - documents



About traceability

IEC61508 V-model requires Forward and Backward traceability among several specification and verification requirements sets. HR for SIL3/4, just R for SIL1/2. Among other techniques, traceability is a very good asset for safety and quality in the development of the software:

Forward traceability: checking that a requirement is adequately addressed in later lifecycle stages.

→ Pro: key asset to proper evaluate a Change Impact depending on high level requirements update/change

Backward traceability: checking that every implementation decision is clearly justified by some requirement.

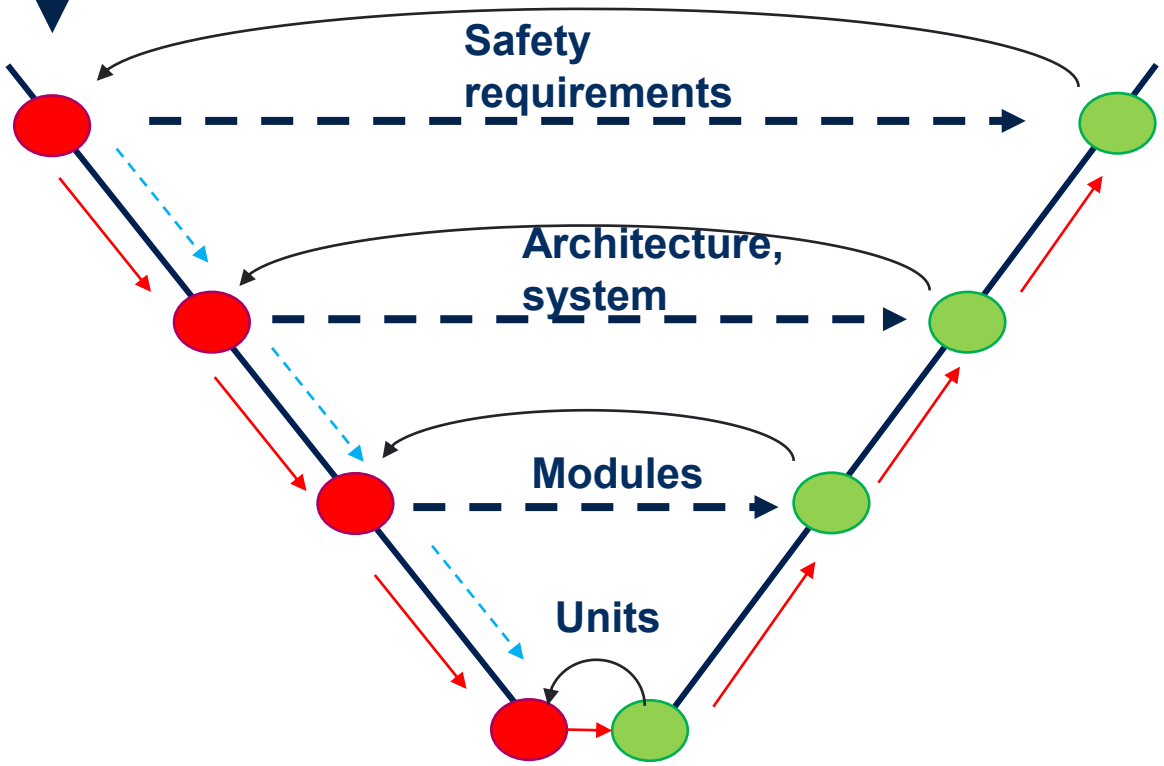
→ Pro: decrease probability of unused or unjustified functions/hw parts

WARNING: adding traceability lately in a project will turn out in a nightmare!

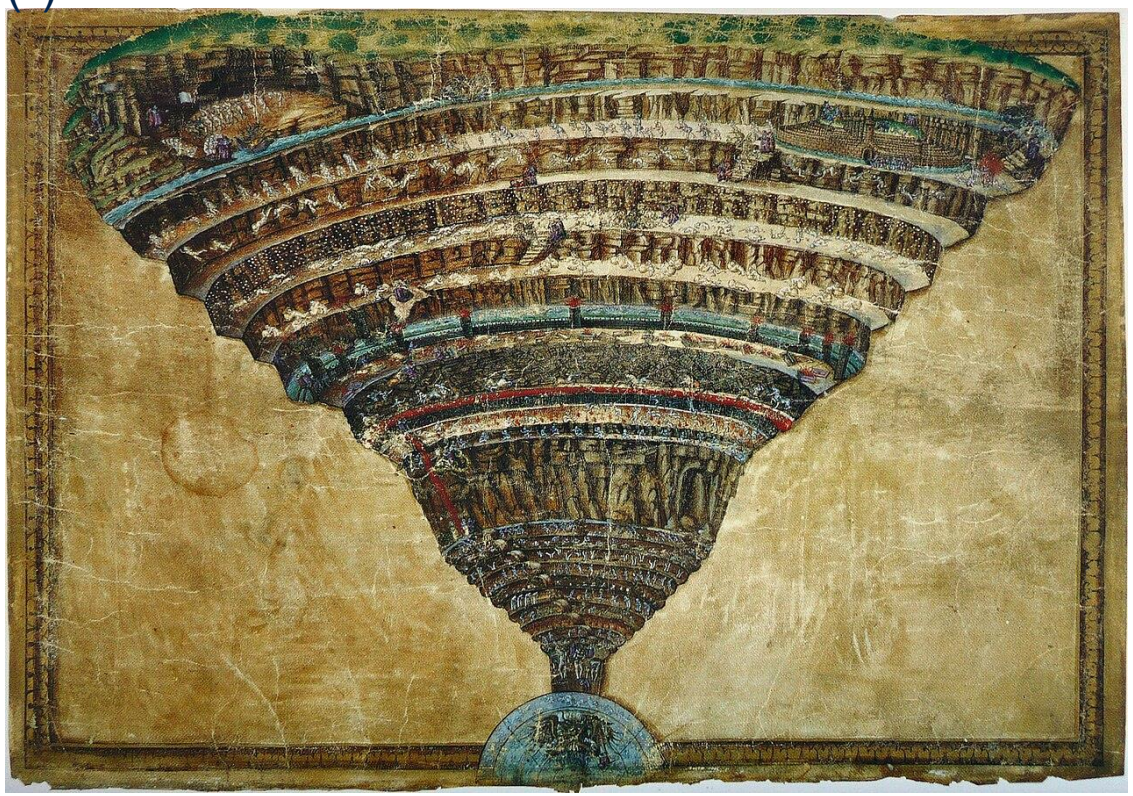
life.augmented

The V-model trap

V-model Devil is here – see reference [R3]



(*)



Dante's Devil is here

(*) *La voragine infernale, Sandro Botticelli (Source: Wikipedia)*

life.augmented

SOTIF (Safety Of The Intended Functionality)

ISO/PAS 21448 — provides guidelines for ensuring safety in advanced driver assistance systems (ADAS) and autonomous vehicles. SOTIF (Safety Of The Intended Functionality)

SOTIF addresses safety risks arising from the intended functionalities of a system, especially when no faults or failures are present.

SOTIF focuses on hazards caused by performance limitations, environmental conditions, or misuse, beyond traditional fault-based safety. It move the focus on incomplete system specifications, which by nature are not intercepted by the V-model.

SOTIF purpose is to identify and mitigate risks related to correct system behavior that can still lead to unsafe situations. It complements ISO26262 by covering scenarios where the system behaves as designed but still poses safety risks.

.

Methods tailoring in V model

The formal V model prescribes for each phase lists of recommended methods.
Recommendations are included in tables, and ranked this way:

HR/++ the technique or measure is highly recommended for related safety integrity level; If not used, then the rationale behind not using it should be detailed and agreed with the assessor.

R/+: the technique or measure is recommended for this safety integrity level as a lower

recommendation to a HR/++ recommendation or as an additional safety margin measure.

NR: the technique or measure is positively not recommended for this safety integrity level.

Methods tailoring in V model

Table notional example.

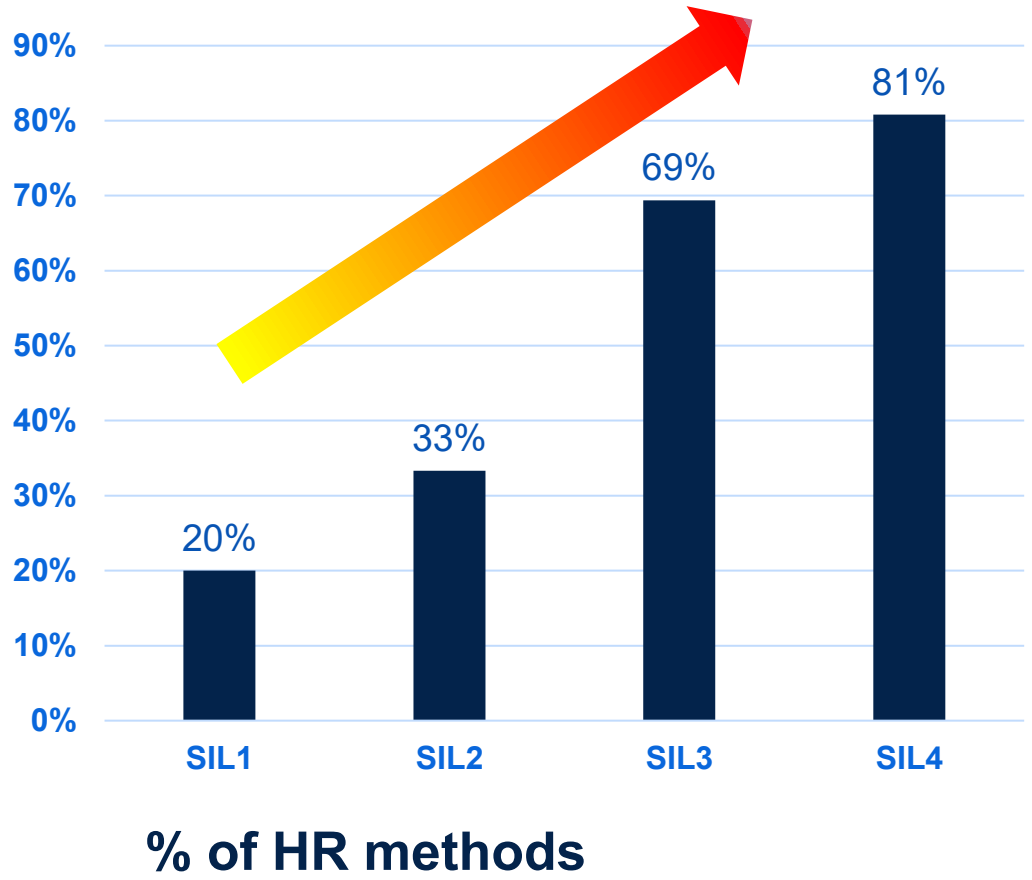
		SIL 1	SIL 2	SIL 3	SIL 4
1	Measures against voltage breakdown, voltage variations, overvoltage, low voltage	R	HR	HR	HR
2	Increase of interference immunity	R	HR	HR	HR
3	Spatial separation of multiple lines	HR	HR	HR	HR
4	…	…	…	…	…
5	…	…	…	…	…

life.augmented

IEC 61508-3 tables contents vs SIL/SC

Request of method application strongly depends on SIL level:

“Generic recommendations” turn into “detailed prescriptions” on low side of the V (e.g. software metrics coverages, numeric approach)



Requirements

A **requirement** is formally defined as:

a documented statement

describing a condition or capability

that a system, product, or service must satisfy

to meet stakeholder needs or constraints.

This definition is consistent across major standards in systems and software engineering.

Requirements are the basic pillar of the formal V-model.

What is really required?

life.augmented

Requirements qualities

ISO/IEC/IEEE 29148:2018 (Requirements Engineering) explicitly describes quality attributes that requirements should satisfy, including:

Atomicity: Requirement expresses a single need or capability (to avoid ambiguity and complexity).

Unambiguity: Requirement has only one interpretation.

Completeness: Requirement includes all necessary information.

Consistency: Requirement does not conflict with others.

Verifiability: Requirement can be verified by inspection, analysis, test, or demonstration.

Modifiability: Requirement can be changed without introducing errors.

Traceability: Requirement can be traced to its origin and related artifacts.

Correctness: Requirement accurately reflects the stakeholder need.

Assured by the V-model

life.augmented

How to write (good) requirements

Semi-formal methods:

- Structured Natural Language (SNL) – ("The system shall [action] [object] [under conditions].“)
- State diagrams
- Decision Tables
- UML use cases,

Formal methods:

- Petri nets
- Alloy
- …

life.augmented

Structured Natural Language (SNL) - example

Based on consistent template e.g.: The [actor] shall [action] [object] [under conditions].

Guidelines:

- Use Clear and Precise Language
- Use Active Voice
- Use Consistent Terminology (glossary)
- Avoid Negative Statements
- Use Singular Nouns and Consistent Numbering
- Limit Use of Pronouns

Proven in use (*) argument

Based on the demonstration, supported by operational experience over a specific, extended period of time, that the probability of unknown systematic failures is low enough for the target safety integrity level.

Main issues related to this approach are:

- ❑ Requires the presence of a credible monitoring procedure for on-field failures
- ❑ In some cases, hardly linkable to a specific systematic integrity levels
- ❑ Usually, suitable just for very simple components, because configuration changes can invalidate the argument
- ❑ Dependency on multiple factors including component manufacturing process

(*) different names can be found on safety standards ecosystem

Tools assessment

IEC 61508 and ISO 26262 defines a structured approach to assess the confidence in software tools used in the development and verification of safety-related systems. This ensures that tools do not introduce or fail to detect errors that could compromise functional safety.

The structure is similar, tools are assessed according to their capability to (negatively) affect the implementation or verification on the safety function. Then, an evaluation is done on the possibility to later identify those introduced issues.

The result of the assessment is the prescription of specific requirements on the tools, possibly requiring:

- Adoption of “certified” tools explicitly developed according a safety lifecycle
- Adoption of additional mitigation measures (e.g. tools output comparison, proven in use argument, etc.)

Tools assessment – IEC61508

A software off-line support tool is a software application that assists with one or more phases of the software development lifecycle but does not have any direct influence on the safety-related system during its runtime. These tools are categorized into three classes based on their interaction with the safety-related system:

T1 Tools: these tools do not produce any outputs that directly or indirectly affect the executable code (including data) of the safety-related system.

T2 Tools: these tools support the testing or verification of the design or executable code. While errors in these tools may cause defects to go undetected, they cannot introduce errors into the executable software itself.

T3 Tools: these tools generate outputs that directly or indirectly contribute to the executable code of the :safety-related system.

Tools assessment – IEC61508

The selection of the tools shall be justified. Then, following requirements apply:

- . **Documentation:**
  All T2 and T3 tools must have clear specifications or documentation detailing their behavior and usage constraints.
- **Assessment:**
  Evaluate T2 and T3 tools to understand how much they are relied upon and identify possible failure modes that could impact the executable software. Apply mitigation if needed.
- **Conformance Evidence (T3 only):**
  Provide evidence that T3 tools meet their specifications, based on successful past use and/or formal validation.

Lesson #4

Safety analysis methods (FMEDA/FTA/DFA/ETA/Markov)

Summar\y:

- **FMEA**
- **FMEDA**
- **FTA**
- **DFA**
- **Markov analysis**

life.augmented

Principles of FMEA (Failure Modes and Effects Analysis)

Proactively identify potential failure modes in products, processes, or systems to improve reliability and safety.

Key Elements:

- Failure Mode: Specific way a part or process can fail (e.g., crack, short circuit).
- Effect: Impact of the failure on the system or user (e.g., loss of function, safety hazard).
- Cause: Root cause or trigger of the failure mode (e.g., material defect, human error).

Use Risk Priority Number (RPN) or similar metrics based on:

- Severity (S): How serious the effect is.
- Occurrence (O): Likelihood of failure happening.
- Detection (D): Probability of detecting the failure before it reaches the customer

Iterative Process: update FMEA regularly during design changes, production, and field feedback to maintain effectiveness.

Example of FMEA

Notional example (process)

.

Process Step	Potential Failure Mode	Potential Effects	Potential Causes	(S)	(O)	(D)	RPN	Recommended Actions
Soldering components	Poor solder joint	Device malfunction or failure	Insufficient solder, operator error	9	4	3	108	Train operators, improve soldering process controls
Component placement	Misaligned components	Short circuit or open circuit	Misalignment during placement	8	3	4	96	Use automated placement machines, add visual inspection
Testing final assembly	Incomplete functional testing	Defective devices shipped to customer	Test procedure incomplete	10	2	5	100	Standardize test procedures, add test checklist

- **Severity (S):** Impact on system or user (1-10 scale, 10=most severe).
- **Occurrence (O):** Likelihood of failure (1-10 scale, 10= most frequent).
- **Detection (D):** Likelihood failure will be detected before release (1-10 scale, 1=highly detected).
- **RPN:** RPN=S×O×D*RPN=S×O×D*; higher values indicate higher priority

.

life.augmented

Principles of FMEDA (Failure Modes, Effects and Diagnostics Analysis)

Similar to FMEA with following specifics:

- Includes indication of Diagnostics (aimed to mitigate/detect the failures)
- Quantitative: get rid of RPN in favor of failure rates computations.
- It provides an overall result in terms of DC and SFF (SPF)
- For each failure mode line, includes information on associated fault model (to correctly compute failure distribution)
- Hw only: it cannot be applied to processes and software

Key topic is the failure distribution problem (how to associate individual failure mode to the system failure rate)

Principles of Fault Tree Analysis (FTA)

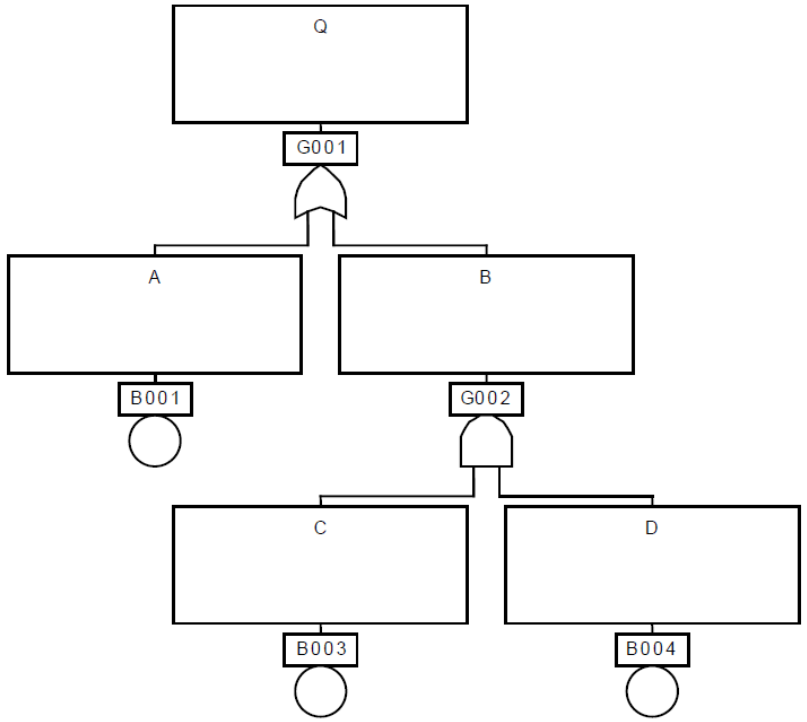
FTA is a top-down, deductive analytical method used to identify the causes of system-level failures.

Purpose: to systematically analyze how combinations of basic faults can lead to a critical undesired event (the top event).

Key Features:

- Starts with a defined top event (system failure or hazard).
- Uses logical gates (AND, OR) to map relationships between faults.
- Top-down analysis flows down until a basic event (root cause) is found
- Helps visualize failure pathways and their interdependencies

Principles of Fault Tree Analysis (FTA)



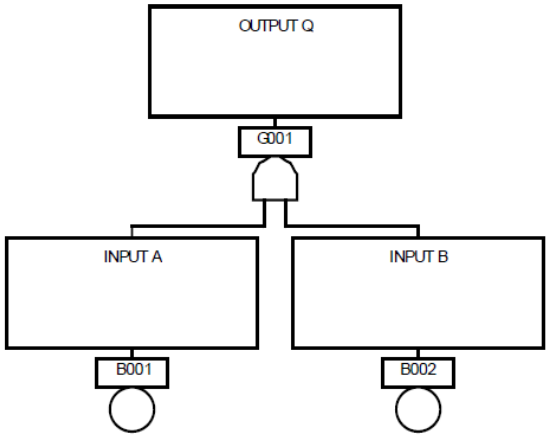


Figure 4-5. The AND-Gate

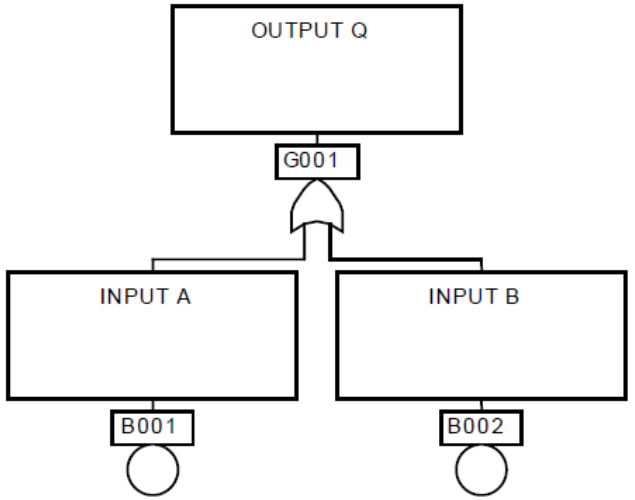


Figure 4-2. The OR-Gate

For full explanations refer to reference document [R4], section 4.1 Symbology—The Building Blocks of the Fault Tree

life.augmented

Principles of Fault Tree Analysis (FTA)

Quick recap of main FTA rules/specificities

Complete-the-Gate Rule: all inputs to a particular gate should be completely defined before further analysis of any one of them is undertaken.

No Gate-to-Gate Rule: gate inputs should be properly defined fault events, and gates should not be directly connected to other gates.

Minimum Cut Set: is the smallest combination of basic faults that can cause the top-level failure (top event). It represents a minimal set of component failures leading to system failure and one of the major benefits of underrunning a FTA analysis in your system..

Dependent Failure Analysis (DFA)

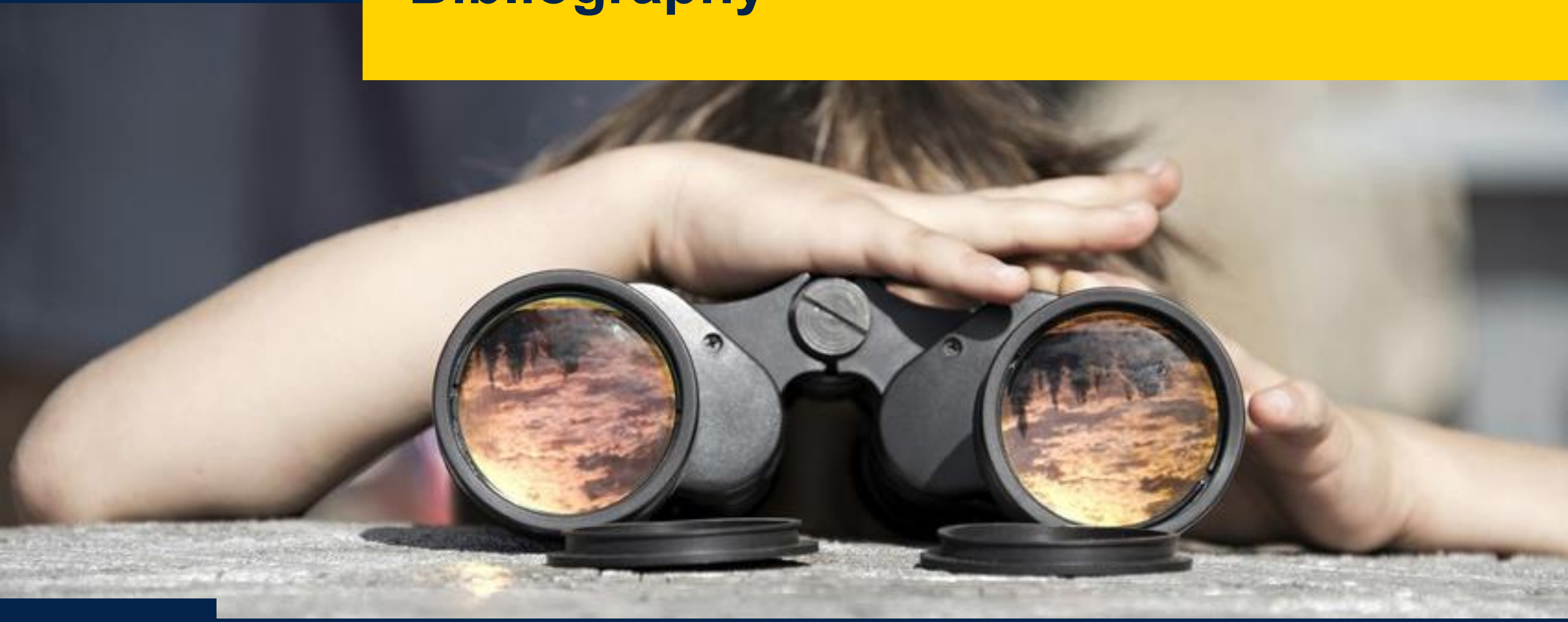
Purpose is to identify and analyze failures that are not independent but occur due to a common cause or dependency between components...

It is explored in detail mainly in ISO26262

It leverages on other safety analysis techniques (mainly FTA and sometimes FMEA), with focus on dependent failures finding.

Safety standards helps the research with specific guidance tables (typical topics to be analyzed in searching such DFA)

Bibliography



life.augmented

Reference documents

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

[R3]: ExoMars 2016 - Schiaparelli Anomaly Inquiry (ESA) downloaded from https://exploration.esa.int/web/mars/-/59176-exomars-2016-schiaparelli-anomaly-inquiry

[R4]: Fault Tree Handbook with Aerospace Applications - NASA Office of Safety and Mission Assurance, V 1.1 , 2002

[R5]: open FTA software can be found on the wed, e.g. https://www.fault-tree-analysis.com/free-fault-tree-analysis-software, or check for OpenFTA download

Thank you

life.augmented

life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni

Functional Safety Expert

STMicroelectronics

Lesson #5

Diagnostics (hw and sw), system evolution, hw/sw partitioning

Summary:

- **Safe State**
- **System evolution in time**
- **Mode of operations, PST, test frequency**
- **Elements on diagnostics, hw, sw, system**

life.augmented

De-rating

Hardware components must be operated at levels which are guaranteed by the design of the system to be well below the maximum specification ratings.

De-rating is the practice of ensuring that - under all normal operating circumstances - hardware components are operated well below their maximum stress levels – it can be defined as a safety margin.

IEC61508 recommend derating (2/3 factor) for hardware cpmponents

IAO13849-1 explicitly mention derating as one of economized additional techniques to lower the possibility of systematic failures (again 2/3 factor).

De-rating can play a relevant tole in guaranteeing that the assumption “failure rate = constant“is still valid.

Safe State

Safe State is formally defined in both main standards:

IEC61508-4: state of the EUC when safety is achieved

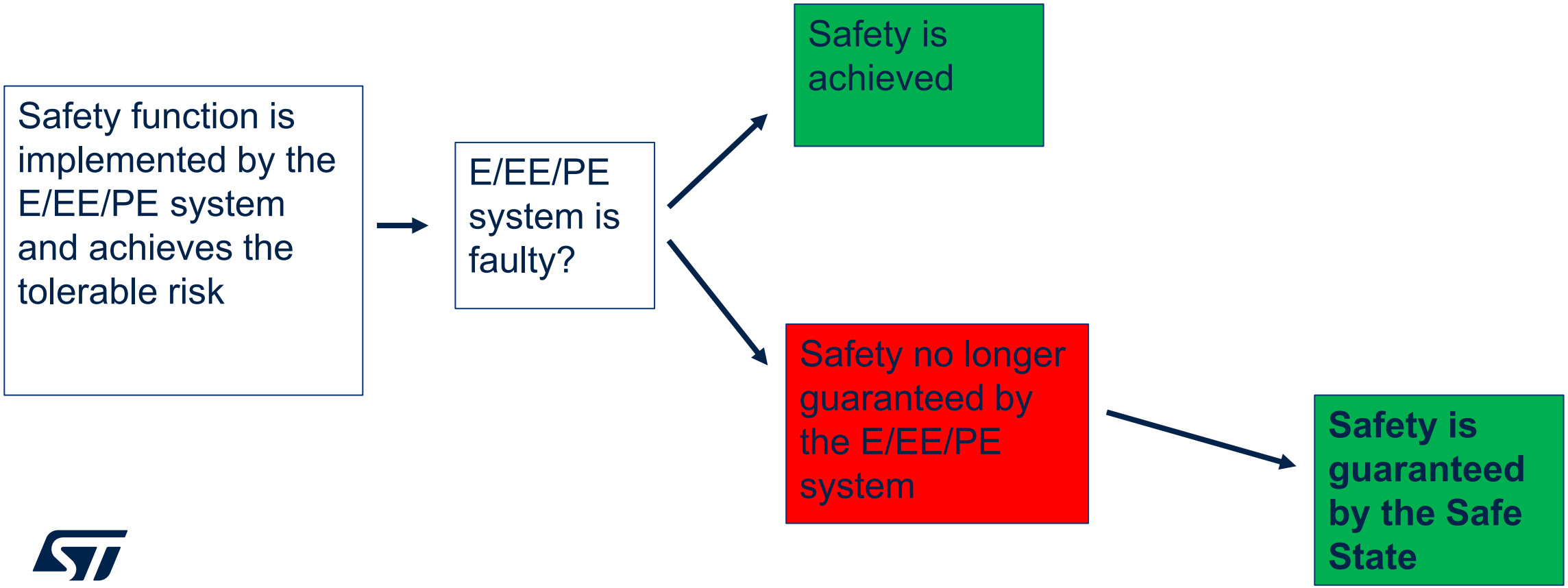
ISO26262-1: operating mode, in case of a failure, of an item without an unreasonable level of risk

In IEC61508 the system must be always in „Safe State“, either when it’s perfectly working or faulty.

In the common usage, “Safe State“ indicates the specific state where the system guarantees the safety in case of failure (ISO26262 bias) – usually, in a “degraded” mode.

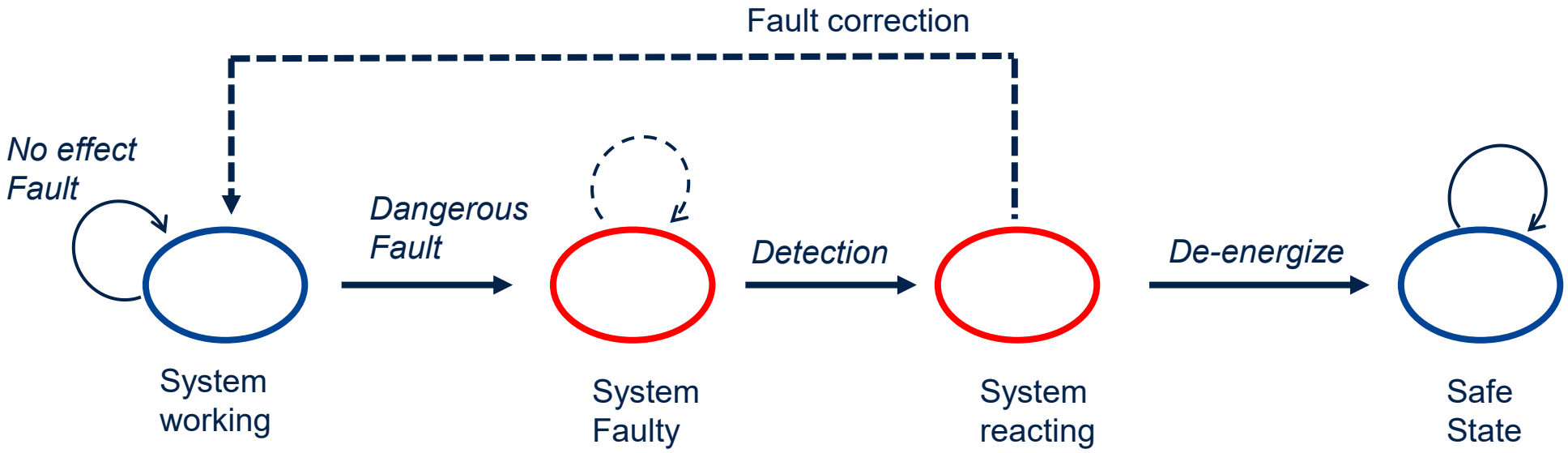
Safe State

Safe State definition is never generic, as it is strongly connected to the final application i.e. to the way the outputs/decisions are communicated/actuated (the safety function). As per safety function, safe state definition is at system level (a local safe state can be defined as well).

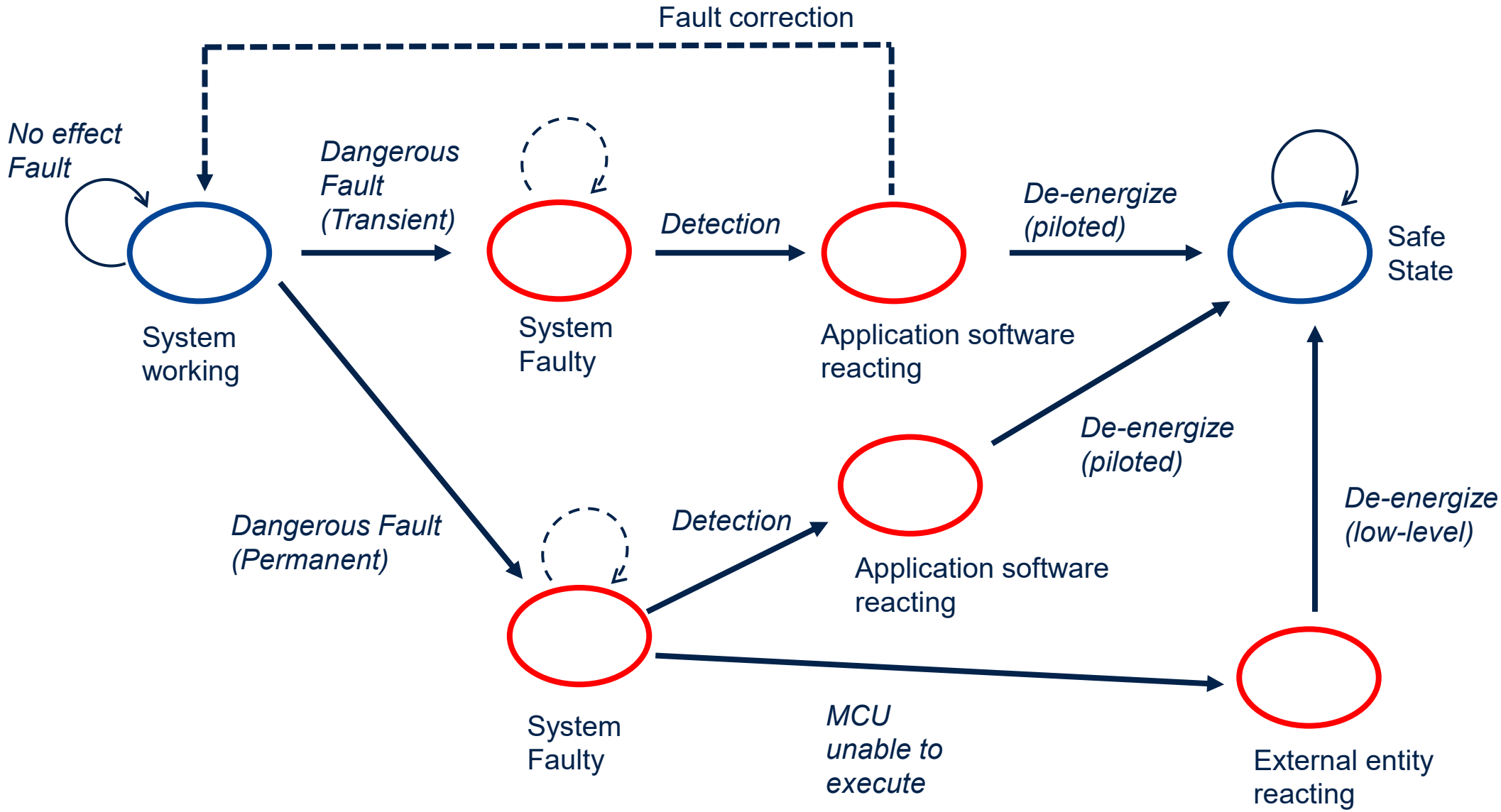


Safe State and system evolution (general)

Important: the Safe State should be always reachable - in an independent way from the actual failure affecting the system.

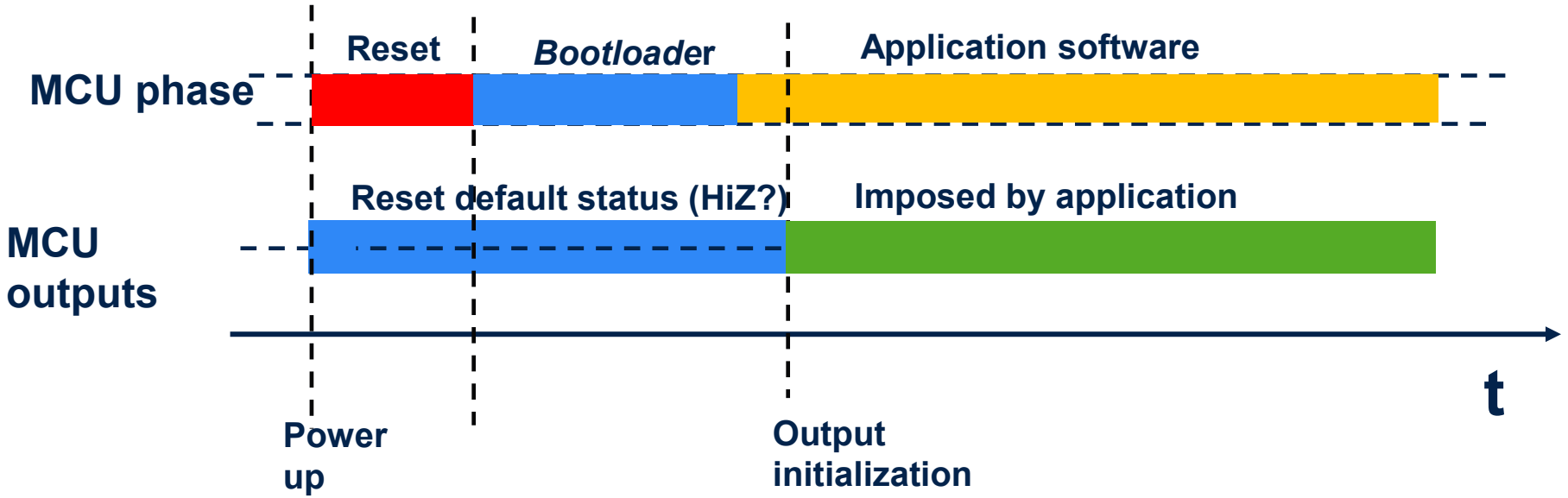


Safe State and system evolution (adding MCU and fault models)



Safe State and system boot

Important: the Safe State must be always guaranteed, even when no software execution is possible



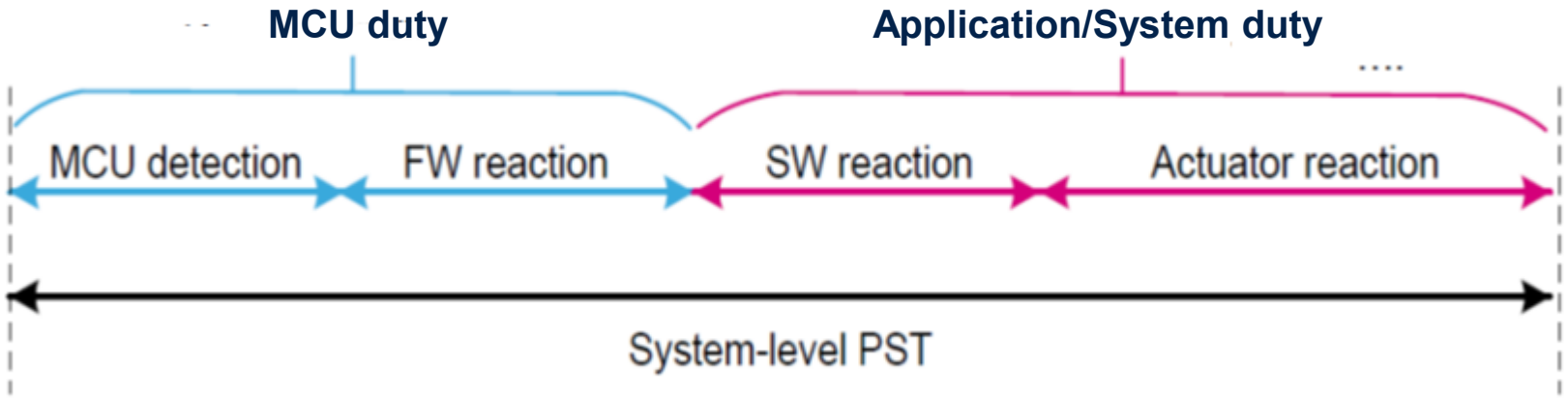
The startup phase (including possible bootloader execution, longer for flash-less devices) requires additional measures external to the MCU to guarantee the safe state

Faults occurring during a power-up phase may let the system hang-up in one of the initial phases.

life.augmented

Time sequence for PST limitation understanding

This is the typical time/causal sequence from the detection of a fault to the achievement of the Safe State.



Note that PST Process Safety Time is defined as the time between the rise of a dangerous failure and the moment a real hazard occurs. In CM systems, diagnostics must be able to intervene within the PST

Attention: not comprehensive of corner cases related to CPU hang up or impossibility to correctly execute software actions. Because of that, Safe State transition by external entities (e.g. a watchdog) is needed.

life.augmented

Mode of operations (IEC61508)

In IEC61508 the Mode of operation is related to which frequency the Safety Function is demanded; it drives the target metric (PFD/PFH) and testing frequency:

Low demand mode: safety function is only performed on demand, to transfer the EUC into a specified safe state, and where the frequency of demands is no greater than one per year

High demand mode: safety function is only performed on demand, to transfer the EUC into a specified safe state, and where the frequency of demands is greater than one per year

Continuous mode: where the safety function retains the EUC in a safe state as part of normal operation

LD → PFD Probability of Failure on Demand: probability

HD/CM → PFH (Probability of Failure per Hour: probability/time)

Mode of operations (IEC61508)

The frequency of periodic diagnostic execution depends on the Mode: DC can be claimed just for safety mechanisms executed within the limits specified here below.

- On LD systems, proof test concept apply (refer to related slide).
- HD systems: test frequency is linked to the frequency of safety function demands (100x faster). This allows software-based concepts.
- CM systems require that each periodic diagnostic is executed at least once per PST (Process Safety Time), introducing related concept

Mode of operations (IEC61508)

Examples of LD/HD/CM safety functions:

Mode	Safety Function	Description
LD	Emergency Shutdown System (ESD)	Shuts down the process safely in case of a hazardous event (e.g., gas leak, fire).
HD	Safety Interlock Systems	Frequently invoked to prevent unsafe operations (e.g., opening a valve only under safe conditions)
CM	Fire and Gas Detection System (continuous monitoring)	Continuously monitors for fire or gas presence and triggers alarms or shutdowns immediately upon detection

Note that in the same safety system it is possible to have coexisting safety functions with different Mode of operations (e.g. in a Fire systems, one CM SF for fire/smoke detection and one LD SF for sprinklers deployment)

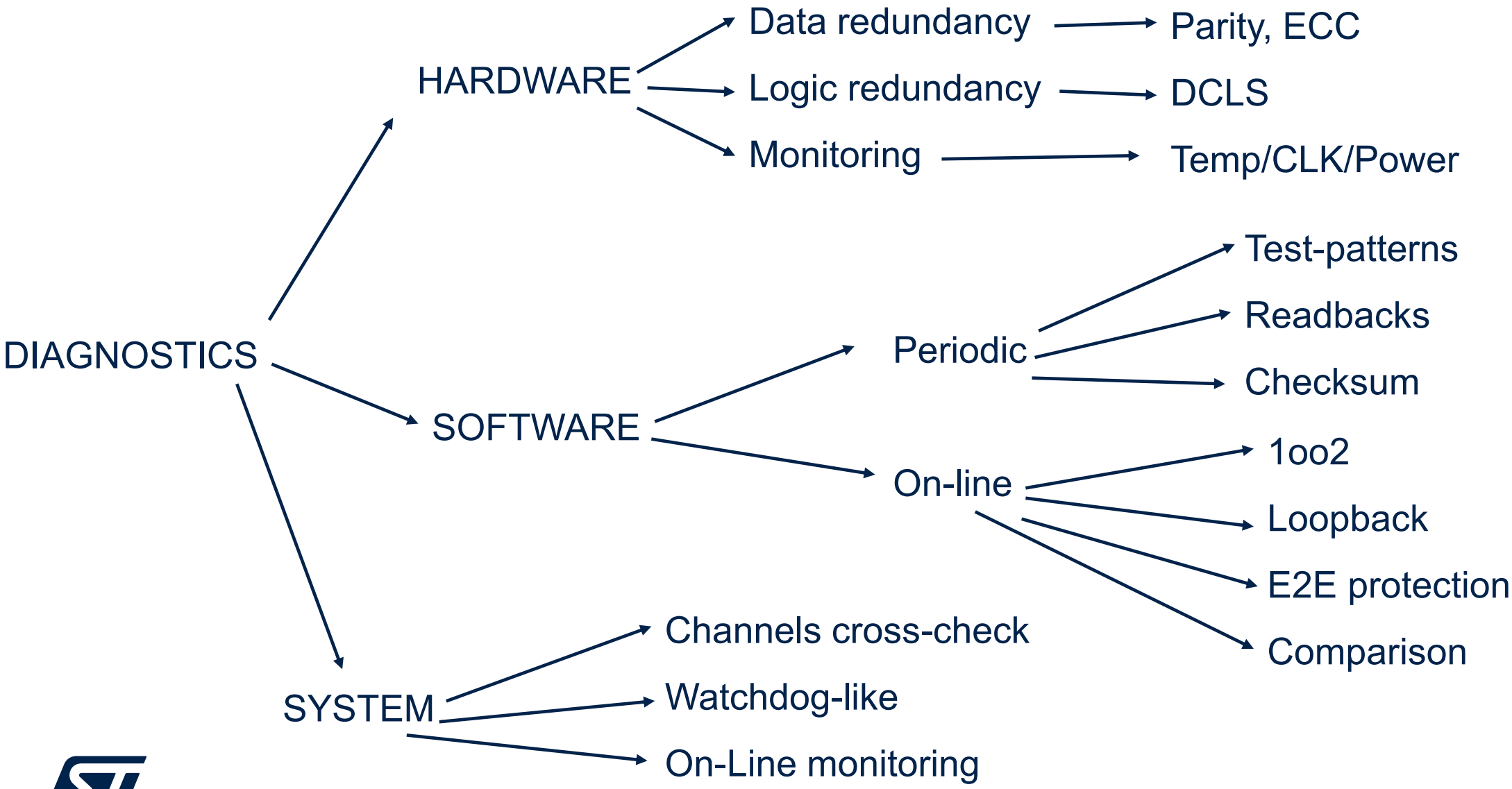
The proof test concept

Proof test is defined as a periodic test performed to detect dangerous hidden failures in a safety-related system, to sllow (if necessary) a repair which can restore the system to an “as new” condition or as close as practical to that condition.

Proof test is the main way to ensure safety on Low Demand systems, where PFD is the dominant metric. The periodicity of the test is imposed by the target SIL level and the device failure rate (the higher λDU is, the shorter is the interval)

Proof test can be applied to HD/CM systems as well with theintention to address “hidden” failures related to structures difficult to be tested during opeting time like diagnostic functions, error chain (reporting and reaction), partially corrected faults. Usually, the effect on PFH is negligible.

Safety mechanisms classification



life.augmented

Pro/Cons per categories

Safety mechanisms characteristics

There are common characteristics to be defined when dwealing with diagnostics/safety mechanisms

- ❑ Addressed fault model (permanent/transient/both?)
- ❑ Periodicity (continuous/on demand/periodic)
- ❑ Error reaction (message/flag/interrupt)
- ❑ Error correction (yes/no/partial)
- ❑ Test/multiple fault protection (listing alternative diagnostic(s) for faults preventing the correct functioning of the safety mechanisms itself
- ❑ Initialization/configuration (some diagnostics are always on, others may needs configuration by sw, etc)

Achieved DC – how to establish

There are multiple patterns to establish the achieved Diagnostic Coverage for a given safety mechanism

- Reference Tables from safety standards: many safety standards include reference table where for a set of high-level specified diagnostics a range/indication for achievable (*) DC is provided. Usually, enumerated values (High=99%, Medium-90%, Low=60%)
- Fault injection/simulation: the component is modelled inside a tool able to emulate the faults affecting the hardware, and the reaction capability of diagnostics. DC is computed on statistical way

(*) pay attention: “achievable” is not “achieved”. Accordingly, those values are considered the maximum achievable DC for such a diagnostics (!).

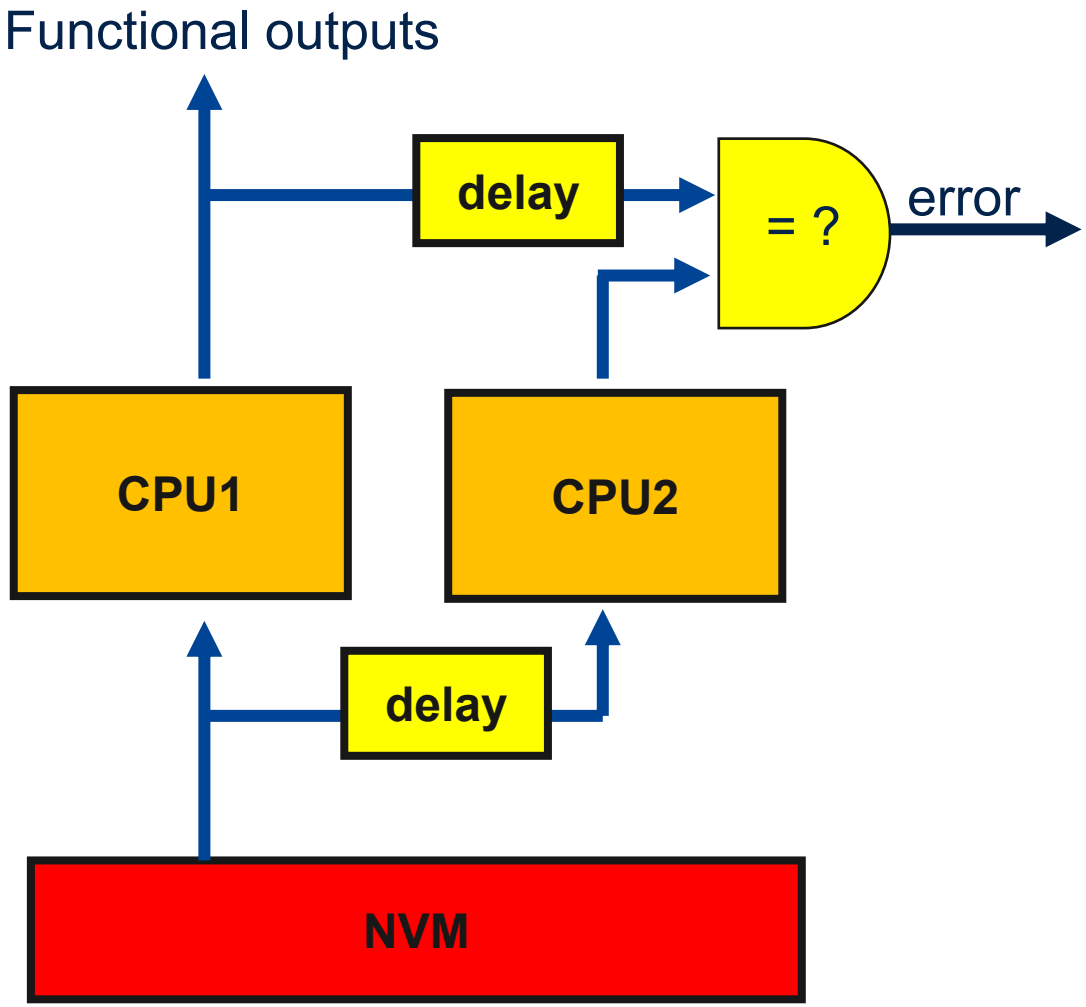
Dual Core Lock Step (DCLS)

PROS

- Fast fault detection
- Detects both permanent and transient failures
- High achieved DC
- Application independent

CONS

- Cost and complexity so available just on safety-ready devices
- Second CPU is just for monitoring, so HFT=0
- Still needs additional entities to manage failures preventing software execution



life.augmented

Parity bits

A parity bit is added to each word (multiple schemes are possible), enabling single bit error detection when data are read.

PROS

- Fast fault detection
- Detects both permanent and transient failures
- Best compromise between device cost and medium achieved DC
- Application independent
- No time penalties from end user perspectives

CONS

- Coverage guaranteed just on single bit failures (50% on dual, etc…)
- Achieved coverage is questionable because of differences between safety standard guidance
- Usually, check is done on reading → error accumulation must be managed (scrubbing)
- If address lines are not included, additional tests for address decoder needed

ECC

A multi-bit redundant code is added to each word (different schemes are possible), enabling single error correction and double error detection when data are read..

PROS

- Fast fault detection
- Allows single error correction so increasing system availability
- Detects both permanent and transient failures
- High achieved DC
- Application independent
- No time penalties from end user perspectives

CONS

- Usually, check is done on reading → error accumulation must be managed (scrubbing)
- Usually, correction is done just on data sent to CPU and not on cells → error still there
- If address lines are not included, additional tests for address decoder needed

Internal watchdog

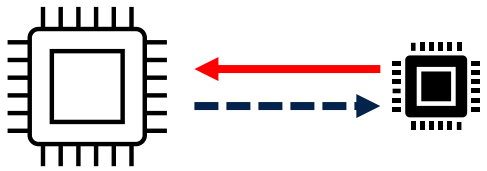
Forces a CPU reset when the required action from software (e.g. register write with a key) is not executed within the programmed period. Timing policy can be enforced by window requirement.

PROS

- Manages permanent of transient failures affecting correct software execution capability
- Contributes to systematic capability of the software by intercepting wrong control flow or timing

CONS

- Lack of hardware diversity as it shares with the CPU the same silicon substrate, and often power/clock as well
- Unable to completely manage failures leading so software execution inability
- Often overlapped by external watchdog as required by IEC61508-2, Table A.1/Table A.14.



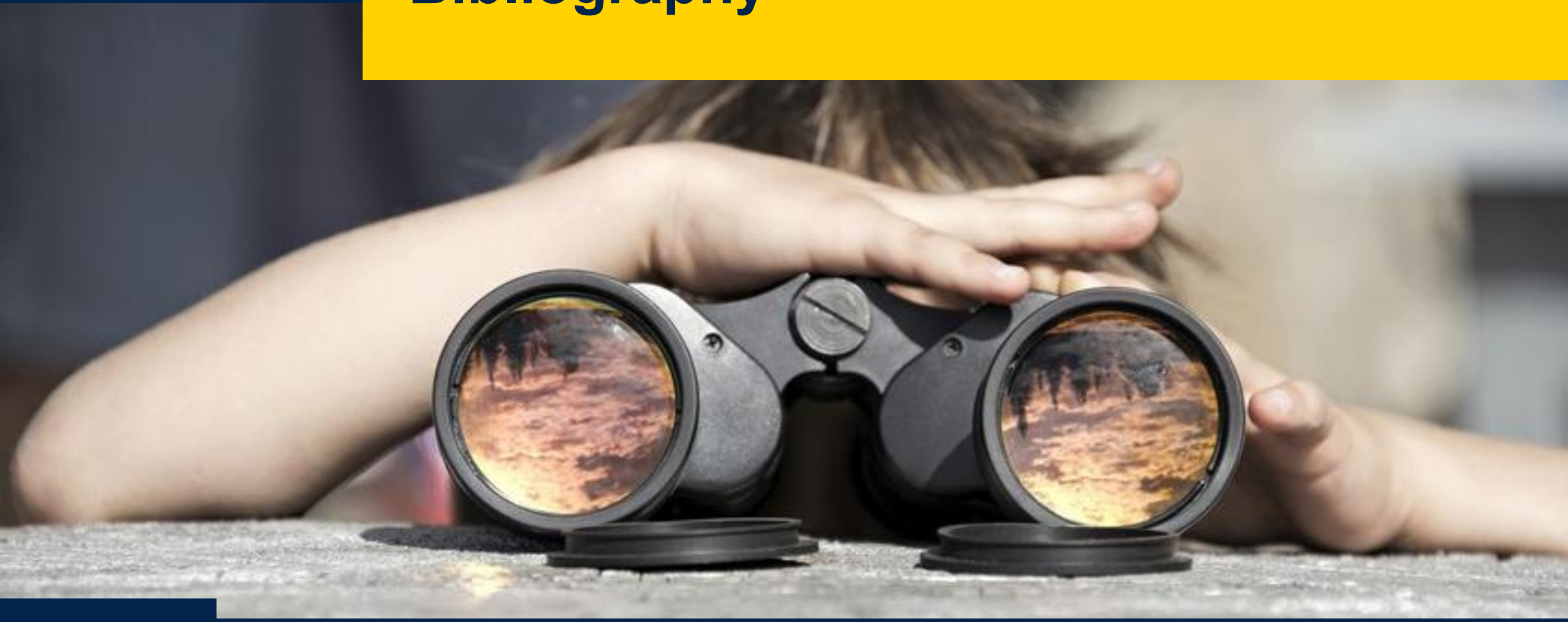
Useful references on safety mechanisms

Reference [6] lists in section “7 Brief Description of Diagnostics “ more than 100 different safety mechanisms defined in a automotive safety ASIL D microcontroller (TI). Check also “Appendix A Summary of Recommended Safety Feature Usage” where an exhaustive table provides a synoptic view of all characteristics for the listed diagnostics.

Reference [7] provides similar descriptions in section “3.6 Hardware and software diagnostics”, in this case an IEC 61508 wording is adopted across the document. The target is also in this case a MCU with intermediate safety level SIL 2.

Reference [8] offers a different perspective on a “simpler” device, a PMIC. Refer to section “5 TPS65919-Q1 Architecture Safety Mechanisms and Assumptions of Use” for a view of the very different set of dedicated diagnostics.

Bibliography



life.augmented

Reference documents 1/2

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

[R3]: ExoMars 2016 - Schiaparelli Anomaly Inquiry (ESA) downloaded from https://exploration.esa.int/web/mars/-/59176-exomars-2016-schiaparelli-anomaly-inquiry

[R4]: Fault Tree Handbook with Aerospace Applications - NASA Office of Safety and Mission Assurance, V 1.1 , 2002

[R5]: open FTA software can be found on the wed, e.g. https://www.fault-tree-analysis.com/free-fault-tree-analysis-software, or check for OpenFTA download

Reference documents 2/2

[R6]:Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM®-Based Safety Critical Microcontrollers

[R7]: UM2331- STM32H7 singlecore series safety manual STMicroelectronics – from https://www.st.com/en/embedded-software/x-cube-stl.html#documentation

[R8]: Safety Manual for TPS65919-Q1 Power Management Unit (PMU)

Thank you

life.augmented

life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni

Functional Safety Expert

STMicroelectronics

Lesson #6

Formal safety architectures (IEC61508, ISO13849) + ISO26262 methods (ASIL decomposition). Mapping on automotive and robotics use cases

Summary:

- **General concepts (HFT)**
- **IEC 61508 architectures (1oo1,1oo2,2oo2)**
- **ASIL decomposition, theory and examples**
- **Considerations on role of the software**

life.augmented

About “diversity” concept

Diversity is defined in IEC61508-4 “different means of performing a required function“

Diversity can be required in some circumstances (e.g. to use the composition rule SC+1 with redundant channels)

Diversity is the key asset to fight common cause failures (and to cope with their potential incomplete analysis)

Diversity is possible in hardware and software.

For software, also timing diversity can be achieved (same computation executed in different moments) providing protection against soft errors impacting the CPU

Diversity is an asset but also a cost (design time, hardware resources, memory space, complexity, increased verification, potential availability issues).

Diversity can be used as mitigation factor for tool failures (tool assessment procedure for T3 tools)

Hardware Fault Tolerance (HFT)

Hardware Fault Tolerance of N means that N+1 is the minimum number of faults that could cause a loss of the safety function.

To determine HFT no considerations can be done on other measure controlling the effect of faults (like diagnostics)

Some faults can be excluded from the considerations, under specific rationale based on their probability.

➔HFT is basically a property of subsystems, not of components.

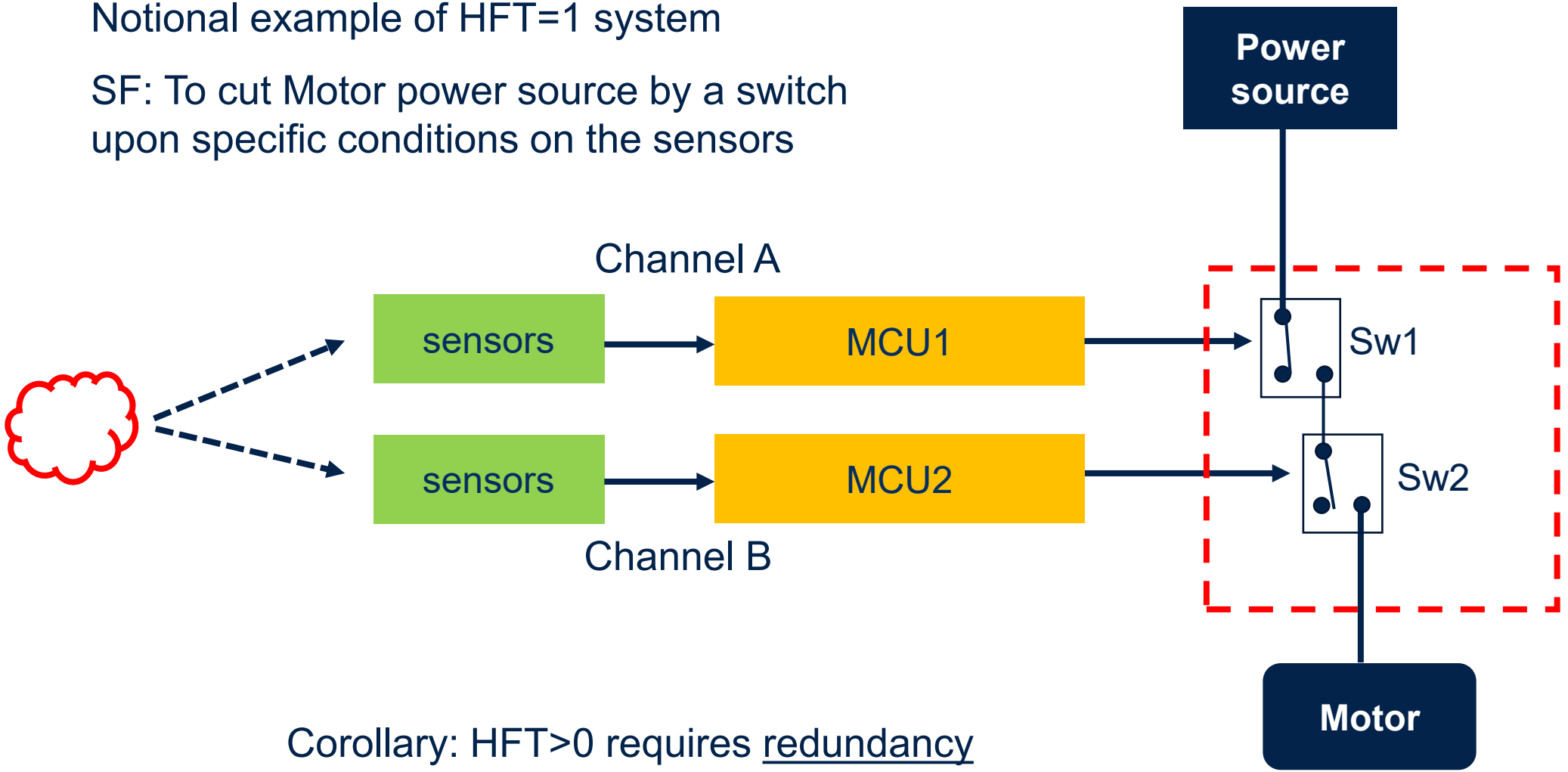
Remember: HFT is on of the entries to establish SFF targets for hw systems

Note: IEC 61508 includes special architecture requirements for ICs with on-chip redundancy

Hardware Fault Tolerance (HFT)

Notional example of HFT=1 system

SF: To cut Motor power source by a switch upon specific conditions on the sensors



Corollary: HFT>0 requires redundancy at a certain level

life.augmented

What is a safety architecture

Generally speaking, a good definition for an ***architecture*** is “representation of the structure of a system that allows identification of building blocks, their boundaries and interfaces, and includes the allocation of requirements to these building blocks. This is akind of gereal definition.

There is no exact definition of “safety architecture” in current safety standards. That can be inferred, looking to an IEC 61508 framework, this way:

The safety architecture defines the structure and organization of the safety-related system, including the safety functions, safety mechanisms, safe state(s), and their allocation to hardware and software elements to achieve the required safety integrity.

IEC 61508: 1oo1

1oo1 is the single channel architecture (PE = Processing Element)

It’s the most simple but also the most challenging / NO benefits from architecture scheme! Everything is solved inside the simple structure.

DC must be guaranteed by intrinsic diagnostic (HW/SW)

HFT = 0

Safe state transition could be complex

No diversity in hardware is possible → hardware SC to be reached at component level

Diversity in software poorly possible → software SC to be reached at component level

life.augmented

IEC 61508: 1oo1 - formulas

Low Demand mode:

$$PFD_G = (\lambda_{DU} + \lambda_{DD}) t_{CE} \qquad t_{CE} = \frac{\lambda_{DU}}{\lambda_D}\left(\frac{T_1}{2} + MRT\right) + \frac{\lambda_{DD}}{\lambda_D} MTTR$$

High Demand/Continuous Mode:

If it is assumed that the safety system puts the EUC into a safe state on detection of any failure, for a 1oo1 architecture the following is obtained

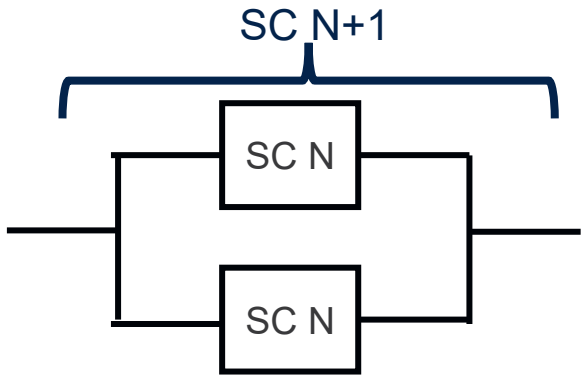
$$PFH_G = \lambda_{DU}$$

life.augmented

IEC 61508: 1oo2

Multiple types of diversity in hardware are possible → hardware SC can be easier thanks to N+N = N+1 rule

Diversity in software is possible → software SC is easier thanks to N+N = N+1 rule



Attention:

- Independence between elements is required
- Allowed only once (no more cascading compositions)

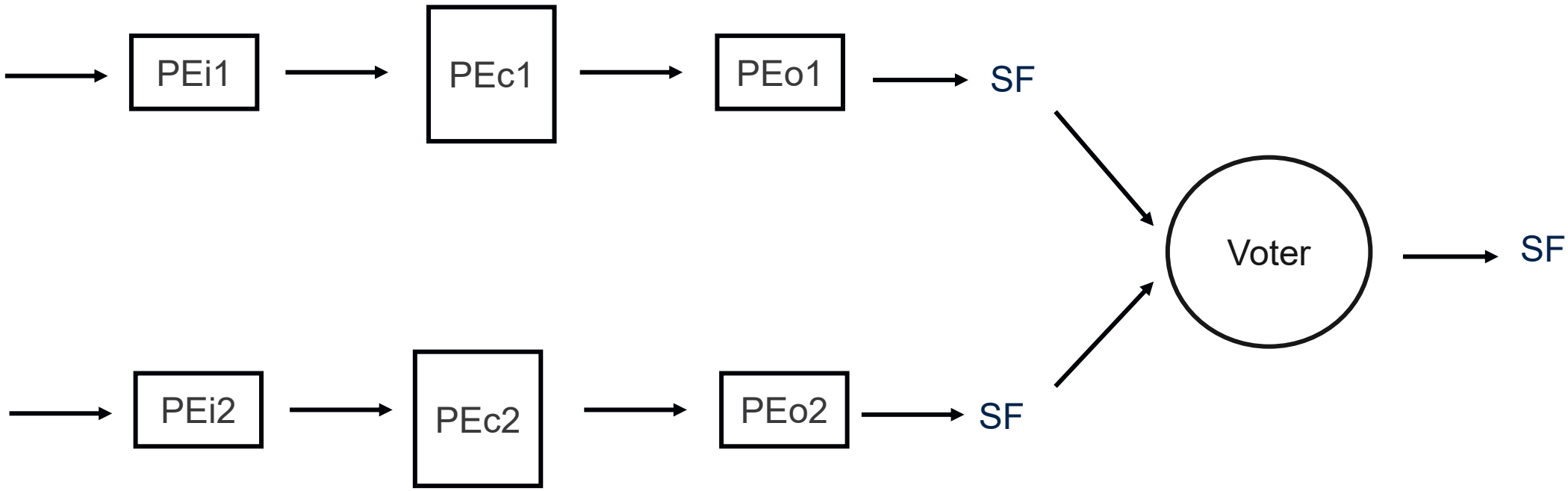
BUT still some complications

Common cause failures to be explored (impact on-safety metrics!)

Voter must be HFT = 1 intrinsically

life.augmented

IEC 61508: 1oo2



Note: be aware that cross-collaboration between the two MCUs may lead to increase of the penalty factors β and βD (common cause failures between MCUs

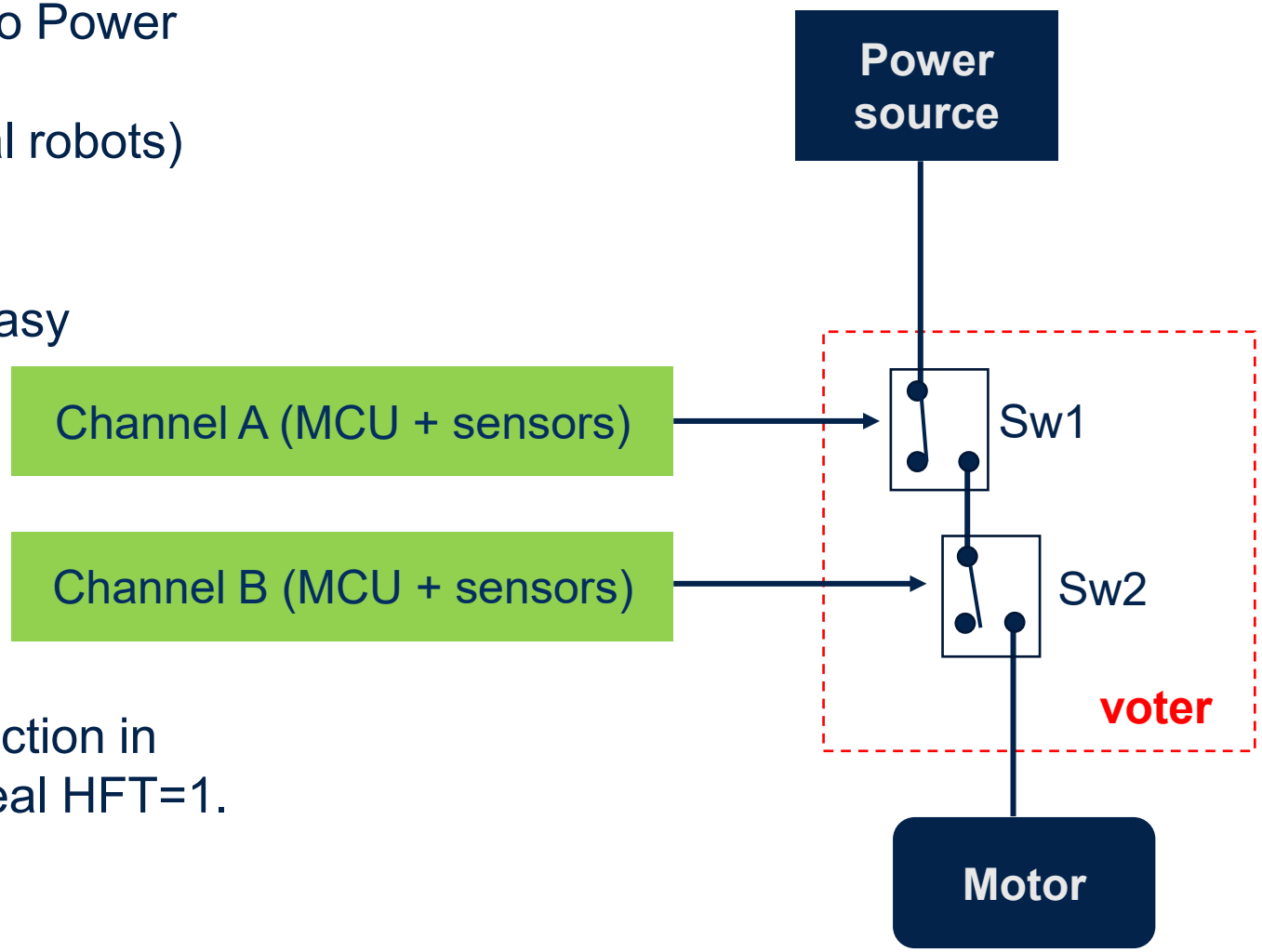
life.augmented

1oo2 notional example

Safety function: “Open Motor connection to Power source on specific input signals condition” (application e.g. active barrier for industrial robots)

Safe state: power connection OPEN

The nature of this safety function allows easy implementation for voter



Note: switch structure forces the disconnection in case of failure for Voter local supply For real HFT=1. structure of the voter, switches need to be implemented in redundant way

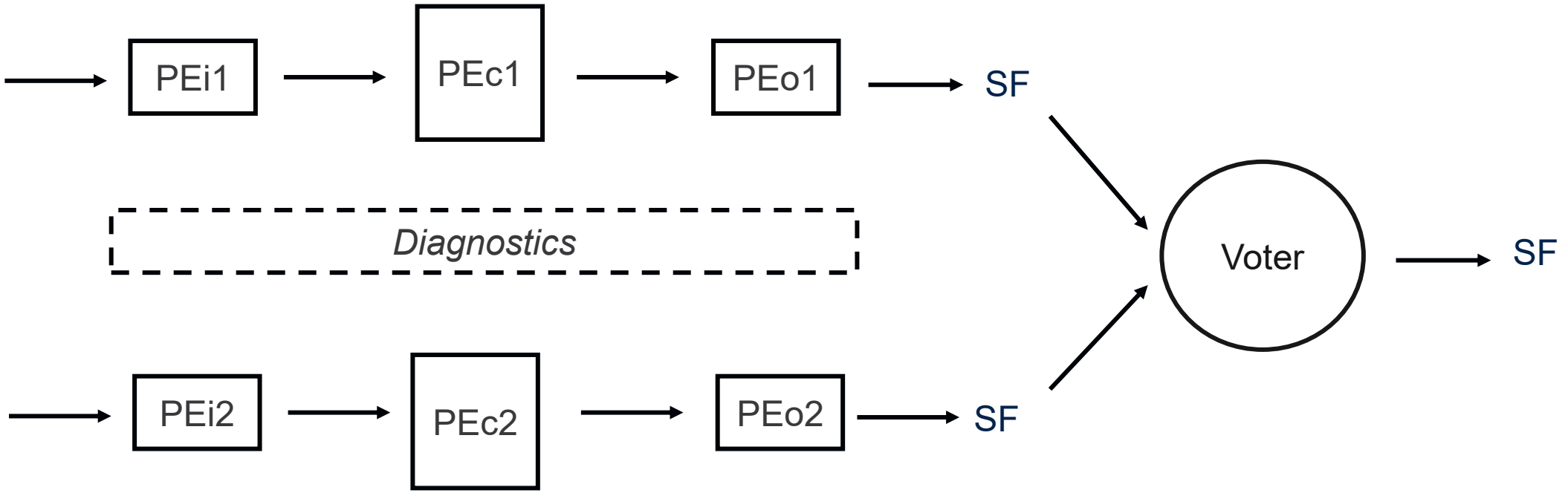
life.augmented

IEC 61508: 2oo2

2oo2 consists of two channels connected in parallel so that both channels need to demand the safety function before it can take place.

DC must be guaranteed by intrinsic diagnostic for each channel. Diagnostics would report faults but not changing voting.

The scheme is HFT = 0. PFH = 2λDU



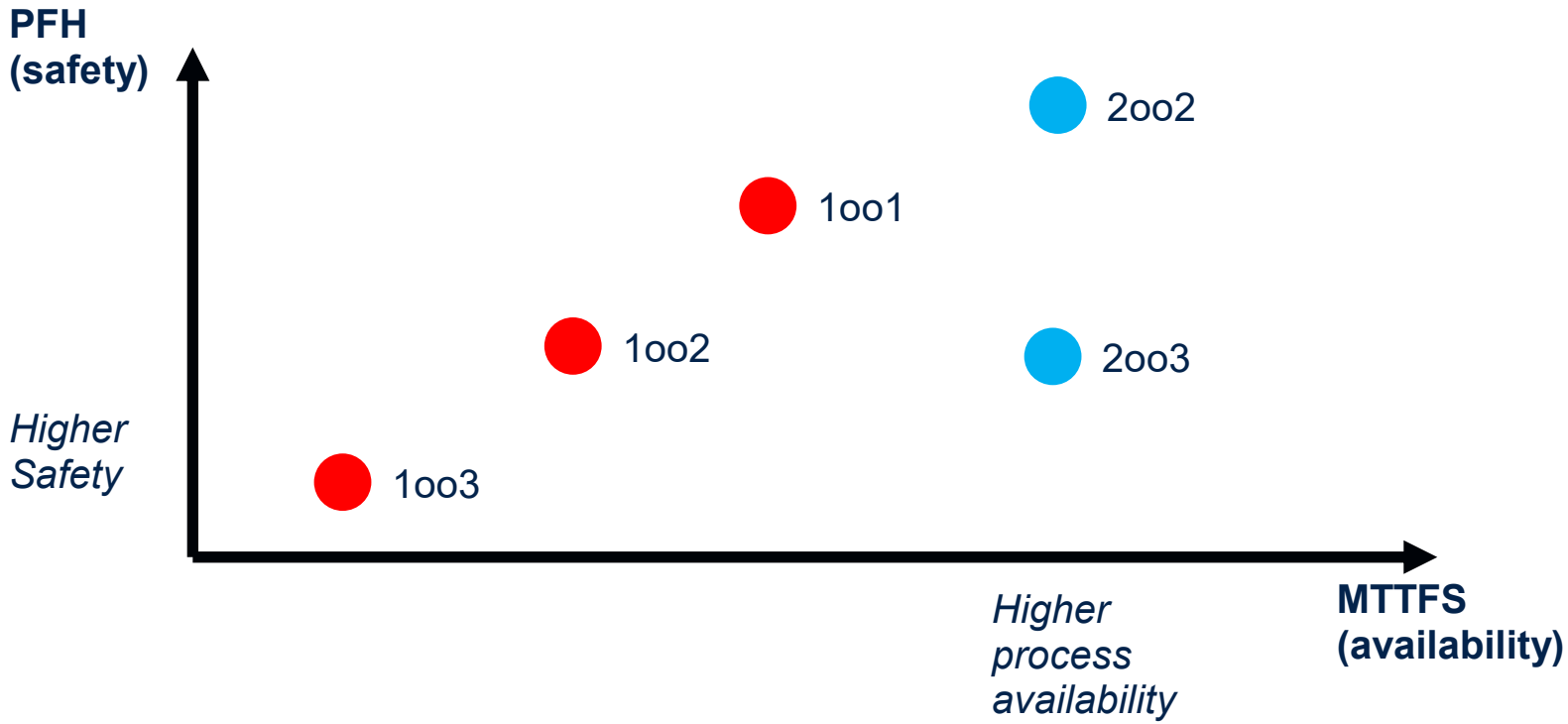
life.augmented

1oo2 vs 2oo2 vs 2oo3

1oo2: priority is on safety (lower PFH)

2oo2: priority is on availability (higher PFH but less false positive safe state transitions)

2oo3: keeps the advantages of two above schemes, at higher cost



Notes: MTFS = Mean Time to Fail Spurious; above graph is qualitative only

ASIL decomposition

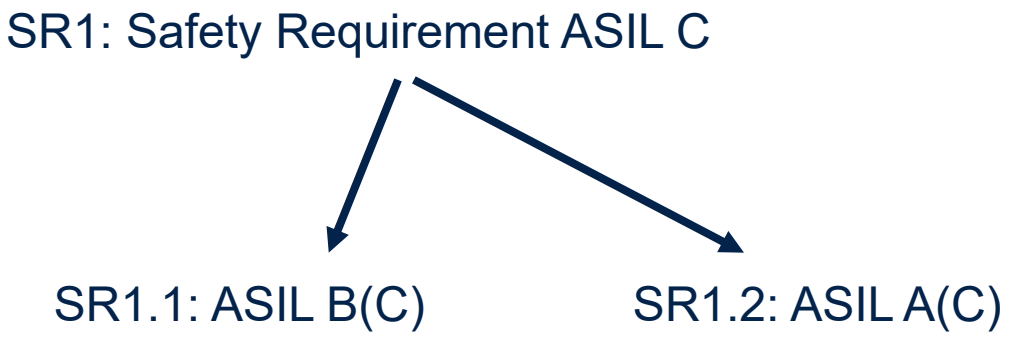
ASIL decomposition is described in ISO 26262-9:2018 clause 5.

The main purpose of implementing ASIL decomposition is to lower the ASIL level.

A safety requirement is decomposed into redundant safety requirements, which are then allocated to sufficiently independent elements.

The ASIL decomposition process must adhere to the scheme given by Table 1 of ISO 26262-9:2018.

D	ASIL D(D) + QM(D)
	ASIL C(D) + ASIL A(D)
	ASIL B(D) + ASIL B(D)
C	ASIL C(C) + QM(C)
	ASIL B(C) + ASIL A(C)
B	ASIL B(B) + QM(B)
	ASIL A(B) + ASIL A(B)
A	ASIL A(A) + QM(A)



life.augmented

ASIL decomposition – general rules

Homogenous redundancy (e.g., by duplicated device or software) is, in general, not sufficient for reducing the ASIL due to the lack of independence between the elements

Safety goals cannot be decomposed

The decomposed requirements must independently meet the original requirement (definition of redundancy)

ASIL decomposition may be applied more than once (!) (big difference with combination of elements in IEC 61508)

What is the bet: the decomposed safety requirements sre simpler and/orcheaper in terms of implementation. So: ASIL decomposition is not always a winning solution…

ASIL decomposition – general rules

(INVARIANTS)

The resulting decomposed ASIL must be expanded with the original ASIL before decomposition, e.g., ASIL A(C)

The functional safety assessment of the item is defined by the original ASIL

The independence of decomposed elements shall be evaluated by analysis of dependent failures

Requirements for test and integration are defined by the ASIL of the relevant integration level

Target values for HW-metrics are defined by the original ASIL at the item level

Safety related software vs non-safety related

In articulated architectures, software is often part of the safety concept.

Coexistence between safety related software and non-safety related software may happen in microcontroller-based systems (often for cost reasons e.g. reuse of open source sw for non-safety related tasks).

Main issue is separation i.e. to guarantee that non-safety related software can’t interfere with the safety function:

- Spatial interference: memory overwrite, peripherals access etc
- Temporal interference: determinism violation, resource occupation, jitter, interrupt masking etc.

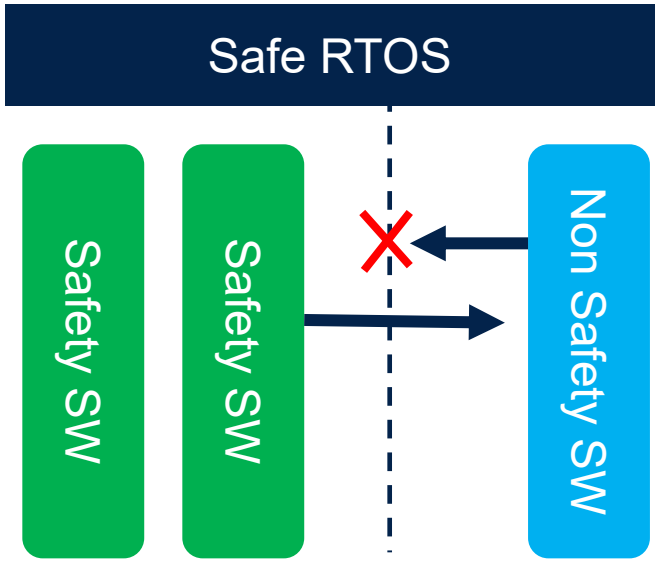
Hardware segregation can be difficult as typical microcontroller don‘t have complex MMUs (often just basic MPU is available)

Safety related software vs non-safety related

Possible vectors for coexistence issue solution:

Non-safety related software is simple: to repatriated it inside the certified V-model for application software could be viable

Non-safety software is complex: software segregation by a Safe RTOS could guarantee a) isolation of the potential spatial interferences b) determinism of the safety software regardless the non-safety one. Overhead is cost and avauilability of the Safe RTOS.



life.augmented

Backup slides

life.augmented

IEC 61508: 1oo2 - formulas

Low Demand Mode:

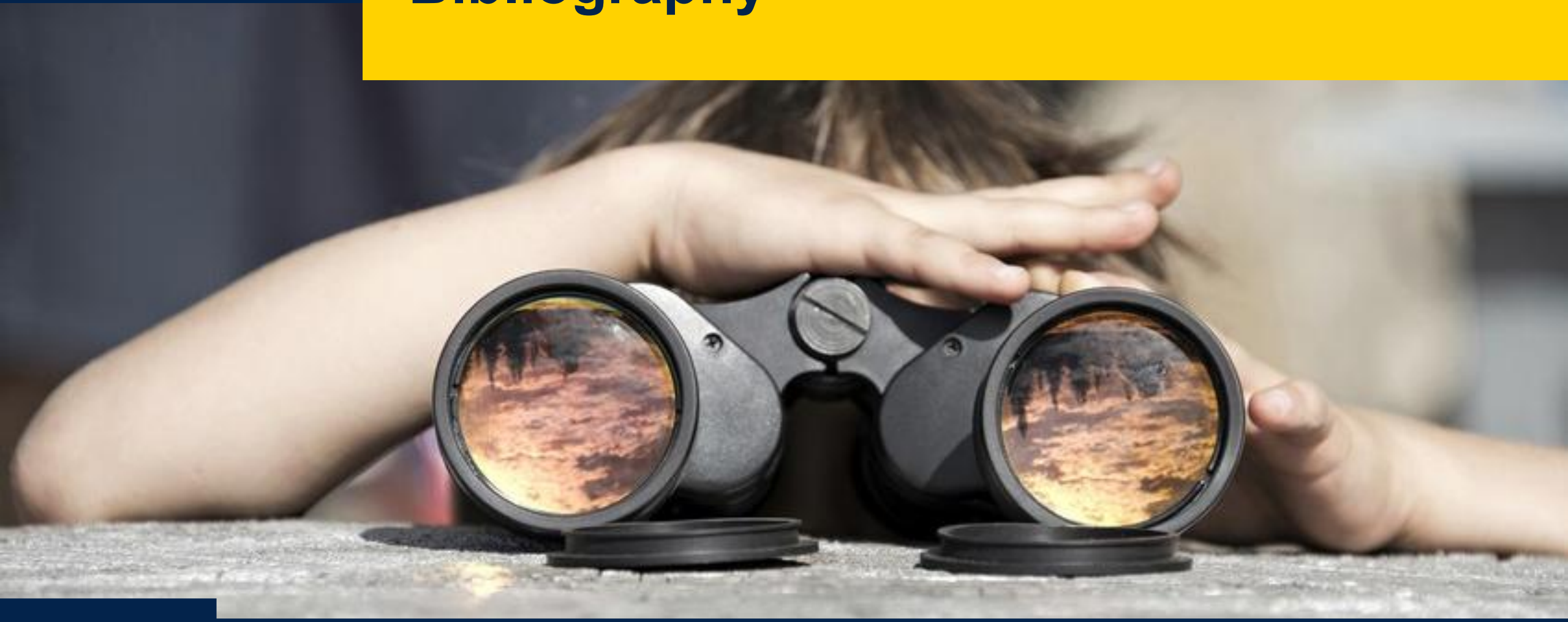
$$PFD_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)^2 t_{CE} t_{GE} + \beta_D \lambda_{DD} MTTR + \beta\lambda_{DU}\left(\frac{T_1}{2} + MRT\right)$$

High Demand/Continuous Mode:

$$PFH_G = 2\left((1-\beta_D)\lambda_{DD} + (1-\beta)\lambda_{DU}\right)(1-\beta)\lambda_{DU} t_{CE} + \beta\lambda_{DU}$$

life.augmented

Bibliography



life.augmented

Reference documents 1/2

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

[R3]: ExoMars 2016 - Schiaparelli Anomaly Inquiry (ESA) downloaded from https://exploration.esa.int/web/mars/-/59176-exomars-2016-schiaparelli-anomaly-inquiry

[R4]: Fault Tree Handbook with Aerospace Applications - NASA Office of Safety and Mission Assurance, V 1.1 , 2002

[R5]: open FTA software can be found on the wed, e.g. https://www.fault-tree-analysis.com/free-fault-tree-analysis-software, or check for OpenFTA download

Reference documents 2/2

[R6]:Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM®-Based Safety Critical Microcontrollers

[R7]: UM2331- STM32H7 singlecore series safety manual STMicroelectronics – from https://www.st.com/en/embedded-software/x-cube-stl.html#documentation

[R8]: Safety Manual for TPS65919-Q1 Power Management Unit (PMU)

Thank you

life.augmented

life.augmented

Functional Safety in Electronic Systems: Principles and Applications

Alessandro Bastoni

Functional Safety Expert

STMicroelectronics

Lesson #7

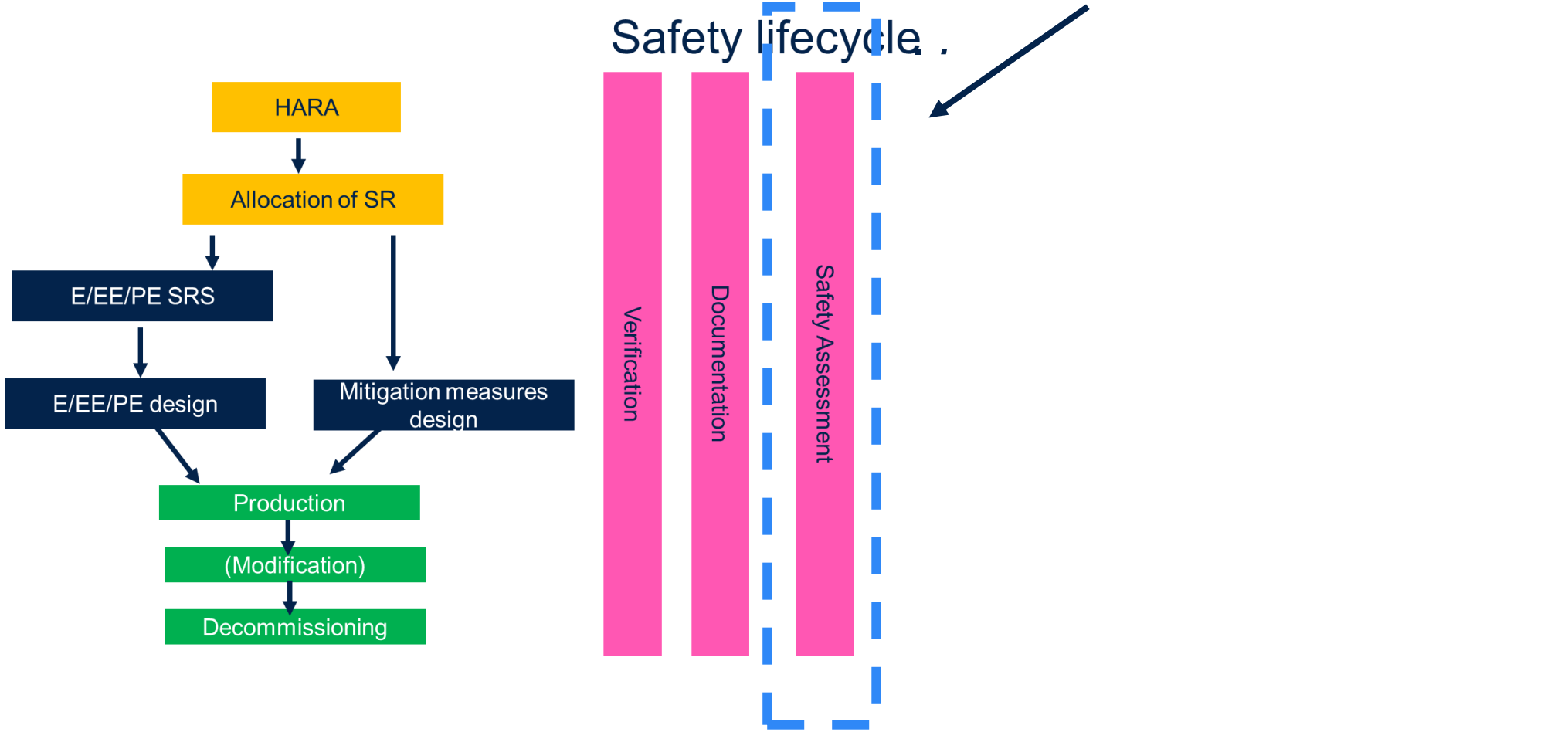
Miscellaneous in safety process

Summary:

- **Functional Safety Assessment, Independency**
- **Safety Case**
- **Certification process**

life.augmented

Recap on safety lifecycle



Functional Safety Assessment

What it is: IEC61508-4: 3.8.3: “investigation, based on evidence, to judge the functional safety achieved by one or more E/E/PE safety-related systems and/or other risk reduction measures”

Who will do it: the Assessor (person, persons or organization that performs the functional safety assessment in order to arrive at a judgement on the functional safety achieved by the E/E/PE safety-related systems and other risk reduction measures) required to be independent as per below table.

Minimum level of independence	Safety integrity level / Systematic capability			
	1	2	3	4
Independent person	Allowed	Allowed on consolidated and simple designs	Insufficient	Insufficient
Independent department		Allowed on novel/complex designs/technologies	Allowed on consolidated and simple designs	Insufficient
Independent organization			Allowed on novel/complex designs/technologies	Allowed

Independency

life.augmented

About independency

Independency criteria are specified in IEC61508-4, 3.8.11-13:

- independent person: person who is separate and distinct from the activities which take place during the specific activity, with not direct responsibility.
- independent department: department that is separate and distinct from the departments responsible for the activities
- independent organization: organization that is separate and distinct, by management and other resources, from the organizations responsible for the activities

When an independent organization is required, even for large organization like semiconductor manufacturers it is common to rely on Competent Bodies/Certification Agency (e.g. TÜEV, UL, etc.) – in that case the independency is clearly achieved and accepted. .

The Safety Case

The Safety Case is perfectly described in ISO26262:10, 5.3.1

The purpose of a safety case is to provide a clear, comprehensive and defensible argument, supported by evidence, that an item is free from unreasonable risk when operated in an intended context

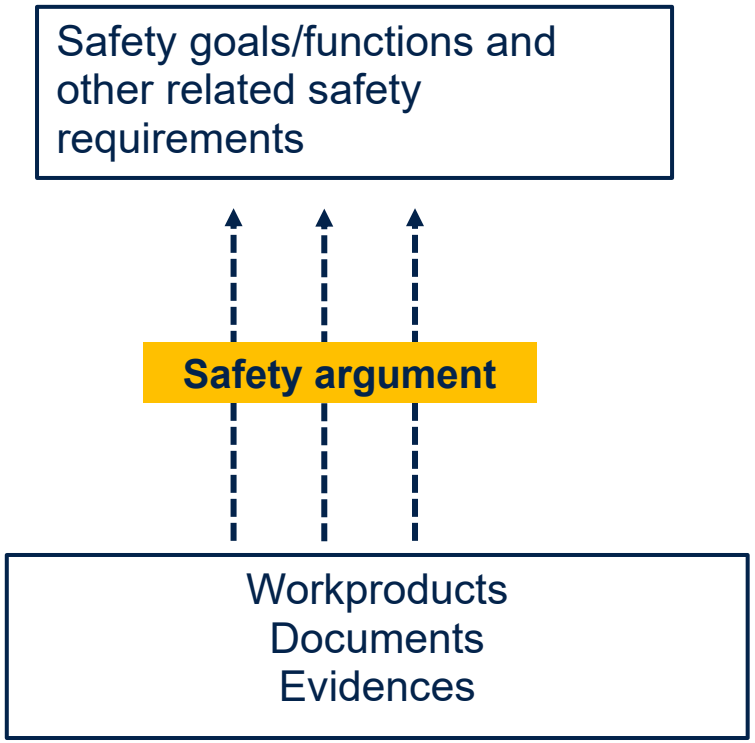
Principal elements:

the safety goals and related safety requirements

the collection of safety arguments (related to product or process)

the ISO 26262 series of standards work products (the evidence)

The description is applicable also for IEC61508 framework (where safety case isn't formally defined)



life.augmented

How to build a Safety Case

The perfect way to build a Safety Case is to follow the ISO26262 description with a formal implementation (in other words, by using formal languages e.g. Goal Structured Network).

The greatest benefit would be to link the description with the tool used to formally manage the requirements.

The approach is often found difficult, so in industry the tendence is to build the Safety Case by “narrative text“ (in the form of a Safety Report).

Narrative text could be confusing as there’s a risk to loose the boundaries between claims and related rationale. So, it’s recommended to formally keep the causal chain

Claims↔ Arguments ↔ Evidences

Note: the presence of a narrative text approach in third parties documentation (Safety Manuals, etc.) usually requires an additional effort to enable a more structured organization for the included requirements.

Certifications zoo in a nutshell

Certification: is one of the most ambiguous (and abused) wordings in functional safety, as it's used to describe multiple, different situations. Some examples:

- Pre-certified: software developed according to a certified V-model. The certification and related claims addresses the generic development flow followed by the organization, and the compliance of each specific software must be evaluated on the basis of the provided artifacts/documents
- Pre-certified: a T2 or T3 tool for which an off-line support tool analysis according to IEC61508-3m 7.4.4 has been executed and independently reviewed, but complex actions on end user side still exist
- Pre-certified: a hardware or software item formally certified vs specific claims, used as part of a whole solution integrating it (and not capable to magically extend the claims to the whole system)

.

The SIL certificate

The SIL certificate is a functional safety certification demonstrating that a product or process meets the IEC 61508 international standards.

It is issued by an independent third party to ensure compliance with IEC 61508-1 requirements for independence, especially for achieving the highest SIL levels 3 and 4, which mandate involvement from a separate division or independent body.

The certificate is self-supporting, providing all necessary information for integration into a safety system. Alongside the certified item's safety manual, it includes instructions for proper use to guarantee the declared SIL level is maintained

The certificate also contains a registered trademark with the identification ID of the certified product that must be affixed to the product to allow traceability and clear differentiation between SIL and non-SIL products. Certifixctes are usually stored I public-available database on the web.…

Different types of SIL certificate

Certificates can be issued in one of the following 3 types:

- **SIL type certificate**: valid for a whole type of product, the certification process is developed starting from the analysis of a prototype which, once certified, will be suitable to be mass-produced. The certificate is therefore valid for an unlimited number of products, when identical in every aspect to the validated prototype. The SIL type certificate has a duration that is defined by the certification body based on the complexity of the object (e..g limited in time, or subject to yearly confirmation). Example: safety Microcontrollers.
- **Single product SIL certificate**: limited to the device expressly covered by the certificate, this type is frequent for non-series production, custom, and project assemblies. The certificate expressly states the serial number or an unique identifier of the addressed product. Example: a factory.
- **Process SIL certificate**: states the compliance with the international standard IEC 61508 of the implementation process of one or more phases of the Safety Lifecycle/ Example: checking the compliance of processes and procedures with regulatory requirements.

Main information in a certificate

IDENTIFICATION OF THE CERTIFIED PRODUCT This isn't trivial. Always pay attention to what actually is the boundary of the certification (a single product? A line of products? etc.)

REFERENCE STANDARD This is the basis used for the certification, important for the understanding of the real scope (e.g. IEC61508:2010 ≠ IEC61508-3:2010)

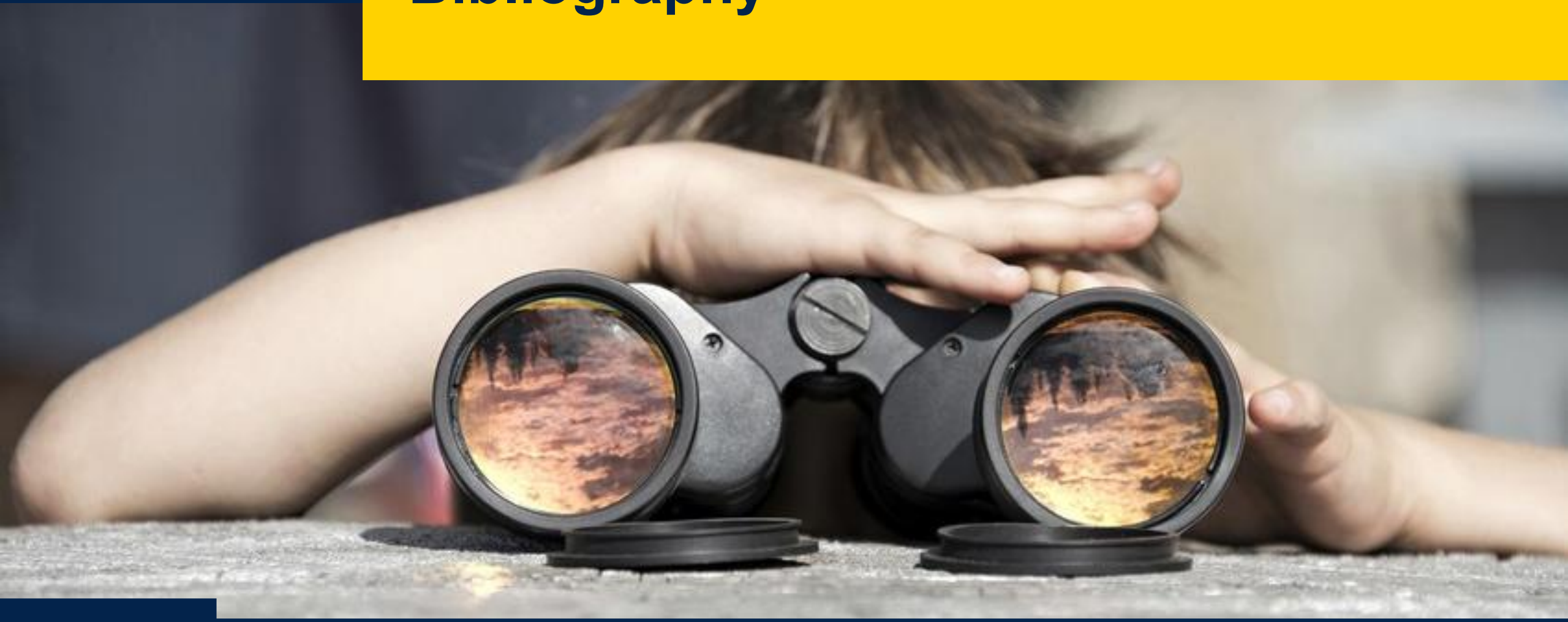
CLAIM The most important information. What is the actual claim done in the certificate? Pay attention to generic statements (e.g. "compliant to IEC 61508"… really all 7 parts???) with no clear boundaries on the achieved capability Always remember: HRF and SC have deterministic targets

ADDITIONAL INFORMATION Very different from one certificate to other. They can recommend complying with additional requirements included in a Safety Manual, or a Safety Report. Pay attention to sentences like „the Report is integral part of this certificate" – immediately inspect the Report in such a case

Backup slides

life.augmented

Bibliography



life.augmented

Reference documents 1/2

[R1]: Microelectronics Reliability:Physics-of-Failure Based Modeling and Lifetime Evaluation - Jet Propulsion LaboratoryCalifornia Institute of Technology Pasadena, California

[R2]: : Semiconductor Reliability Handbook – Renesas Electronics, Rev.2.50 Jan. 2017

[R3]: ExoMars 2016 - Schiaparelli Anomaly Inquiry (ESA) downloaded from https://exploration.esa.int/web/mars/-/59176-exomars-2016-schiaparelli-anomaly-inquiry

[R4]: Fault Tree Handbook with Aerospace Applications - NASA Office of Safety and Mission Assurance, V 1.1 , 2002

[R5]: open FTA software can be found on the wed, e.g. https://www.fault-tree-analysis.com/free-fault-tree-analysis-software, or check for OpenFTA download

Reference documents 2/2

[R6]:Safety Manual for TMS570LS31x and TMS570LS21x Hercules™ ARM®-Based Safety Critical Microcontrollers

[R7]: UM2331- STM32H7 singlecore series safety manual STMicroelectronics – from https://www.st.com/en/embedded-software/x-cube-stl.html#documentation

[R8]: Safety Manual for TPS65919-Q1 Power Management Unit (PMU)

Thank you

life.augmented

Functional Safety in Electronic Systems: Principles and Applications

GLOSSARY

Rev 1.0

ALARP As Low As Reasonably Practicable

ASIC Application Specific Integrated Circuit

ASIL Automotive Safety Integrity Level

BFR Base Failure Rate

BIST Built-In Self-Test

CCF Common Cause Failure

CM Common Mode

COTS Commercial Off-The-Shelf

CPU Central Processing Unit

CRC Cyclic Redundancy Check

DC Diagnostic Coverage

DFA Dependent Failure Analysis

DTI Diagnostic Test Interval

ECC Error Correction Code

EMC ElectroMagnetic Compatibility

EMI ElectroMagnetic Interference

E/E/PE Electrical/Electronic/Programmable Electronic

EEPROM Electrically Erasable Programmable Read-Only Memory

ETA Event Tree Analysis

EUC Equipment Under Control

FHTI Fault Handling Time Interval

FIT Failures In Time

FMEA Failure Modes and Effects Analysis

FMEDA Failure Modes, Effects, and Diagnostic Analysis

FPGA Field Programmable Gate Array

FSR Functional Safety Requirements

FTA Fault Tree Analysis

FTTI Fault Tolerant Time Interval

HARA Hazard Analysis and Risk Assessment

HAZOP HAZard and OPerability analysis

HD High Demand

HFT Hardware Fault Tolerance

HMI Human-Machine Interface

HW Hardware

IEC International Electrotechnical Commission

ISO International Organization for Standardization

LD Low Demand

LFM Latent-Fault Metric

MooN M out of N channel architecture

MRT Mean Repair Time

MTBF Mean Time Between Failures

MTTF Mean Time To Failure

MTTR Mean Time To Repair

OEM Original Equipment Manufacturer

PE Programmable Electronic

PFD Probability of Failure on Demand

PFH Probability of Dangerous Failure per Hour

PL Performance Level

PLA Programmable Logic Array

PMHF Probabilistic Metric for Hardware Failure

PST Proof Test

QM Quality Management

RAM Random Access Memory

RF Residual Fault

ROM Read-Only Memory

RPN Risk Priority Number

RTL Register Transfer Level

RTOS Real Time Operating System

SFF Safe Failure Fraction

SIL Safety Integrity Level

SOTIF Safety Of The Intended Function

SRS Safety Requirements Specification

SW Software

TSR Technical Safety Requirements

UML Unified Modeling Language

VHDL Very High Speed Integrated Circuit Hardware Description Language

V&V Verification and Validation

XML eXtensible Markup Language