Safety-Critical Systems

Design and Integration of Embedded Systems

István Majzik

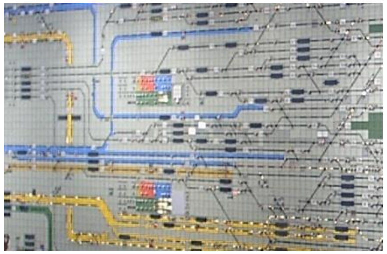
Department of Measurement and Information Systems

Goal of this study block

- Based on previous topics:
  - Requirements specification
  - Architecture design
  - Testing and analysis
- Focus on the design of safety-critical systems
- Specific steps and techniques
  1. Requirements in critical systems: Safety, dependability
  2. Architecture design in critical systems
  3. Hazard analysis: Evaluation of design decisions
  4. Quantitative analysis of safety and dependability
  5. Model-based design

Introduction

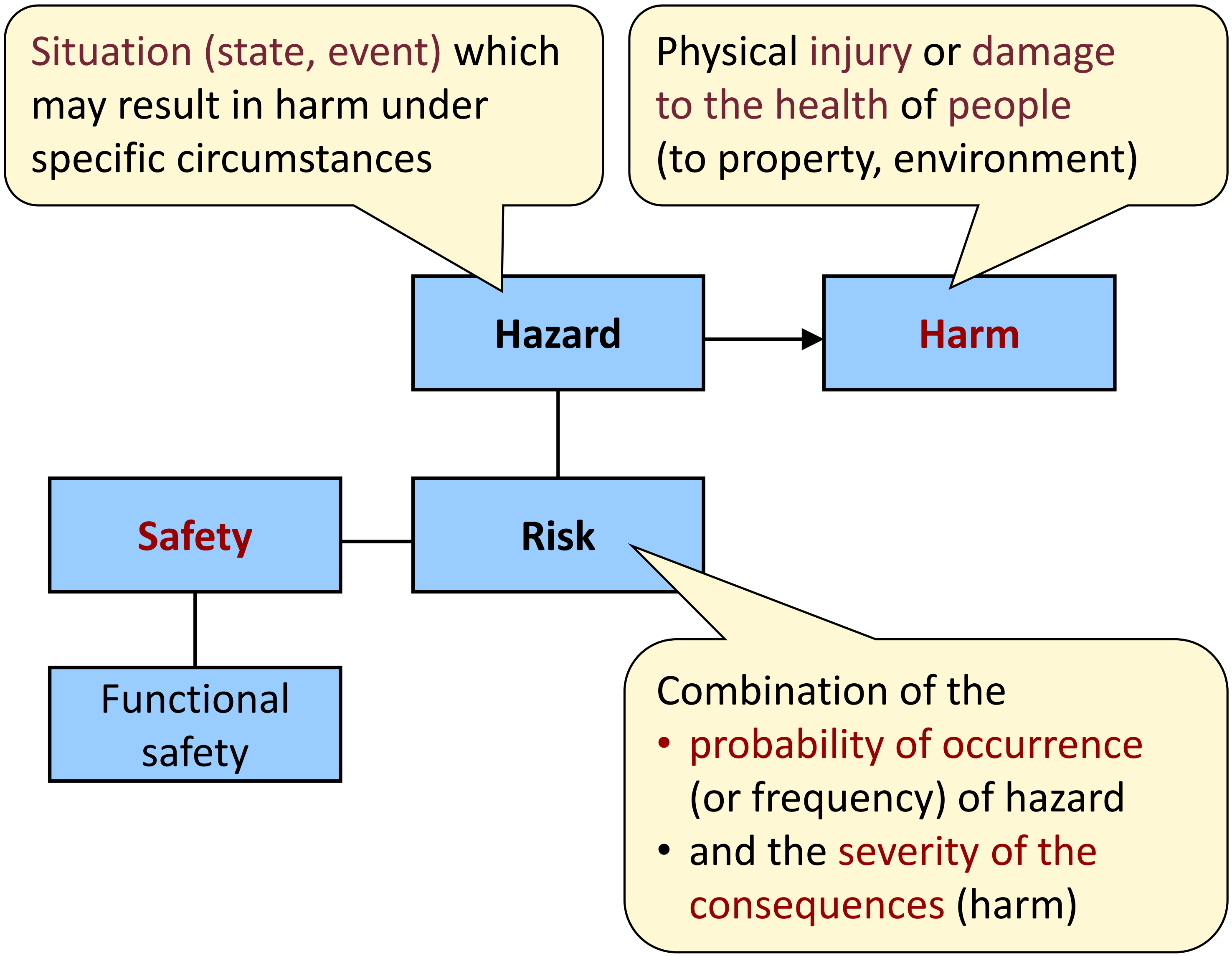
- Safety-critical systems
  - Informally: Malfunction may cause injury of people
- Safety-critical computer-based systems
  - E/E/PE: Electrical, electronic, programmable electronic systems
  - Provide control, protection, or monitoring
  - EUC: Equipment under control
- Basis of development: Standards
  - IEC 61508: Generic standard (for electrical, electronic or programmable electronic systems)
  - DO178B/C: Software in airborne systems and equipment
  - EN 50129/8: Railway control systems / software
  - ISO 26262: Automotive systems
  - Other sector-specific standards: Medical, process control, nuclear, etc.



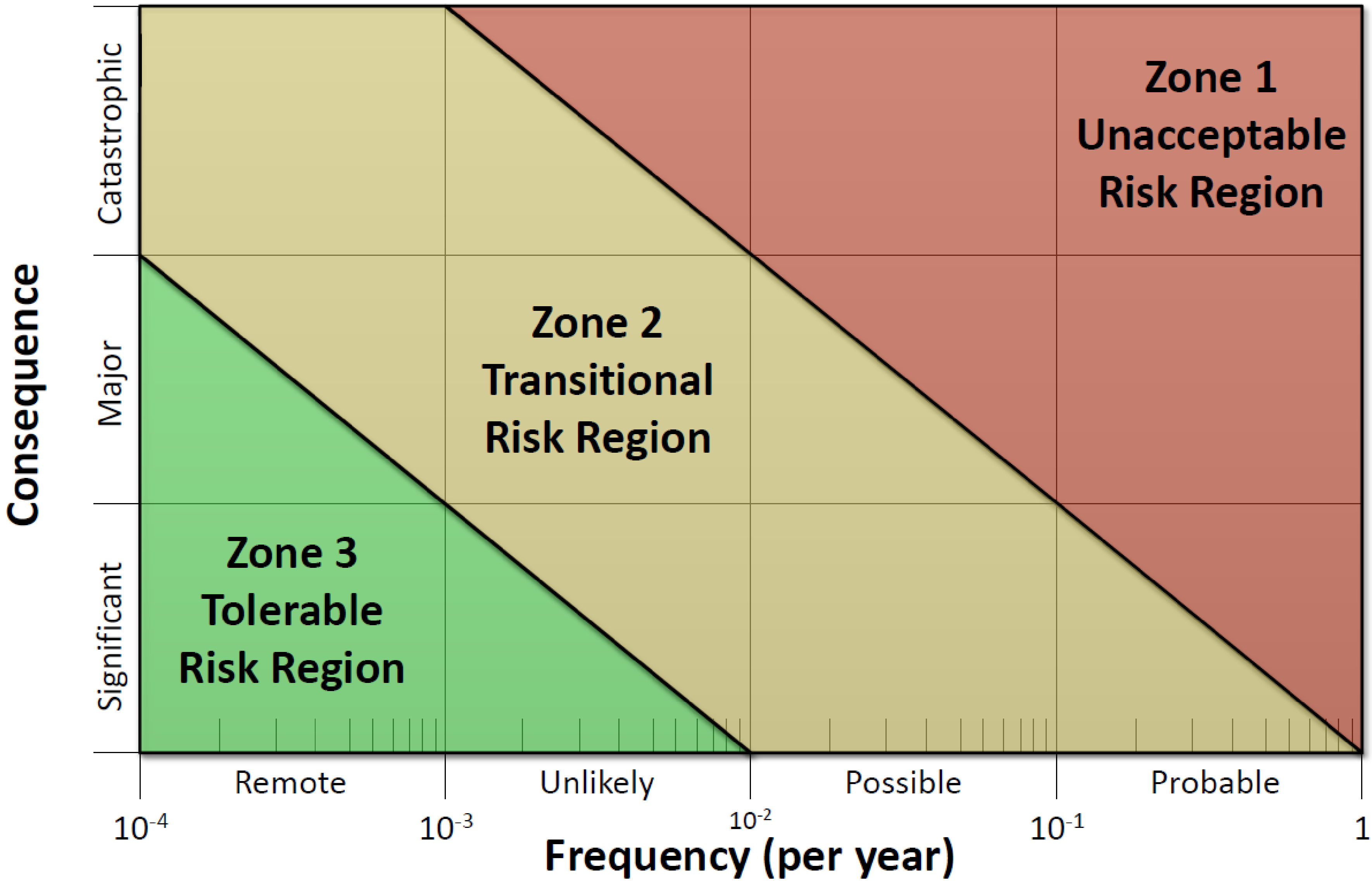
X-by-wire,
engine control,
railway interlocking,
signaling, …

Safety Requirements

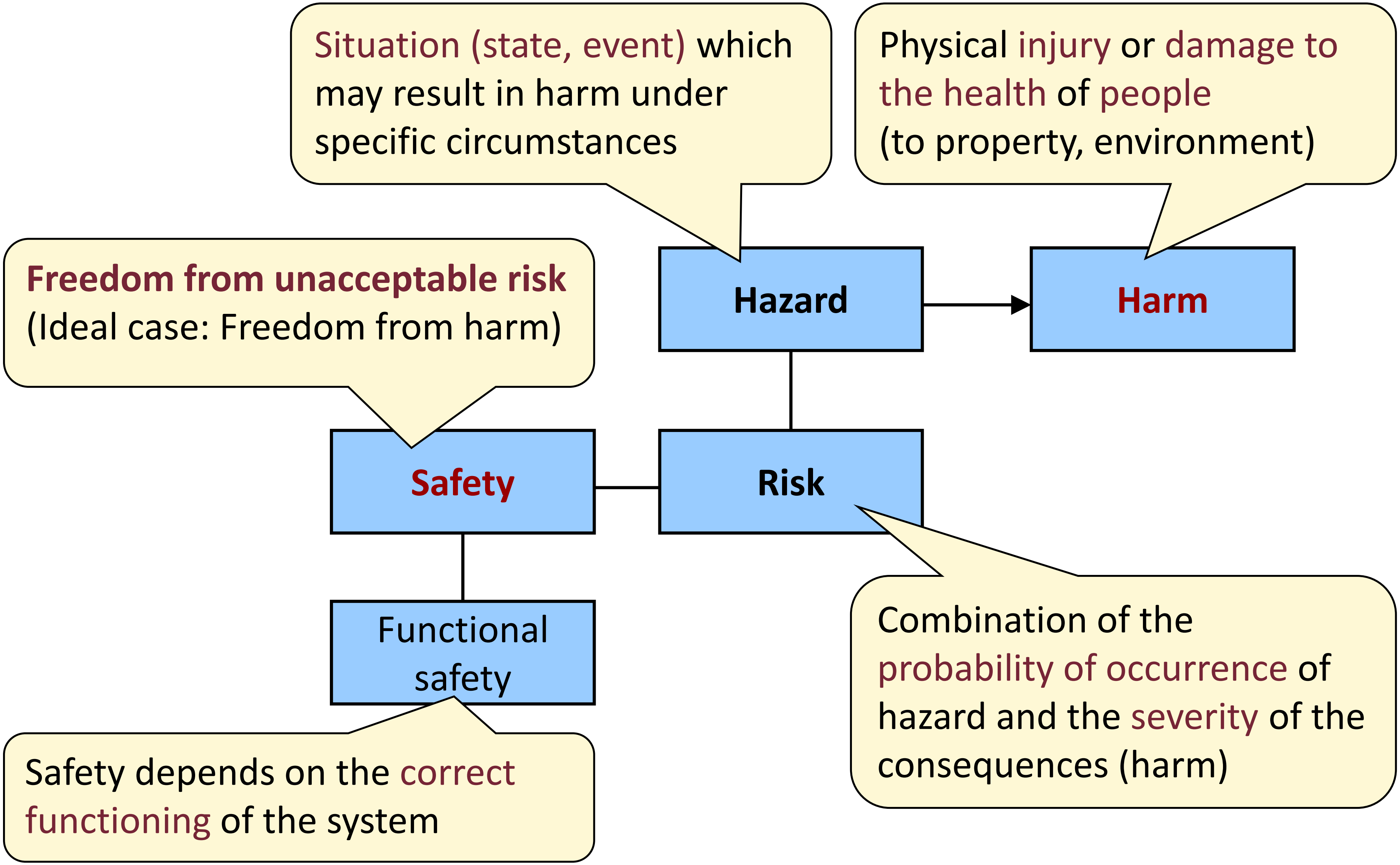
Terminology in the requirements



Risk categories



Terminology in the requirements



Example: Level crossing with barrier and control light

Car is on the rail when the train is approaching

Hazard → **Harm**

Collision of a car and a train

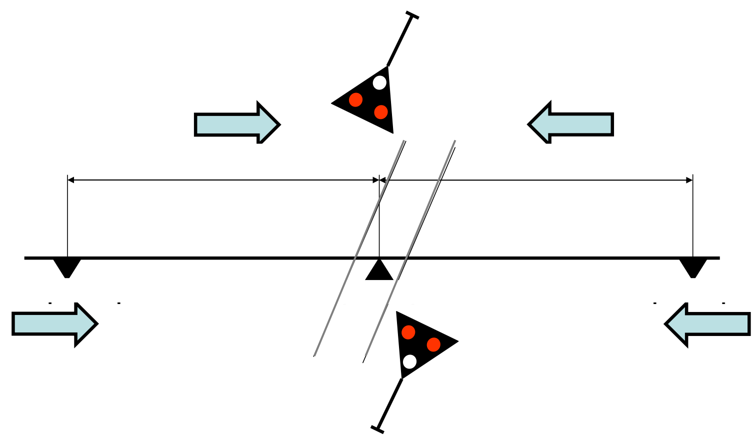
Goal: Freedom from unacceptable risk

Safety — **Risk**

Frequency: Probable
Consequence: Catastrophic

Installation of a barrier and control light;
Closing the level crossing when the train is approaching

Functional safety



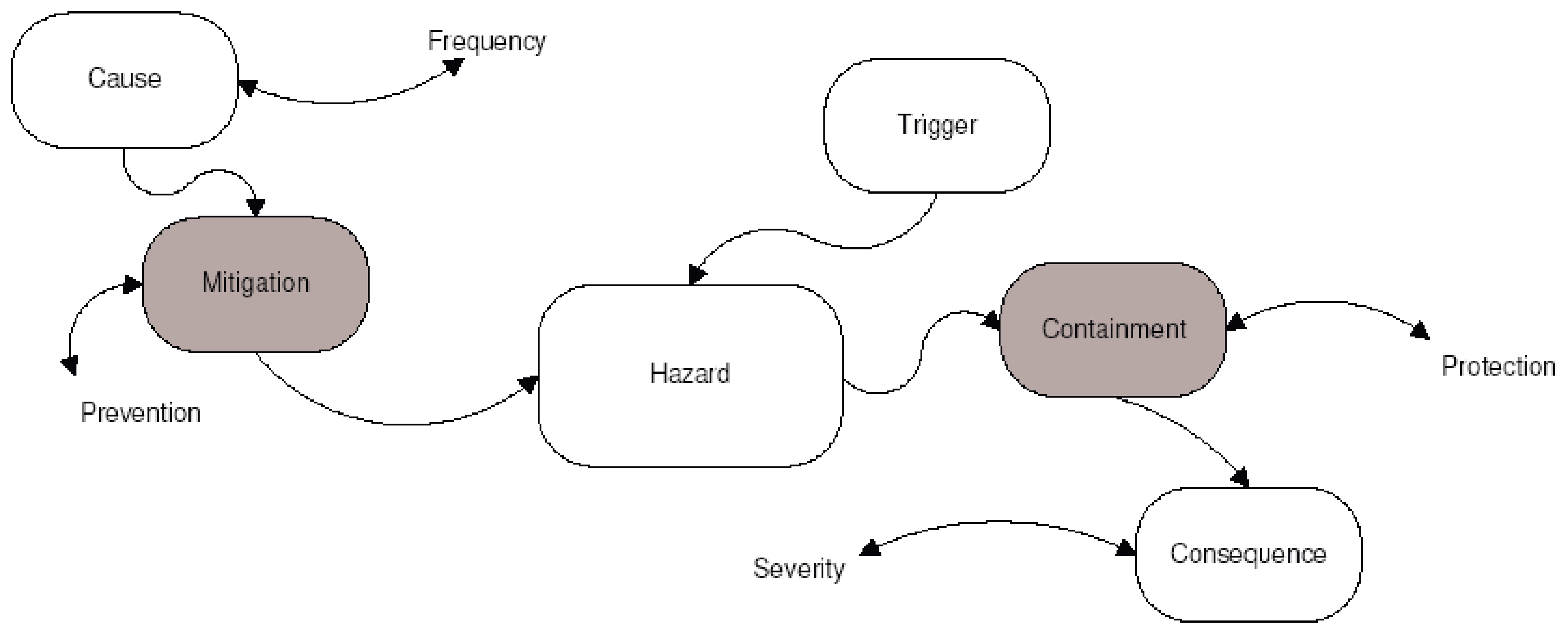
What we have to specify?

- Safety function requirements
  - Function which is intended to achieve or maintain a safe state for the EUC
    - What the system shall do in order to avoid or control the hazard
  - It is part of the functional requirements specification
- Safety integrity requirements
  - Probability that the safety-related system performs the required safety functions (i.e., without failure)
  - Probabilistic approach to safety
    - Example 1: Buildings are designed to survive earthquake that occurs with probability >0.1 in 50 years
    - Example 2: Dams of rivers are designed to withhold the highest water measured in the last 100 years

Safety function requirements

Role of safety functions in hazard control:

- Hazard mitigation
  - Eliminate or decrease the cause of a hazard
- Hazard containment
  - Protect or reduce the consequence of a hazard



Safety integrity requirements

Specification of integrity depends on mode of operation

- Low demand mode:
  - Occasional, rare operation (e.g., a protection system operating only in case of a failure of an EUC)
  - Specified: The allowed average probability of failure to perform the desired function on demand
  - PFD: Probability of Failure on Demand
- High demand mode:
  - Continuous operation (e.g., a system provides continuous control to an EUC)
  - Specified: Average rate of failure to perform the desired function (rate: failure per hour)
  - PFH: Probability of Failure per Hour

    → THR: Tolerable Hazard Rate

Safety integrity levels (SIL)

- Low demand mode:

SIL	Average probability of failure to perform the function on demand
1	$10^{-2} \leq PFD < 10^{-1}$
2	$10^{-3} \leq PFD < 10^{-2}$
3	$10^{-4} \leq PFD < 10^{-3}$
4	$10^{-5} \leq PFD < 10^{-4}$

- High demand mode:

Operation without failures in more than 11.000 years???

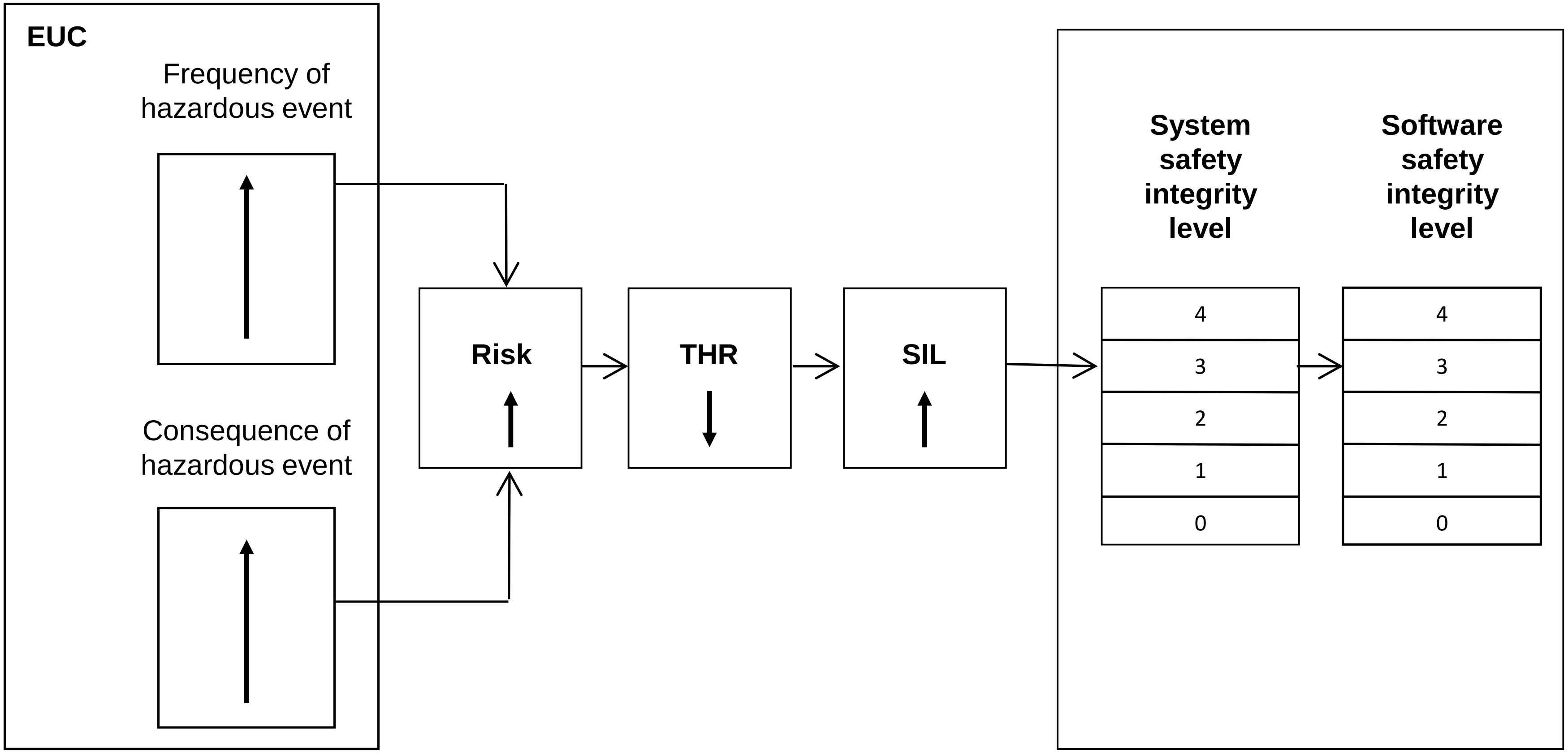
15 years lifetime: 1 failure in case of 750 devices

SIL	Average probability of failure per hour per safety function
1	$10^{-6} \leq PFH < 10^{-5}$
2	$10^{-7} \leq PFH < 10^{-6}$
3	$10^{-8} \leq PFH < 10^{-7}$
4	$10^{-9} \leq PFH < 10^{-8}$

(PFH or THR)

Determining SIL: Overview

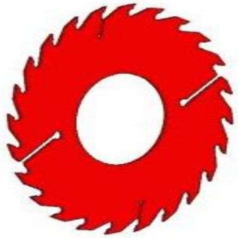
- Hazard identification and risk analysis -> Target SIL



Example: Safety requirements

- Machine with a rotating blade and a solid cover
  - Cleaning of the blade: Lifting up the cover

- Risk analysis: Injury of the operator is possible when cleaning the blade while it is rotating
  - Hazard: If the cover is lifted more than 50 mm and the motor of the bladed does not stop in 3 sec
  - There are 20 machines; during the lifetime 500 cleaning is needed for each machine; it is tolerable only once that the motor is not stopped



- Safety function: Protection mechanism
  - Safety function requirement: When the cover is lifted to 25 mm, the motor shall be stopped in 2.5 sec
- Safety integrity requirement:
  - The probability of failure of the protection mechanism (as a safety function) shall be less than 10^{-4} (one failure in 10.000 operation)

Satisfying safety integrity requirements

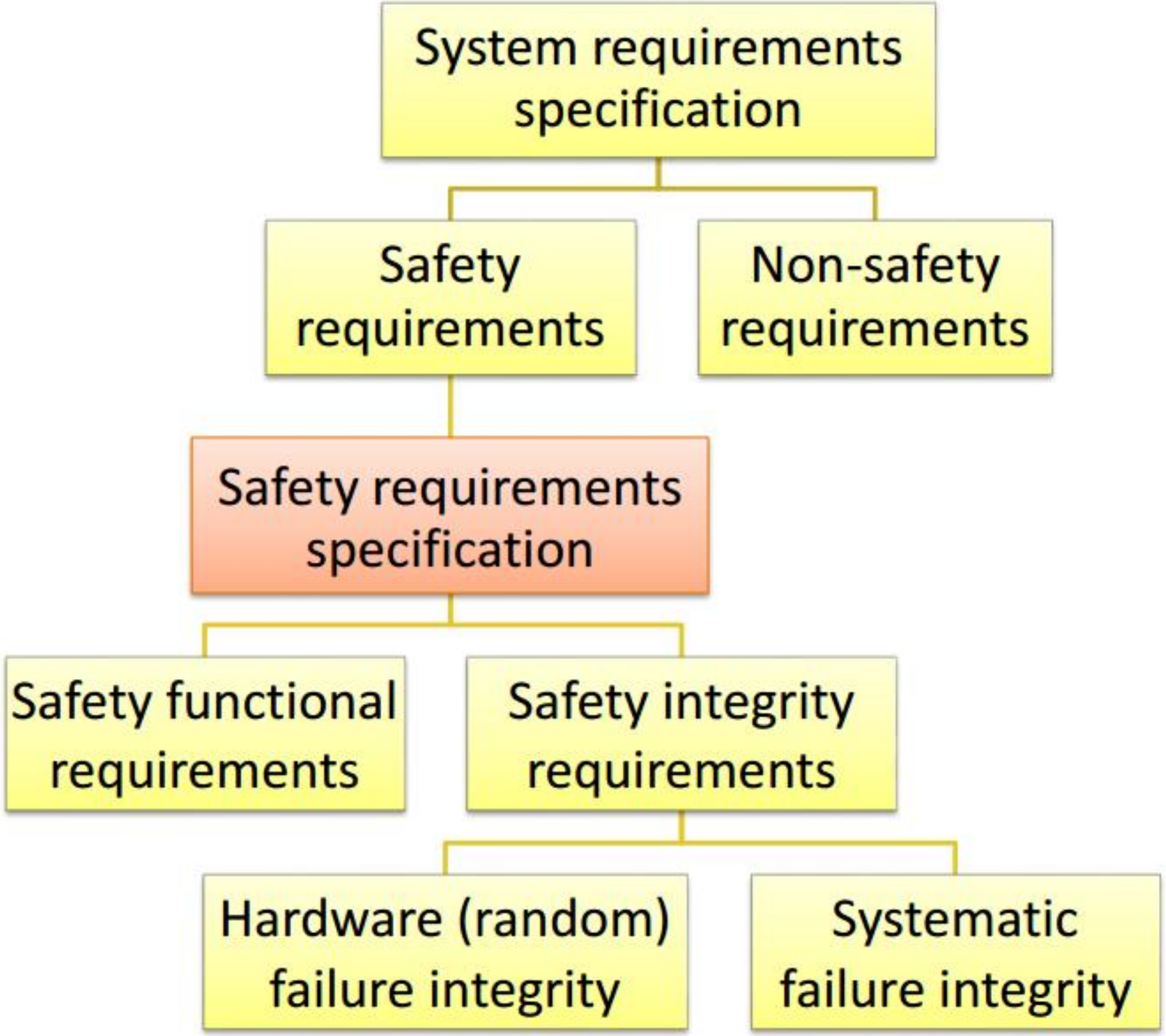
"Safety case" is needed

- Documented demonstration that the product complies with the specified safety requirements

How to demonstrate safety integrity - depends on failures

- Random (hardware) failures:
  - Occur accidentally at a random time due to degradation mechanisms
  - Random failure integrity: Calculations on the basis of component fault rates ← depends on selection of components and the system architecture
- Systematic (software) failures:
  - Occur in a deterministic way due to design / manufacturing / operating flaws
  - Systematic failure integrity: Rigor in the development
    - Development life cycle: Well-defined phases
    - Techniques and measures: Verification, testing, measuring, …
    - Documentation: Development and operation related
    - Independence of persons: Developer, verifier, assessor, …

Summary: Structure of requirements



Dependability requirements

Characterizing the system services

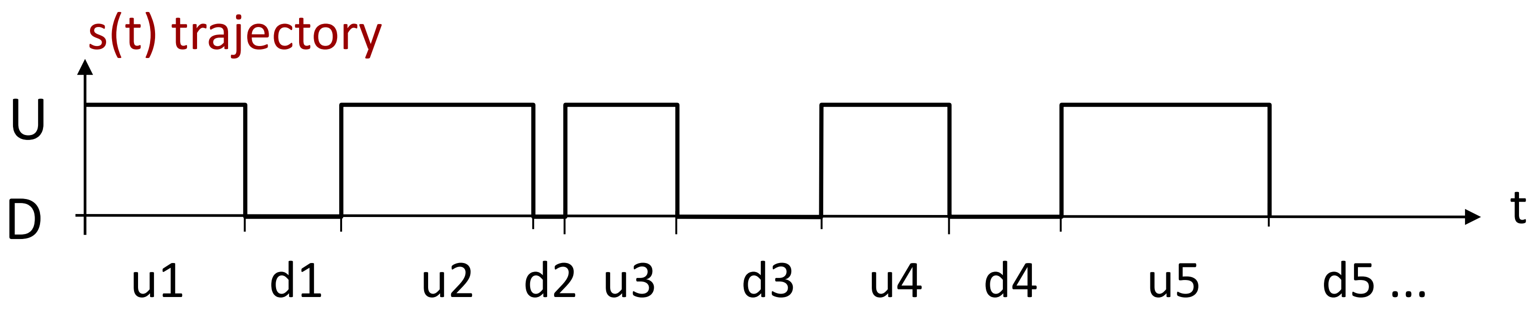
- Requirement: Useful, functioning services
  - Characterized by: Reliability, availability, integrity, ...
  - These depend on the faults occurring during the use of the services of the system
  - Basic question: How to avoid or handle the faults affecting the services?
- Composite characteristic: Dependability
  - Definition: Ability to provide service in which reliance can justifiably be placed
    - Reliance: the service satisfies the needs
    - Justifiably: based on analysis, evaluation, measurements

Attributes of dependability

Attribute	**Definition**
Availability	Probability of correct service (considering failures, repairs and maintenance) E.g., availability of a web service shall be 95%
Reliability	Probability of continuous correct service (until the first failure) E.g., after departure, the flight control system shall function correctly until the arrival
Safety	Freedom from unacceptable risk of harm
Integrity	Avoidance of erroneous changes or alterations (e.g., in data)
Maintainability	Possibility of repairs and improvements

Dependability metrics: Mean values

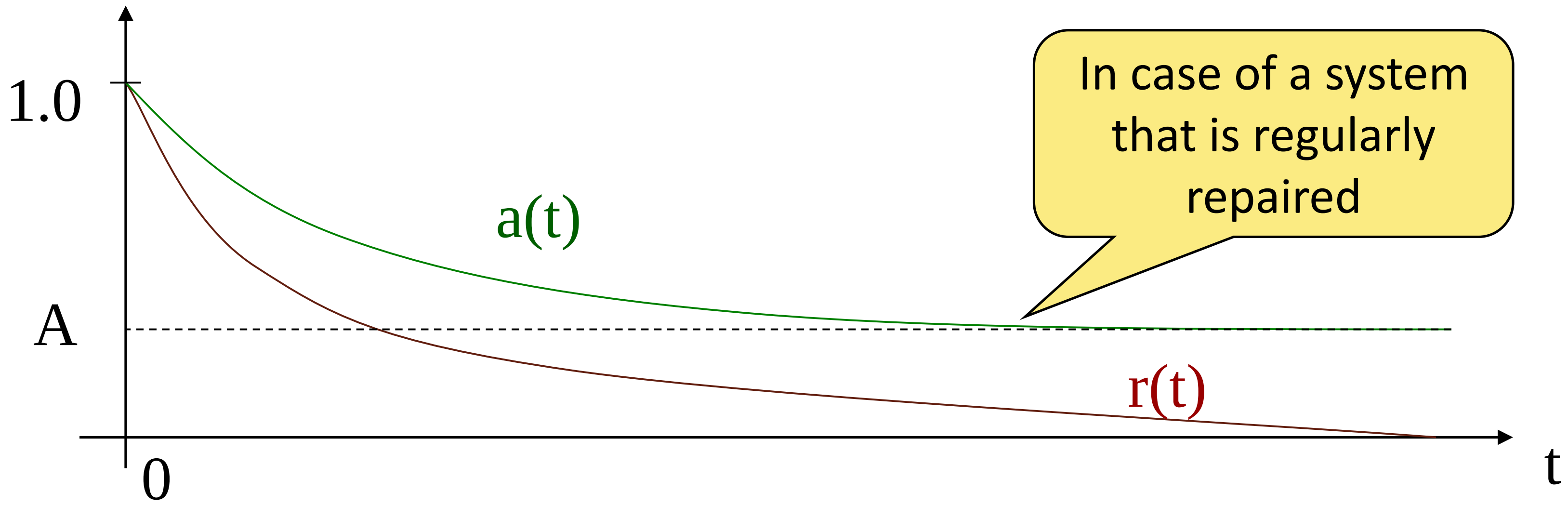
- Basis: Partitioning the states of the system s(t)
  - Correct (U, up) and incorrect (D, down) state partitions



- Mean values:
  - Mean Time to First Failure: MTFF = E{u1}
  - Mean Up Time: MUT = MTTF = E{u_i}
    (Mean Time To Failure)
  - Mean Down Time: MDT = MTTR = E{d_i}
    (Mean Time To Repair)
  - Mean Time Between Failures: MTBF = MUT + MDT

Dependability metrics: Probability functions

- Availability: $$a(t) = P\{s(t) \in U\}$$
- Asymptotic availability: $$A = \lim_{t \to \infty} a(t)$$
$$A = \frac{MTTF}{MTTF + MTTR}$$
- Reliability: $$r(t) = P\{s(t') \in U, \forall t' < t\}$$



Component attribute: Fault rate

- Fault rate (fault occurrence rate): $\lambda(t)$

  $\lambda(t)\Delta t$ gives the probability that the component will fail in the interval Δt at time point t given that it has been correct until t

$$\lambda(t)\Delta t = P\left\{s(t+\Delta t) \in D \mid s(t) \in U\right\} \text{ while } \Delta t \to 0$$

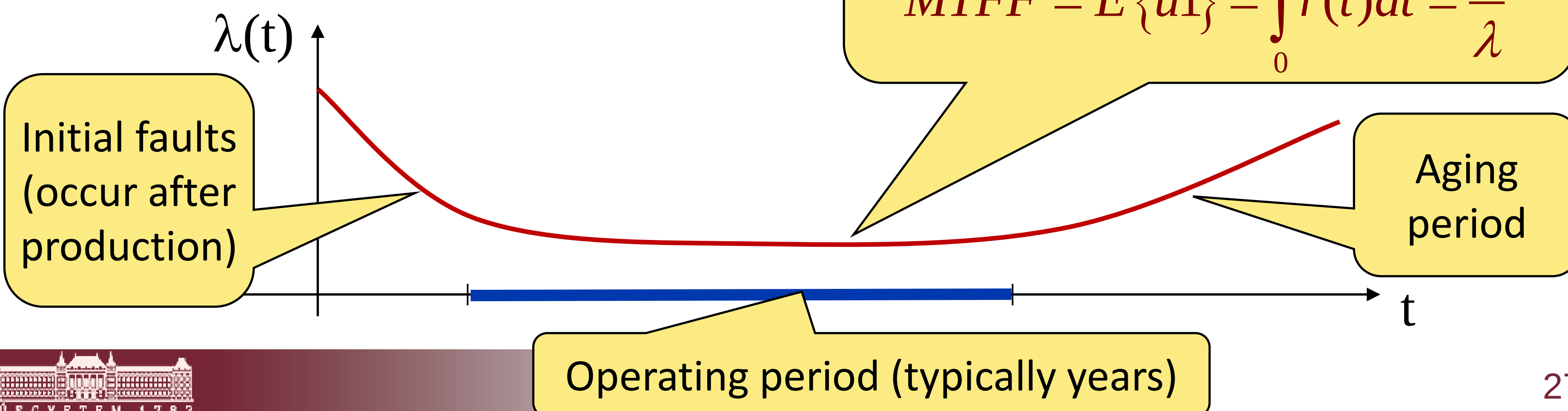
- Reliability of a component can be derived using $\lambda(t)$:

$$r(t) = e^{-\int_0^t \lambda(t)\,dt}$$

- For electronic components:

Here $r(t) = e^{-\lambda t}$

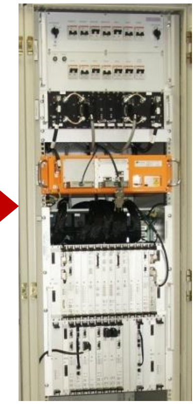
$$MTFF = E\{u1\} = \int_0^\infty r(t)\,dt = \frac{1}{\lambda}$$



Example: Development of a DMI

Driver

Driver-Machine Interface



EVC

EVC:
European Vital Computer (on board of the train)

Maintenance center

Characteristics:

- Safety-critical functions
  - Visualization of information
  - Processing driver commands
  - Data transfer from/to EVC
- Safe wireless communication
  - System configuration
  - Diagnostics
  - Software update

Example: DMI requirements

- Safety:
  - Tolerable Hazard Rate: 10^{-7} <= THR < 10^{-6} 1/hours hazardous failures per hours
  - Safety Integrity Level: SIL 2
- Reliability:
  - Mean Time To Failure: MTTF > 5 000 hours (5000 hours: ~ 7 months)
- Availability:
  - A = MTTF / (MTTF+MTTR), A > 0.9952
    - In faulty state: less than 42 hours per 1 year
    - Satisfied: if MTTF = 5000 hours then MTTR < 24 hours

Threats to dependability: Faults

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Verification during the development

Fault tolerance during the operation

The characteristics of faults

- **Fault**
  - In space
    - Internal
      - Physical (hardware)
      - Design (typ. software)
    - External
      - Physical (environment)
      - Data (input)
  - In time
    - Intermittent (transient)
    - Permanent

Software fault:

- Permanent, internal design fault (systematic)
- Activation of the fault depends on the operational profile (inputs)

How faults lead to failures?

Fault:
Adjudged or hypothesized cause of an error

→ Component or system

Error: State leading to the failure →

Failure:
The delivered service deviates from correct service

Fault → Error → Failure chain examples:

Fault →	Error →	Failure
Bit flip in the memory due to a cosmic particle	Reading the faulty memory cell will result in incorrect control value	The robot arm collides with the wall
The programmer increases a variable instead of decreasing	The faulty statement is executed and thus the value of a state variable will be incorrect	The final result of the computation will be incorrect

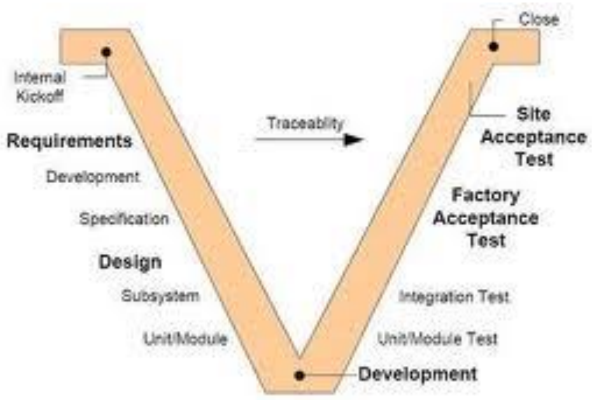
Means to improve dependability

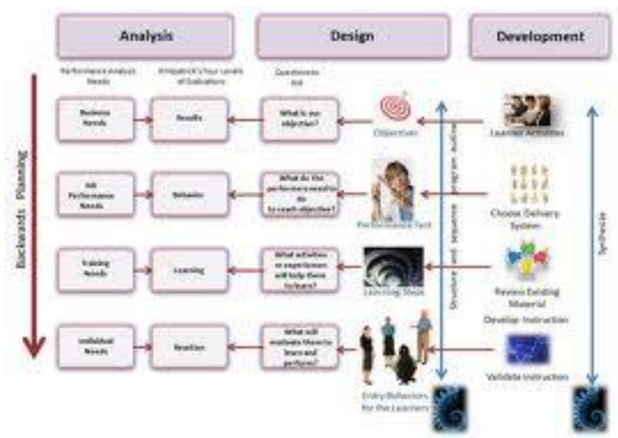
- Fault prevention:
  - Physical faults: Good components, protection, ...
  - Design faults: Good design methodology
- Fault removal:
  - Design phase: Verification and corrections
  - Production phase: Testing, diagnostics, and repair
- Fault tolerance: Avoiding service failures
  - Operation phase: Fault handling, reconfiguration
- Fault forecasting: Estimating faults and their effects
  - Operation phase: Measurements and prediction

Summary

- Safety requirements
  - Basic concepts: Hazard, risk, safety
  - Safety function and safety integrity requirements
  - Safety integrity levels
- Dependability requirements
  - Attributes of dependability
  - Quantitative definitions: reliability and availability
  - Threats: The fault → error → failure chain
  - Means to improve dependability: fault prevention, fault removal, fault tolerance, fault forecasting

Overview of the development of safety-critical systems





Recap: Demonstrating safety integrity

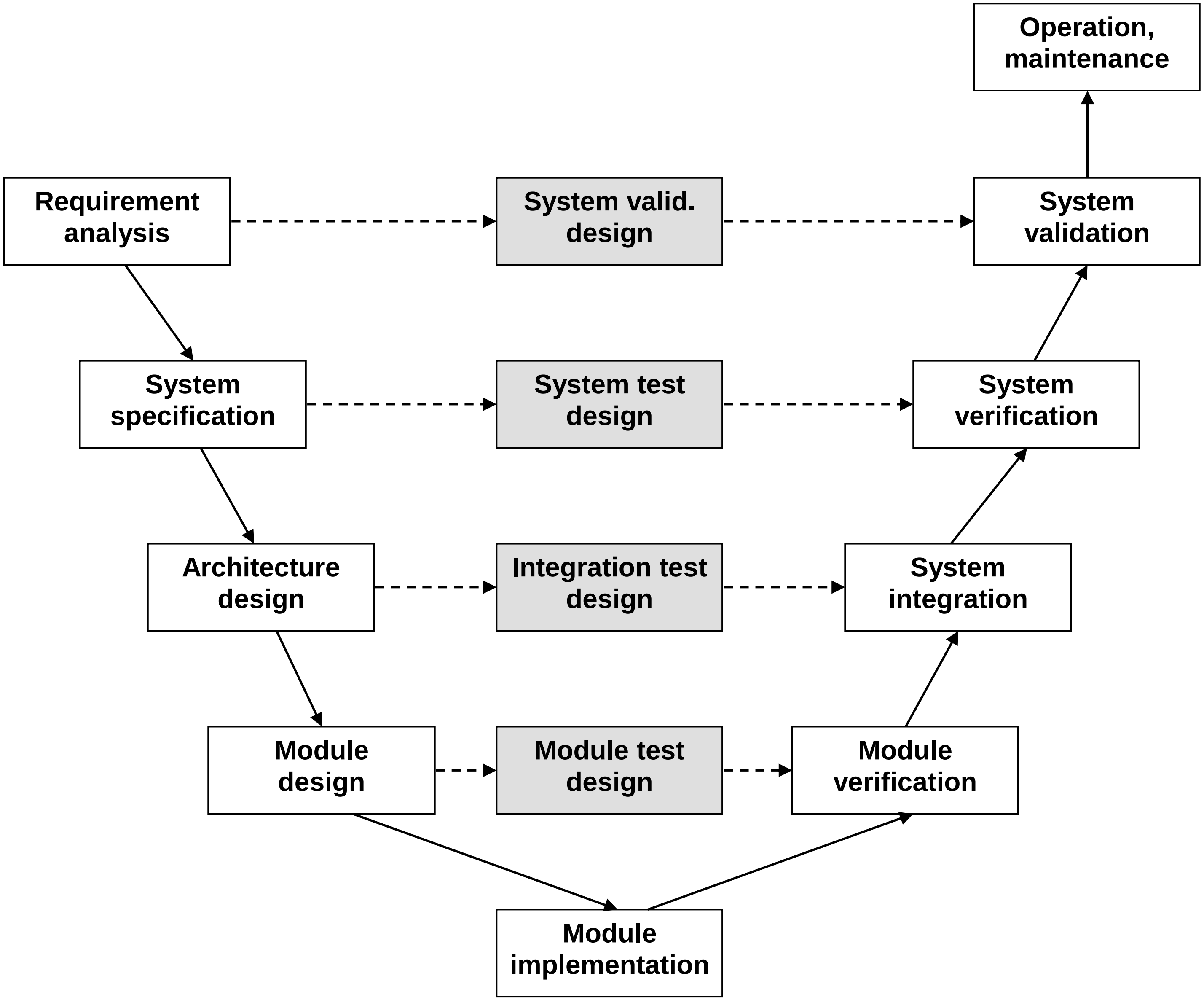
- Random (hardware) failures:
  - Occur accidentally at a random time due to degradation mechanisms
  - Random failure integrity: Statistical calculations on the basis of component fault rates
- Systematic (software) failures:
  - Occur in a deterministic way due to design / manufacturing / operating flaws
  - No accepted general method to calculate safety integrity
  - Systematic failure integrity: Rigor in the development
    - Development lifecycle: Well-defined, verified phases
    - Techniques and measures: Design, verification, …
    - Documentation: Development and operation related
    - Independence of persons: Developer, verifier, assessor, …

1. Development lifecycle

Goals of the overall safety lifecycle model:

- Provide well-defined technical framework for the activities necessary for ensuring functional safety
  - E.g., verification in each phase before proceeding
- Cover all lifecycle activities
  - Initial concept
  - Hazard analysis and risk assessment
  - Specification, design, implementation
  - Operation and maintenance
  - (Final decommissioning and/or disposal)

Example software lifecycle (V-model)



2. Techniques and measures

- Goals of the required techniques:
  - Preventing the introduction of systematic faults
  - Controlling the residual faults
- SIL determines the set of techniques to be applied as
  - M: Mandatory
  - HR: Highly recommended (rationale behind not using it should be detailed and agreed with the assessor)
  - R: Recommended
  - ---: No recommendation for or against being used
  - NR: Not recommended
- Combinations of techniques are allowed
  - E.g., alternative or equivalent techniques
- Hierarchy of techniques is provided

Example: Testing techniques (EN 50128)

- Testing in the software design and implementation phase:

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
14. Functional/ Black-box Testing	D.3	HR	HR	HR	M	M
15. Performance Testing	D.6	-	HR	HR	HR	HR
16. Interface Testing	B.37	HR	HR	HR	HR	HR

- D3: Functional / black box testing:

1. Test Case Execution from Cause Consequence Diagrams	B.6	-	-	-	R	R
2. Prototyping/Animation	B.49	-	-	-	R	R
3. Boundary Value Analysis	B.4	R	HR	HR	HR	HR
4. Equivalence Classes and Input Partition Testing	B.19	R	HR	HR	HR	HR
5. Process Simulation	B.48	R	R	R	R	R

Example: Testing techniques (EN 50128)

- D6: Performance testing:

TECHNIQUE/MEASURE		Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Avalanche/Stress Testing	B.3	-	R	R	HR	HR
2.	Response Timing and Memory Constraints	B.52	-	HR	HR	HR	HR
3.	Performance Requirements	B.46	-	HR	HR	HR	HR

Example: Software architecture design (IEC 61508)

- IEC 61508: Functional safety in electrical / electronic / programmable electronic safety-related systems

- Here: Techniques that are NR (not recommended)

Table A.2 – Software design and development: software architecture design (see 7.4.3)

	Technique/Measure*	Ref	SIL1	SIL2	SIL3	SIL4
1	Fault detection and diagnosis	C.3.1	---	R	HR	HR
2	Error detecting and correcting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Safety bag techniques	C.3.4	---	R	R	R
3c	Diverse programming	C.3.5	R	R	R	HR
3d	Recovery block	C.3.6	R	R	R	R
3e	Backward recovery	C.3.7	R	R	R	R
3f	Forward recovery	C.3.8	R	R	R	R
3g	Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h	Memorising executed cases	C.3.10	---	R	R	HR
4	Graceful degradation	C.3.11	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	HR	HR	HR
7b	Semi-formal methods	Table B.7	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8	Computer-aided specification tools	B.2.4	R	R	HR	HR

NOTE – The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in IEC 61508-2.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

3. Precise documentation

- Types of documentation
  - Comprehensive (covers overall lifecycle)
    - E.g., Software Verification Plan
  - Specific for a given lifecycle phase
    - E.g., Software Source Code Verification Report
- Document Cross Reference Table
  - Specifies documentation for each lifecycle phase
  - Determines relations among documents
- Traceability of documents is required
  - Relationship between documents is specified (e.g., “based on”, “includes”)
  - Consistent terminology, references, abbreviations

Example: Document structure (EN50128)

Software Planning Phase
- Software Development Plan
- Software Quality Assurance Plan
- Software Configuration Management Plan
- Software Verification Plan
- Software Integration Test Plan
- Software/hardware Integration Test Plan
- Software Validation Plan
- Software Maintenance Plan

System Development Phase
- System Requirements Specification
- System Safety Requirements Specification
- System Architecture Description
- System Safety Plan

Software Requirements Spec. Phase
- Software Requirements Specification
- Software Requirements Test Specification
- Software Requirements Verification Report

Software Architecture & Design Phase
- Software Architecture Specification
- Software Design Specification
- Software Architecture and Design Verification Report

Software Module Design Phase
- Software Module Design Specification
- Software Module Test Specification
- Software Module Verification Report

Coding Phase
- Software Source Code & Supporting Documentation
- Software Source Code Verification Report

Software Module Testing Phase
- Software Module Test Report

Software Integration Phase
- Software Integration Test Report

Software/hardware Integration Phase
- Software/hardware Integration Test Report

Software Validation Phase
- Software Validation Report

Software Assessment Phase
- Software Assessment Report

Software Maintenance Phase
- Software Maintenance Records
- Software Change Records

30 documents in a systematic structure

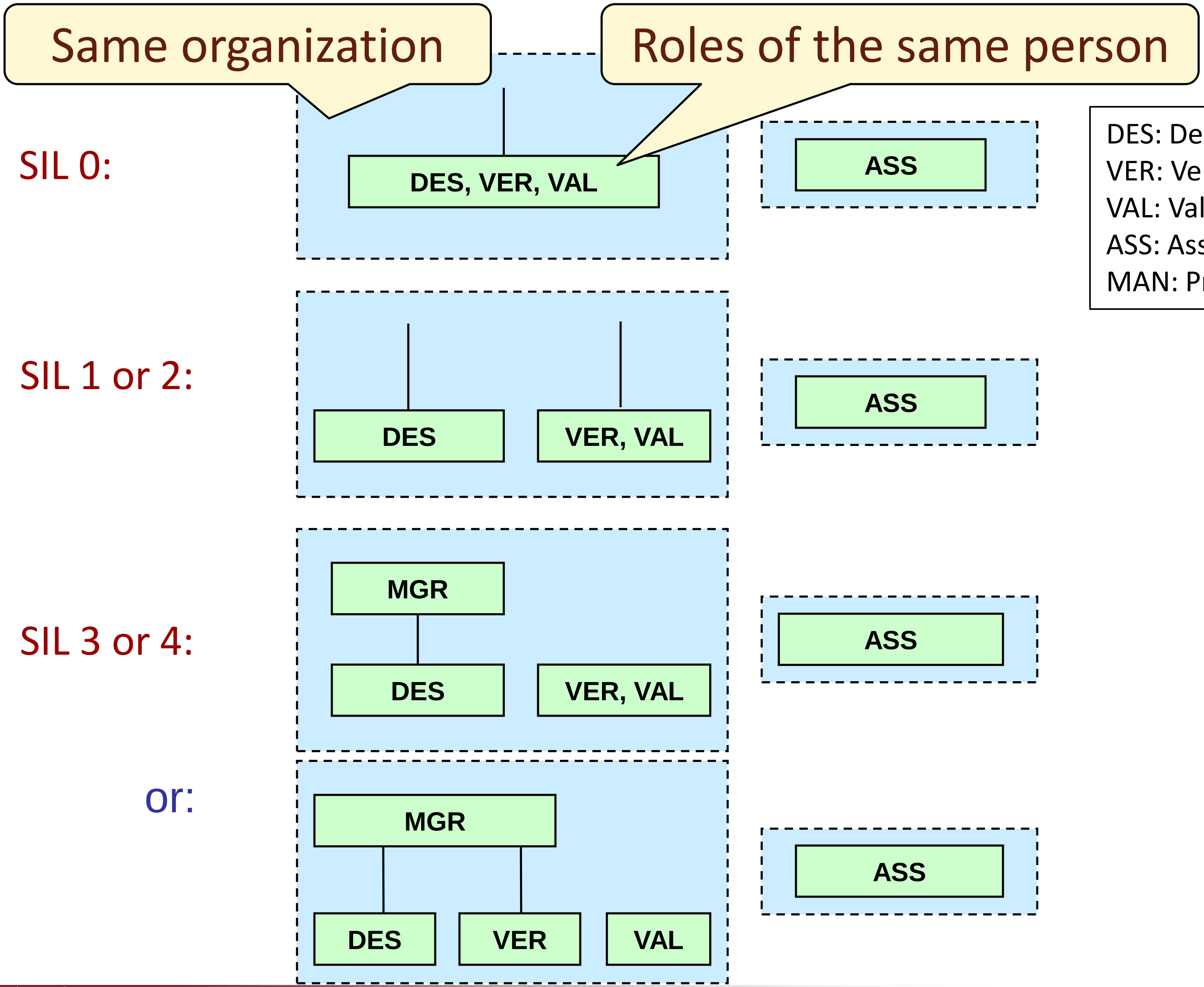
- Specification
- Design
- Verification

4. Organization and independence of roles

- Safety management
  - Quality assurance personnel
  - Safety Organization (responsible persons)

- Competence shall be demonstrated
  - Training, experience and qualifications
- Independence of roles:
  - DES: Designer (analyst, architect, coder, unit tester)
  - VER: Verifier (incl. integration and system tester)
  - VAL: Validator
  - ASS: Assessor
  - MAN: Project manager
  - QUA: Quality assurance personnel

Example: Responsibilities (EN 50128)



DES: Designer
VER: Verifier
VAL: Validator
ASS: Assessor
MAN: Project manager

Summary

- Basic notions of safety-critical systems
  - Hazard, risk, safety
  - Safety integrity, THR, SIL
- Dependability
  - Attributes
  - Fault –> Error –> Failure chain
  - Means for improving dependability
- Development processes and standards
  - Lifecycle, measures and techniques
  - Documentation, organizational structure