Safety-critical systems: Requirements

Systems Engineering course

István Majzik
majzik@mit.bme.hu

Budapest University of Technology and Economics
Department of Measurement and Information Systems

Overview of the goals

Goal of this study block

- Based on previous topics…
  - Requirements specification
    - Functional and extra-functional requirements
  - Architecture design
    - Components based on functional decomposition
- Focus on the design of critical systems
  - From requirements to architecture design and evaluation
  - Safety and dependability as extra-functional requirements
- Steps
  1. Requirements in critical systems: Safety, dependability
  2. Architecture design (patterns) in critical systems
  3. Evaluation of system architecture

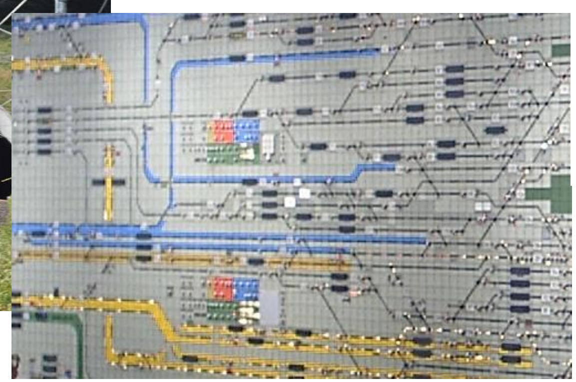
Learning objectives

- Safety requirements
  - Understand the basic concepts of safety
  - Identify the relation of safety functions and safety integrity level
  - Understand the structure of the requirement specification in safety-critical systems
- Dependability requirements
  - Understand the attributes of dependability
  - Capture reliability and availability requirements in quantitative format
  - Understand the role of the fault – error – failure chain
  - Identify the means for improving dependability

Safety requirements

Introduction

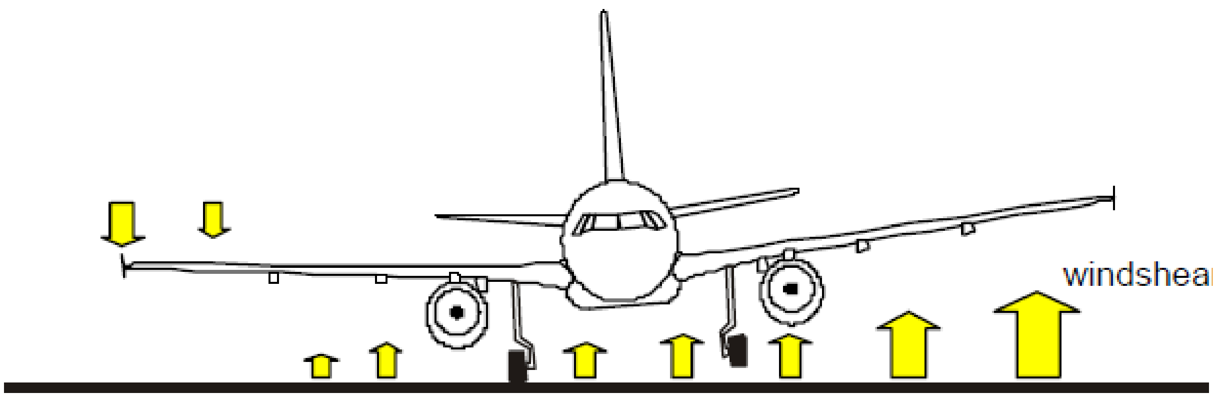
- Safety-critical systems
  - Informal definition: Malfunction may cause injury of people
- Safety-critical computer-based systems
  - E/E/PE: Electrical, electronic, programmable electronic systems
  - Control, protection, or monitoring
  - EUC: Equipment under control



Railway signaling, x-by-wire, interlocking, emergency stopping, engine control, …

Accident examples

- A320-211 Accident in Warsaw (14 September 1993)
  - Windshear
  - Left gear touched the ground 9 sec later than the right
  - Intelligent braking is controlled by shock absorber + wheel rotation -> delayed braking -> hitting the embankment
- Is the control system ”too intelligent”?
- Correct functioning but not safe behaviour!

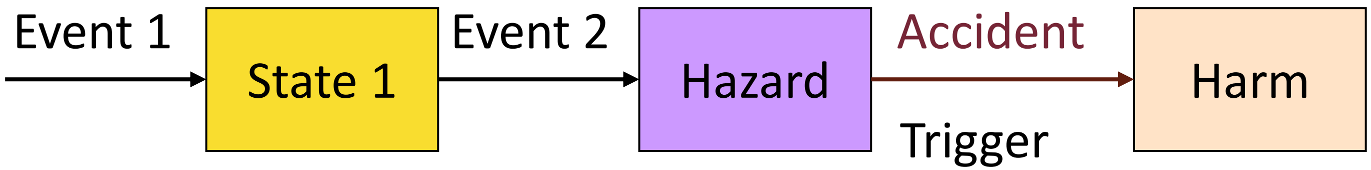


Accident examples

- Toyota car accident in San Diego, August 2009
- Hazard: Stuck accelerator (full power)
  - Floor mat problem
- Hazard control: What about…
  - Braking?
  - Shutting off the engine?
  - Putting the vehicle into neutral? (gearbox: D, P, N)

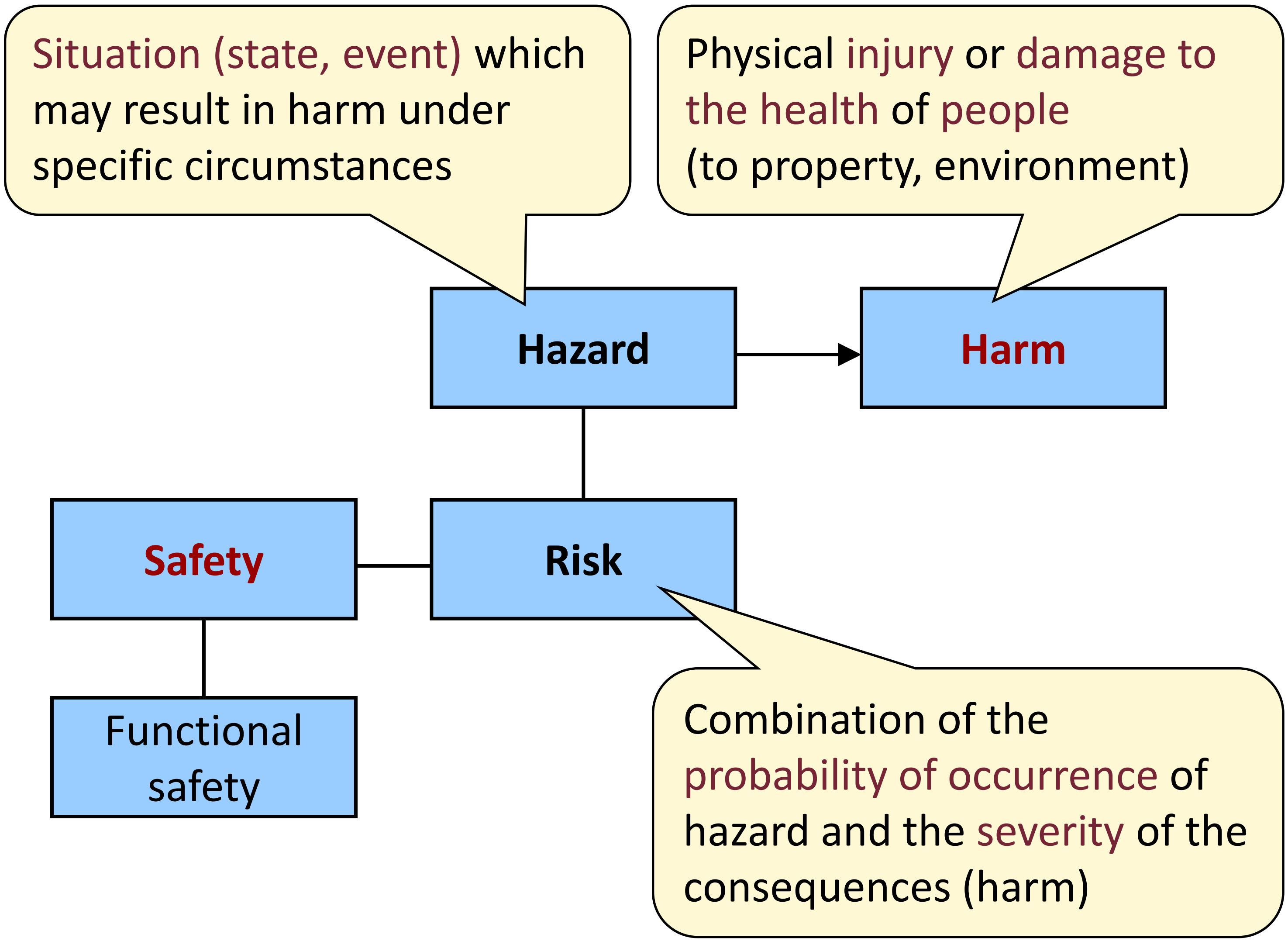
Conclusions from accident examples

- Harm is typically a result of a complex scenario
  - (Temporal) combination of failure(s) and/or normal event(s)
  - Hazards may not result in accidents

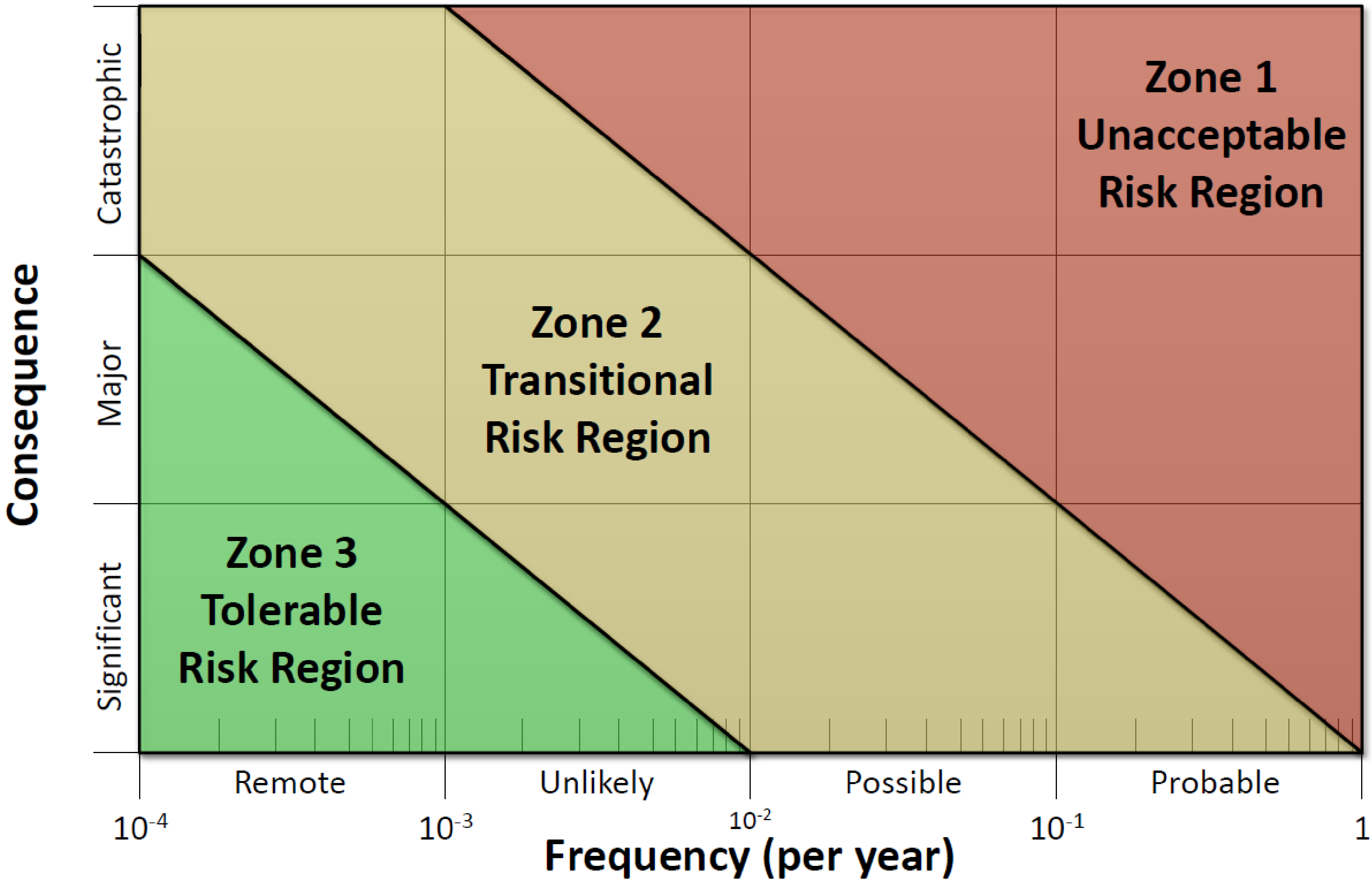


- Hazard ≠ failure
  - Undetected (and unhandled) error is a typical cause of hazards
  - But hazard may also be caused by (unexpected) combination of normal events (correct operation)
- Central problems in safety-critical systems:
  - Analysis of situations that may lead to hazard: Risk analysis
  - Assignment of functions to avoid hazards → accidents → harms
  - Specification of (extra-functional) safety requirements

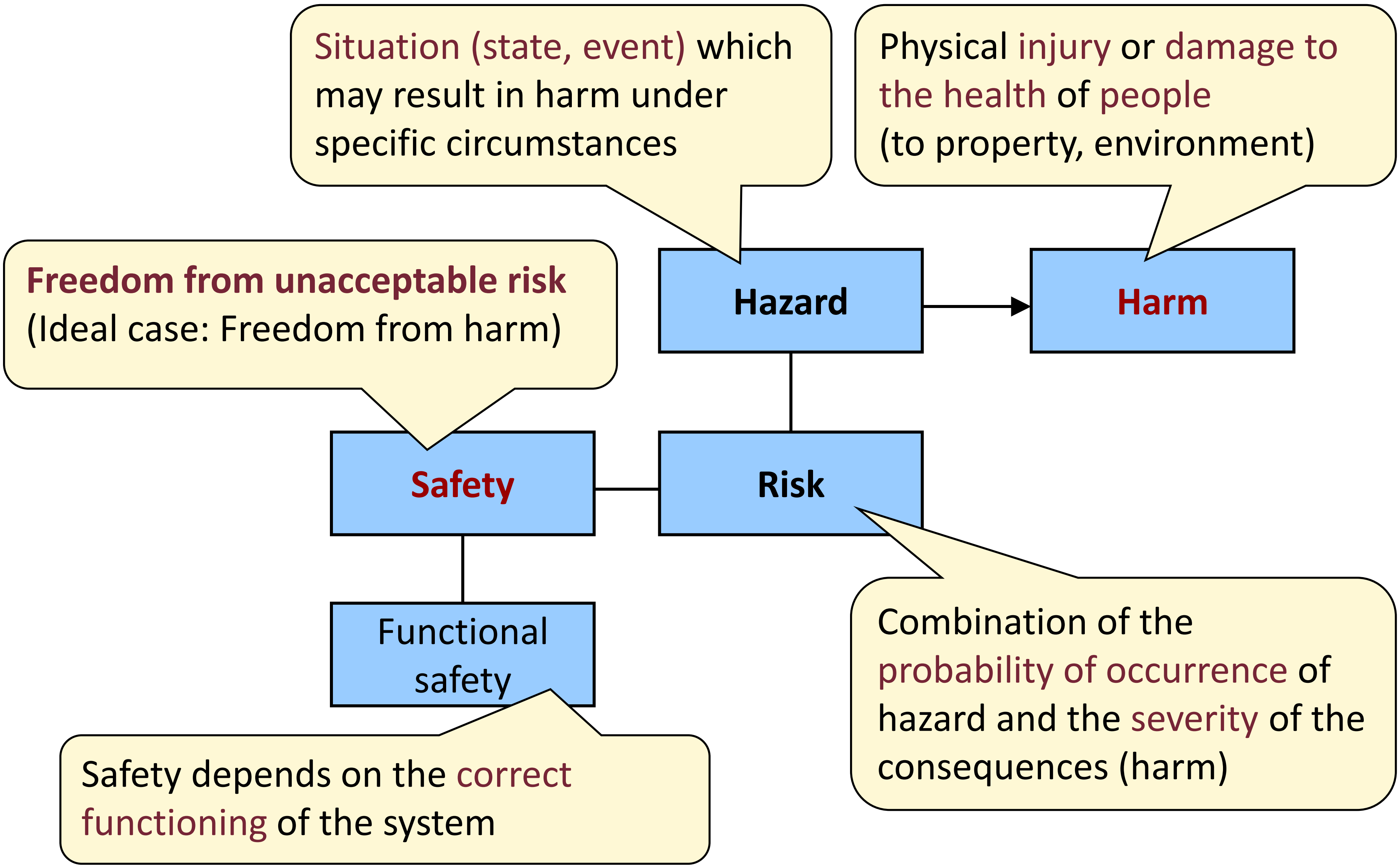
Terminology in the requirements



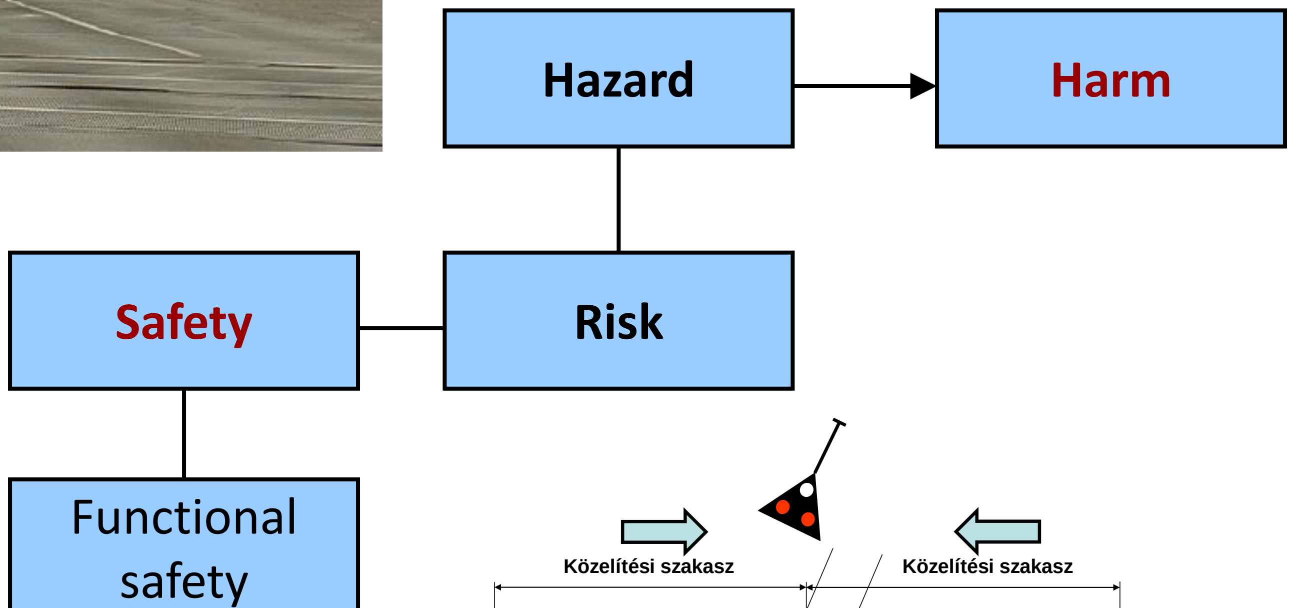
Risk categories

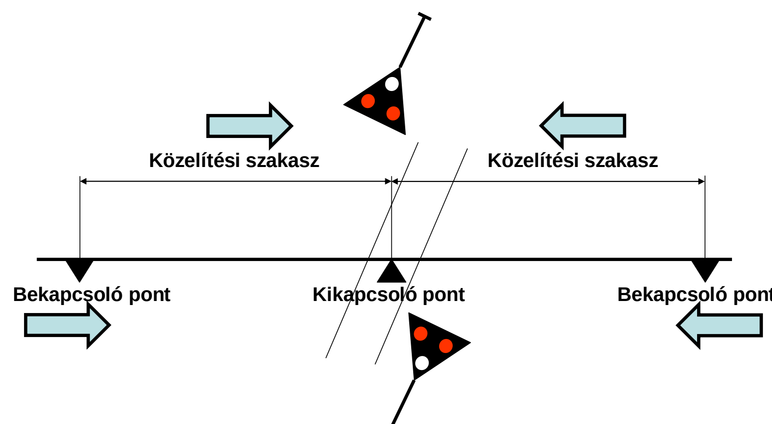


Terminology in the requirements



Example: Application of the terminology





What we have to specify?

- Safety function requirements
  - Function which is intended to achieve or maintain a safe state for the EUC
    - In other words: What the system shall do in order to avoid / control the hazard
  - (Part of the) functional requirements specification
- Safety integrity requirements
  - Probability that the safety-related system satisfactorily performs the required safety functions (without failure)
  - Probabilistic approach to safety
    - Example 1: Buildings are designed to survive earthquake that occurs with probability >10% in 50 years
    - Example 2: Dams are designed to withhold the highest water measured in the last 100 years

Safety integrity requirements

- Integrity depending on the mode of operation
  - Low demand mode: Average probability of failure to perform the desired function on demand
  - High demand (continuous) mode: Average rate of failure to perform the desired function
    (rate: failure per hour)
- High demand mode: Tolerable Hazard Rate (THR)

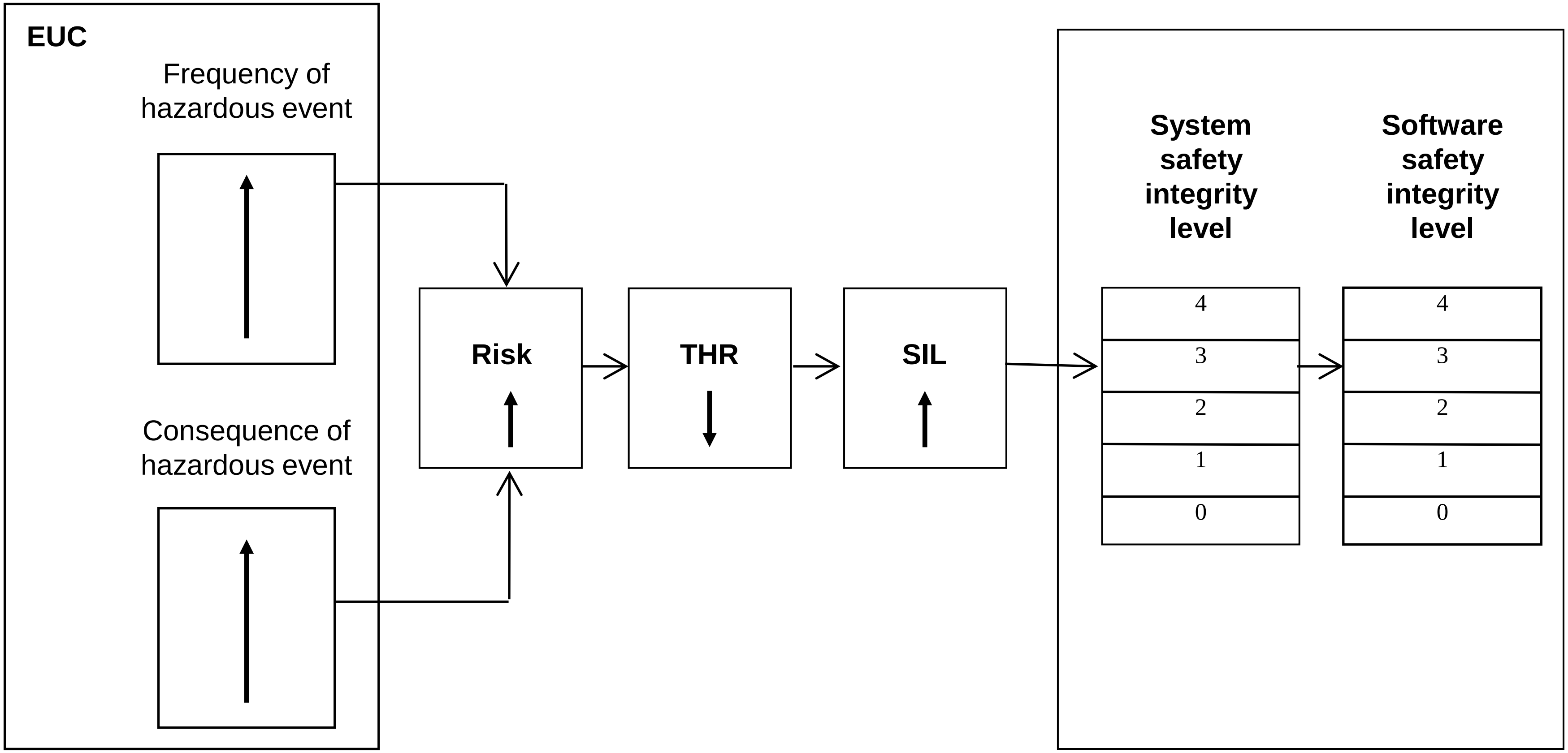
If the lifetime is 15 years then 1 equipment will fail out of the 750 equipments

SIL	Failure of a safety function per hour
1	$10^{-6} \leq THR < 10^{-5}$
2	$10^{-7} \leq THR < 10^{-6}$
3	$10^{-8} \leq THR < 10^{-7}$
4	$10^{-9} \leq THR < 10^{-8}$

Operation without failures in more than 11.000 years??

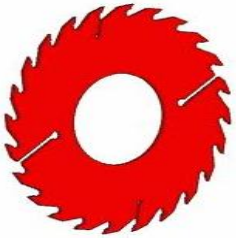
Determining SIL: Overview

- Hazard identification and risk analysis -> Target failure measure



Example: Safety requirements

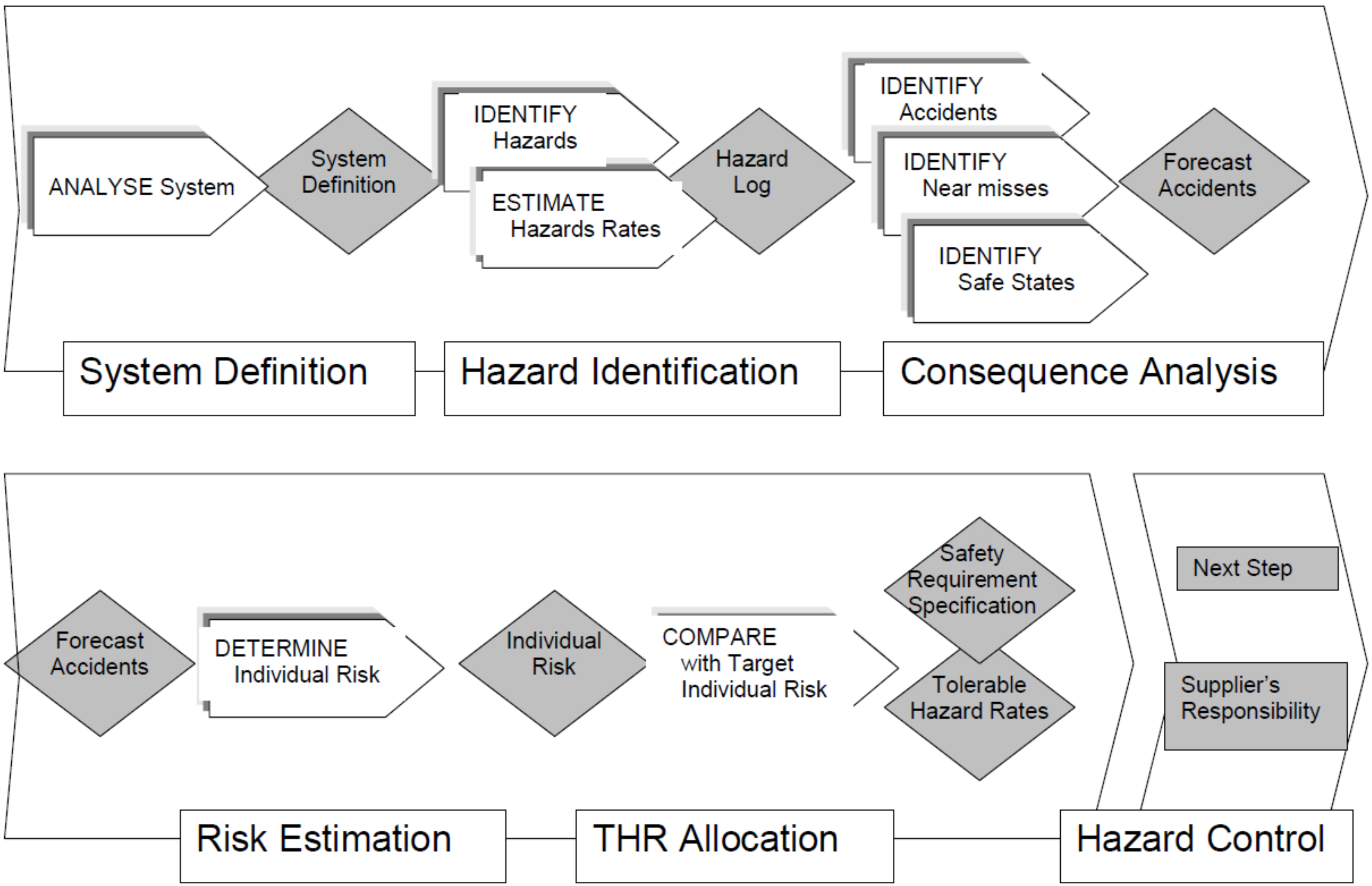
- Machine with a rotating blade and a solid cover
  - Cleaning of the blade: Lifting of the cover is needed
- Risk analysis: Injury of the operator
  when cleaning the blade while the motor is rotating
  - Hazard: If the cover is lifted more than 50 mm and
    the motor does not stop in 1 sec
  - There 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: Interlocking
  - Safety function requirement: When the cover is lifted to 15 mm,
    the motor shall be stopped and braked in 0.8 sec
- Safety integrity requirement:
  - The probability of failure of the interlocking (safety function) shall be less than 10^{-4} (one failure in 10.000 operation)

Requirements specification process

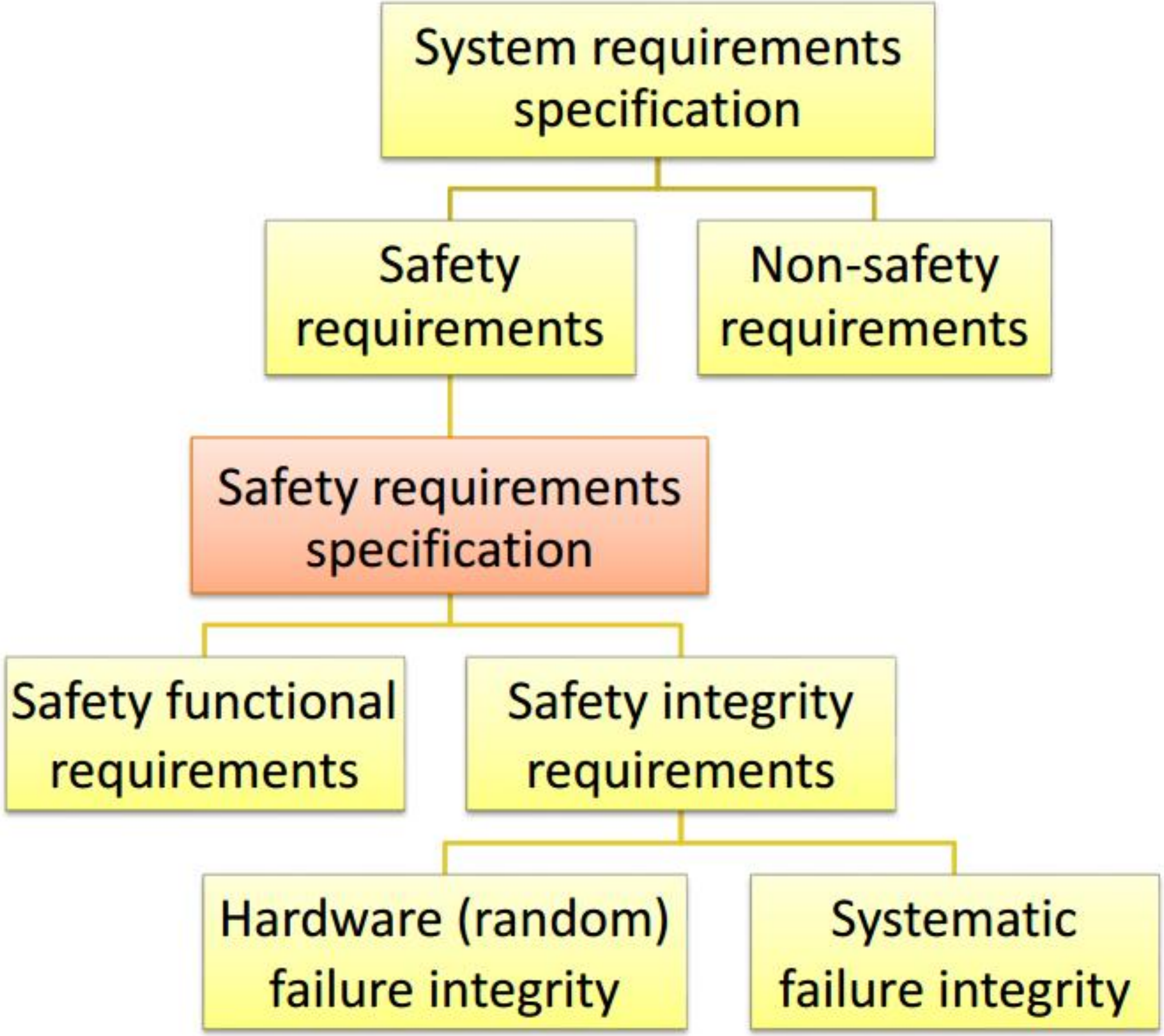
- Example: EN50129 (railway applications)



Satisfying safety integrity requirements

- Failures that influence safety integrity:
  - Random (hardware) failures: Occur accidentally at a random time due to degradation mechanisms
  - Systematic (software) failures: Occur in a deterministic way due to design / manufacturing / operating flaws
- Achieving safety integrity:
  - Random failure integrity: Selection of components (considering failure parameters) and the system architecture
  - Systematic failure integrity: Rigor in the development
    - Development life cycle: Well-defined phases
    - Techniques and measures: Verification, testing, measuring, …
    - Documentation: Development and operation
    - Independence of persons: Developer, verifier, assessor, …
- Safety case:
  - Documented demonstration that the product complies with the specified safety requirements

Summary: Structure of requirements



Dependability related requirements

(When safety is not enough)

Characterizing the system services

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

Threats to dependability

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Threats to dependability

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Development process:

- Better quality management, better methodology
- But: increasing complexity, difficulty in verification

Typical estimations for 1000 lines of code:

- Good development “by hand”: ~10 faults
- Tool-supported development: ~1-2 faults
- Application of formal methods: <1 faults

Threats to dependability

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Limits of the technology:

- Better quality control, better materials
- But: increasing sensitivity to environment effects

Typical estimations:

- CPU: 10^{-5}…10^{-6} faults/hour
- RAM: 10^{-4}…10^{-5} faults/hour
- LCD: ~ 2…3 years lifetime

Threats to dependability

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Verification during the development

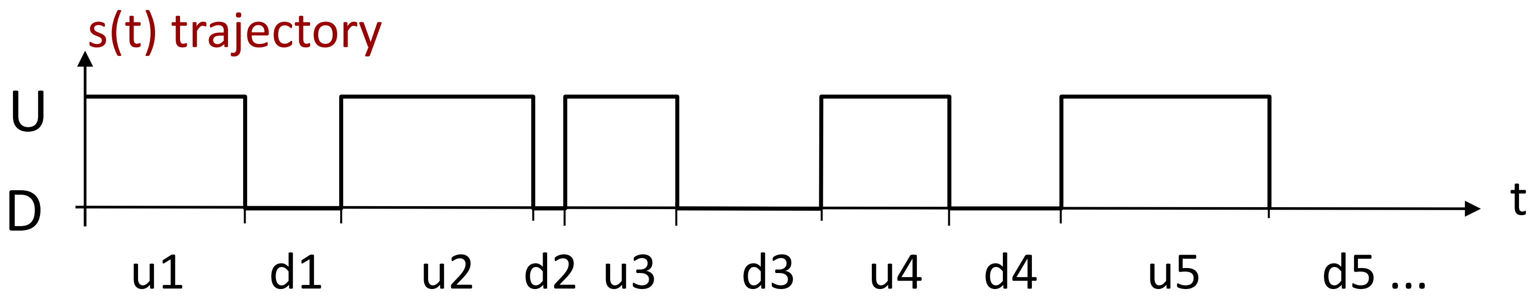
Fault tolerance during operation

Attributes of dependability

Attribute	Definition
Availability	Probability of correct service (considering repairs and maintenance) “Availability of the web service shall be 95%”
Reliability	Probability of continuous correct service (until the first failure) “After departure the onboard control system shall function correctly for 12 hours”
Safety	Freedom from unacceptable risk of harm
Integrity	Avoidance of erroneous changes or alterations
Maintainability	Possibility of repairs and improvements

Dependability metrics: Mean values

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



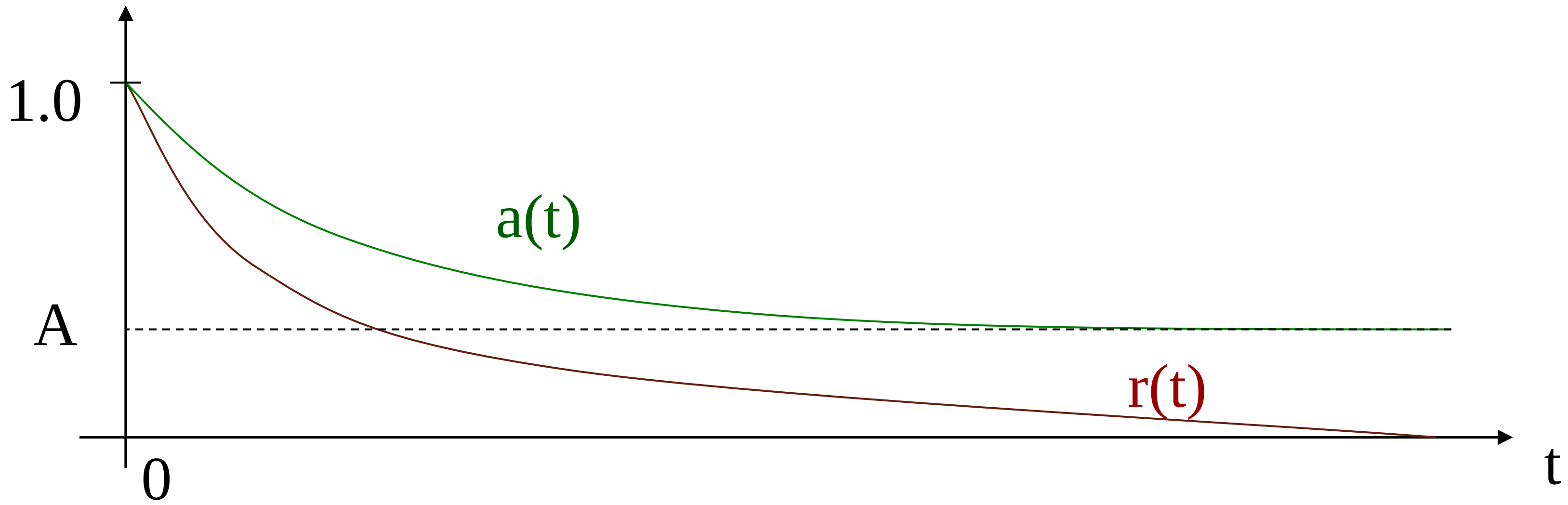
- Mean values:
  - Mean Time to First Failure: MTFF = E{u1}
  - Mean Up Time: MUT = MTTF = E{ui}
    (Mean Time To Failure)
  - Mean Down Time: MDT = MTTR = E{di}
    (Mean Time To Repair)
  - Mean Time Between Failures: MTBF = MUT + MDT

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: $r(t) = P\{s(t') \in U, \forall t' < t\}$



Availability related requirements

Availability	Failure period per year
99%	~ 3,5 days
99,9%	~ 9 hours
99,99% („4 nines”)	~ 1 hour
99,999% („5 nines”)	~ 5 minutes
99,9999% („6 nines”)	~ 32 sec
99,99999%	~ 3 sec

Availability of a system built up from components,
where the availability of single a component is 95%,
and all components are needed to perform the system function:

- Availability of a system built from 2 components: 90%
- Availability of a system built from 5 components : 77%
- Availability of a system built from 10 components : 60%

Attributes of components

- **Fault rate:** $\lambda(t)$

  Probability that the component will fail at time point **t** given that it has been correct until **t**

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

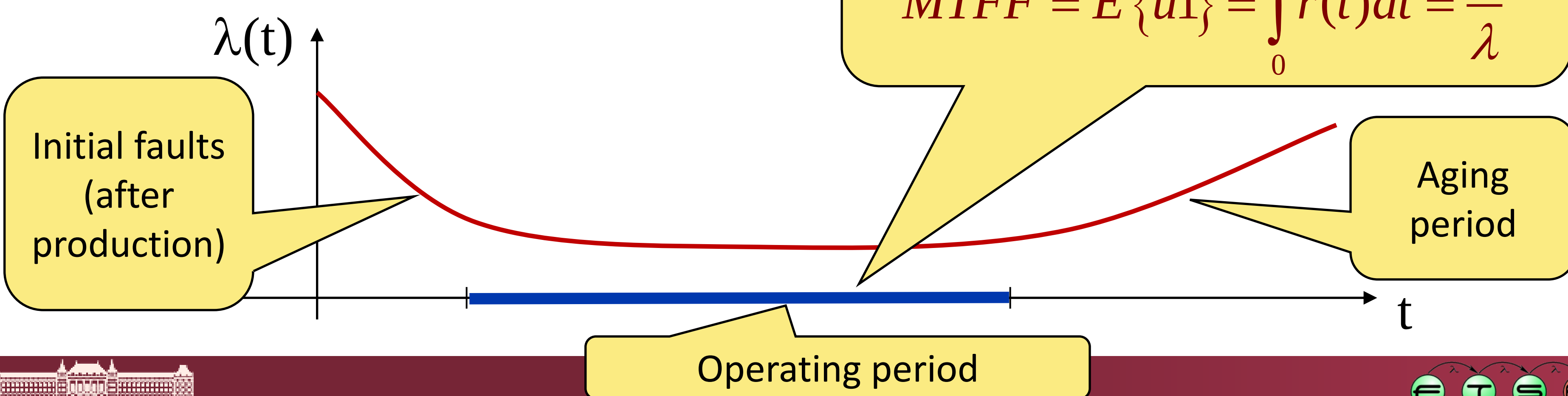
- Reliability of a component on the basis of this definition:

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

- For electronic components:

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

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



Case study: Development of a DMI

Driver

DMI

EVC

EVC: European Vital Computer (on board)

Maintenance centre

Characteristics:

- Safety-critical functions
  - Information visualization
  - Processing driver commands
  - Data transfer to EVC
- Safe wireless communication
  - System configuration
  - Diagnostics
  - Software update

Case study: DMI requirements

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

Threats to dependability

Fault:
adjudged or hypothesized cause of an error

→ **Component or system**

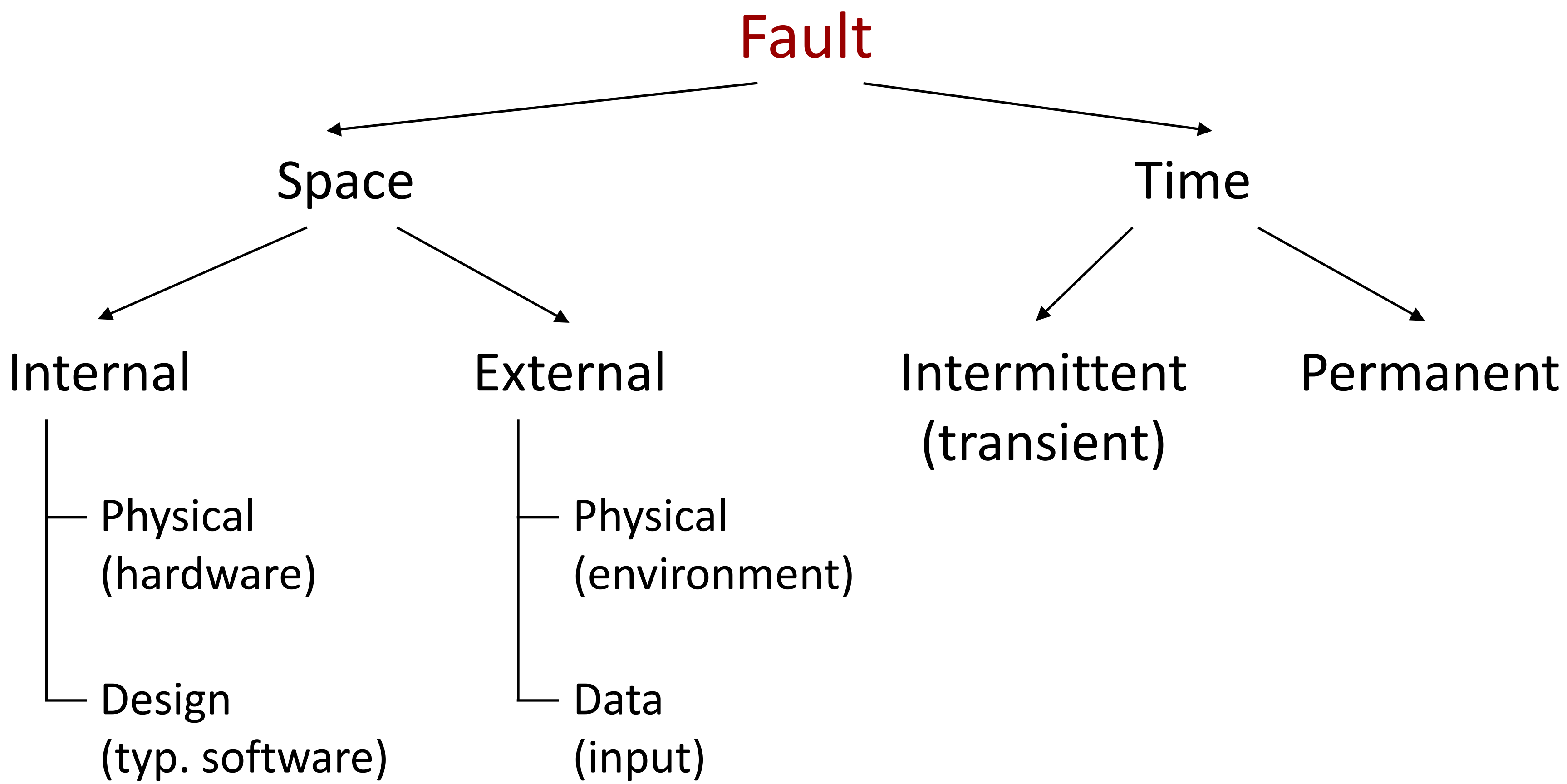
Error: State leading to the failure

→ **Failure**:
the delivered service deviates from correct service

Fault → Error → Failure examples:

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

The characteristics of faults



Software fault:

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

Means to improve dependability

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

    E.g., Self-Monitoring, Analysis and Reporting Technology (SMART)

Summary

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