Safety-critical systems: Basic definitions

Ákos Horváth

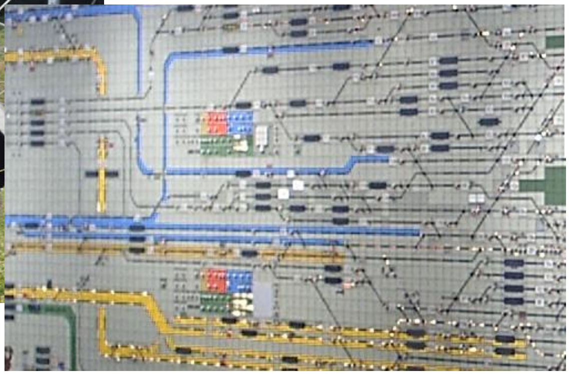
Based on István Majzik's slides

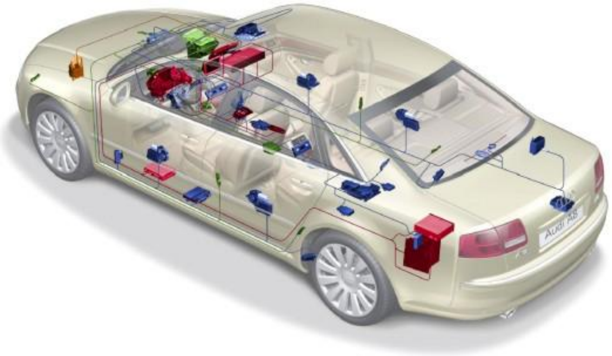
Dept. of Measurement and Information Systems

MŰEGYETEM 1782

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





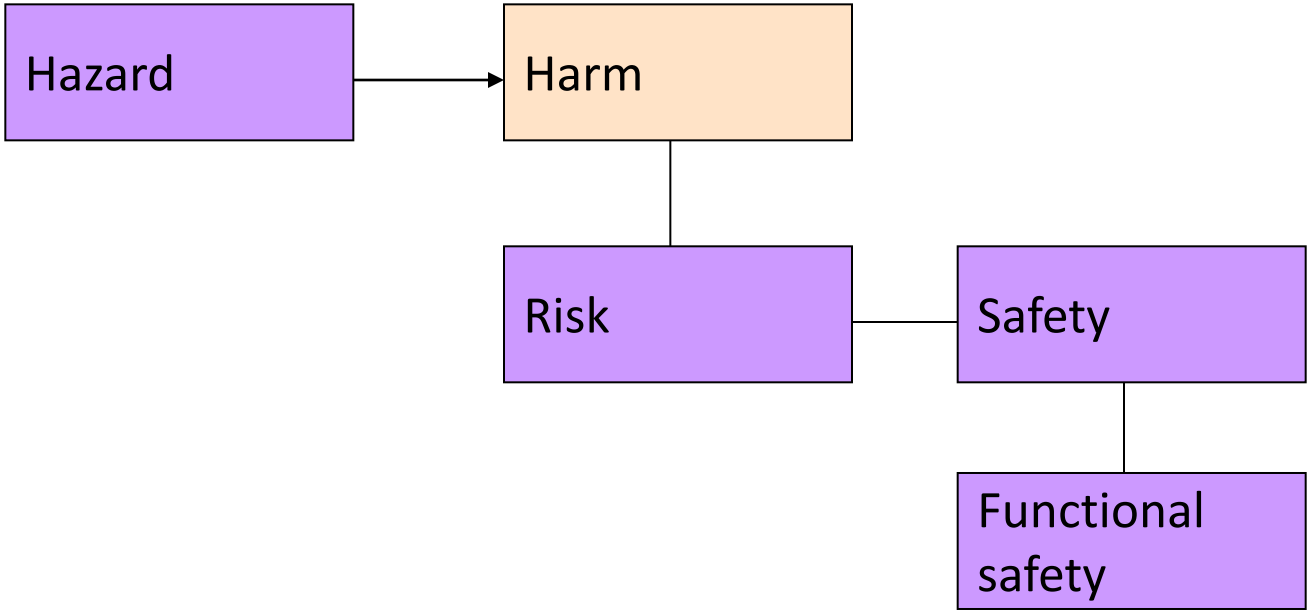
Railway signaling, x-by-wire, interlocking, emergency stopping, engine control, …

Specialities of safety critical systems

- Special solutions to achieve safe operation
  - Design: Requirements, architecture, tools, …
  - Verification, validation, and independent assessment
  - Certification (by safety authorities)
- Basis of certification: Standards
  - IEC 61508: Generic standard (for electrical, electronic or programmable electronic systems)
  - DO178B/C: Software in airborne systems and equipment
  - EN50129: Railway (control systems)
  - EN50128: Railway (software)
  - ISO26262: Automotive
  - Other sector-specific standards: Medical, process control, etc.

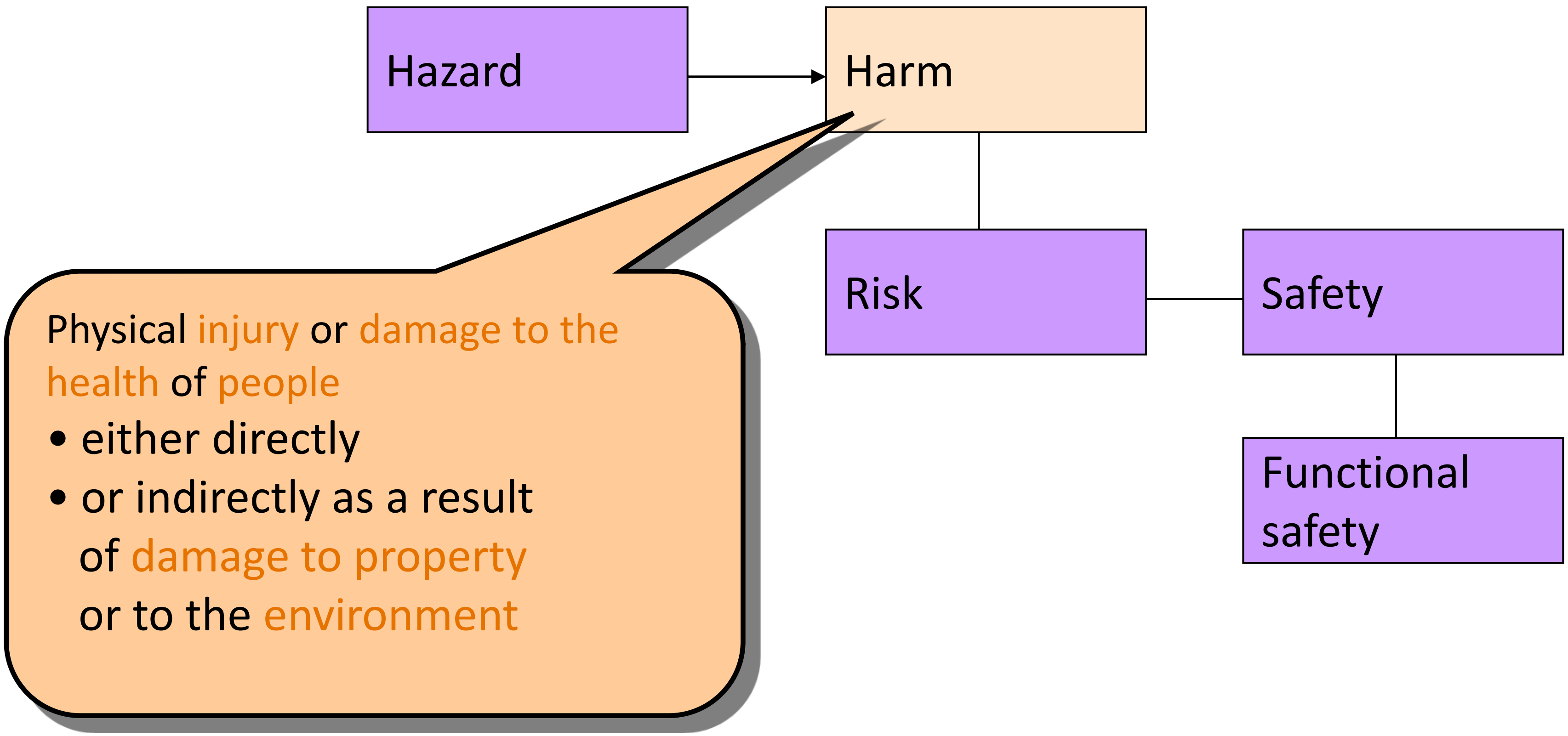
Definition of safety

- Central concepts: Hazard, risk and safety



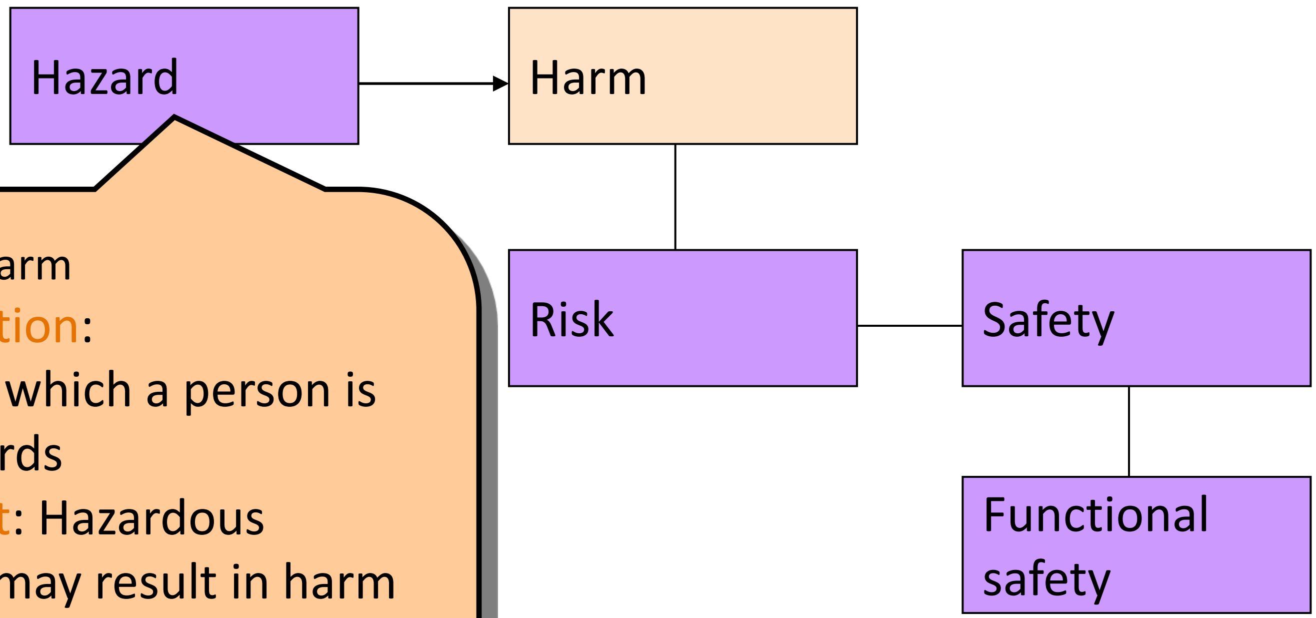
Definition of safety

- Central concepts: Hazard, risk and safety



Definition of safety

- Central concepts: Hazard, risk and safety

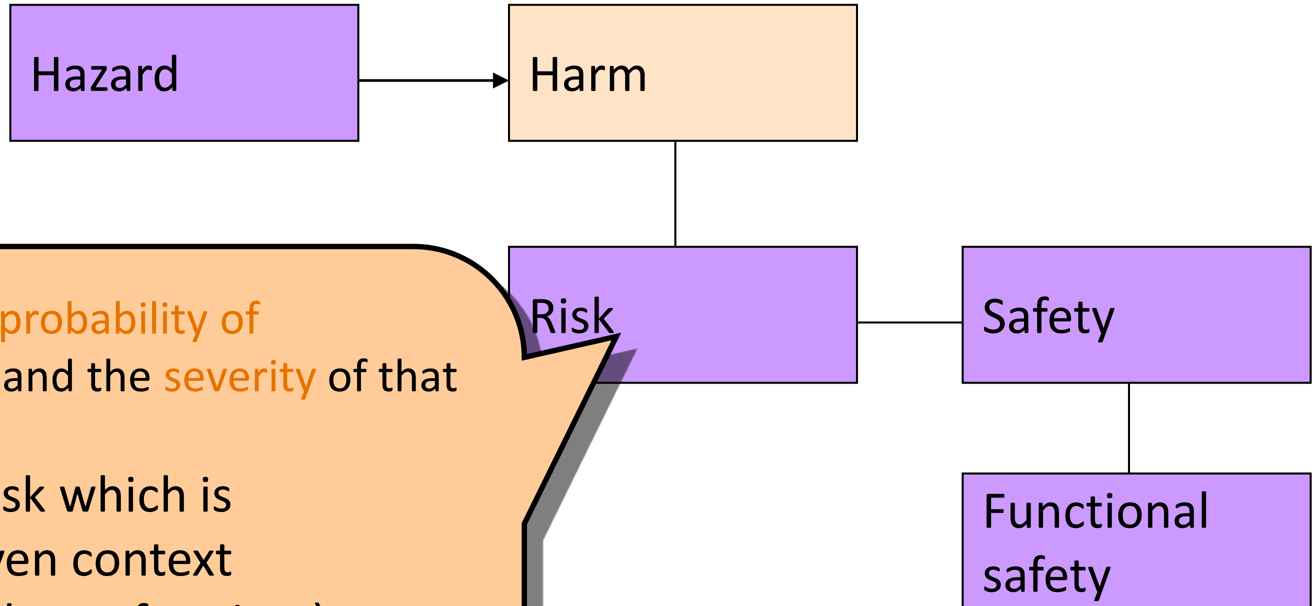


Potential cause of harm

- Hazardous situation: Circumstance in which a person is exposed to hazards
- Hazardous event: Hazardous situation which may result in harm
- Accident: Unintended event that results in harm
- Incident (near miss): Event that has the potential of harm

Definition of safety

- Central concepts: Hazard, risk and safety



Combination of the probability of occurrence of harm and the severity of that harm

- Tolerable risk: Risk which is accepted in a given context (based on the values of society)
- Residual risk: Risk remaining after protective measures have been taken

Definition of safety

- Central concepts: Hazard, risk and safety

Hazard → Harm

Safety

Functional safety

Freedom from unacceptable risk

Forms of safety in computer systems:

Primary safety:

- Dangers caused directly by the system itself (e.g., electric shock)

Functional safety:

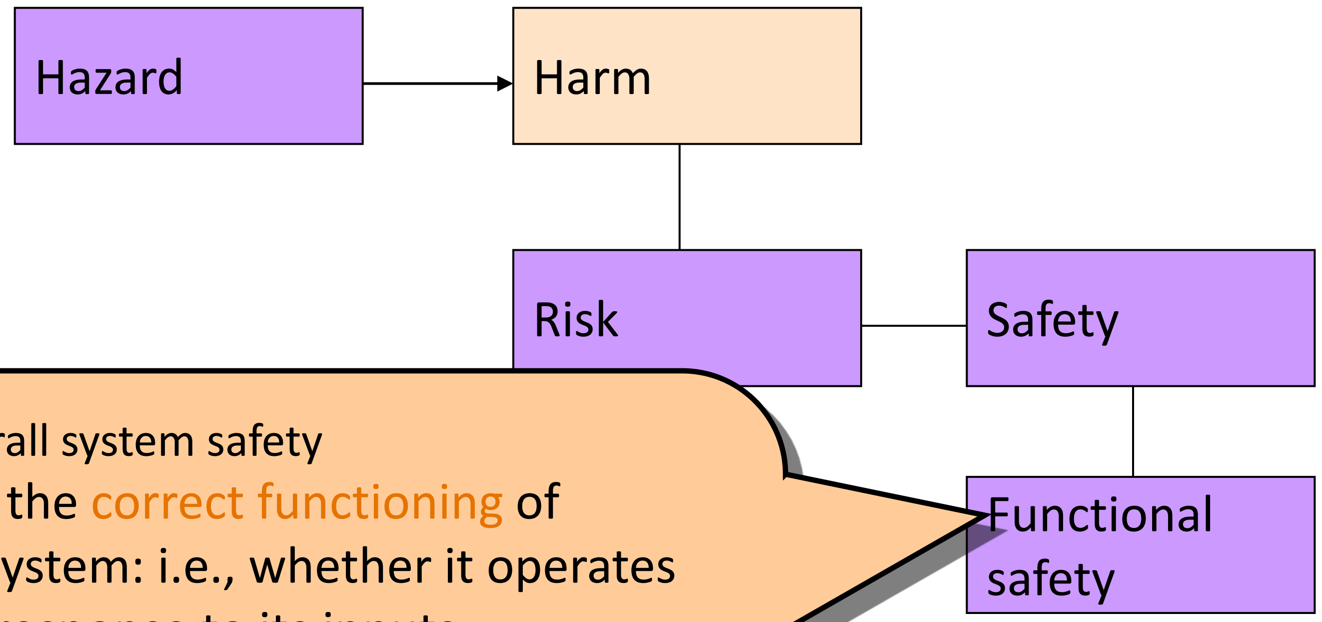
- This concerns the EUC controlled by the computer and is related to the correct functioning of the computer and software.

Indirect safety:

- This relates to the indirect consequences of a computer failure or the production of incorrect information.

Definition of safety

- Central concepts: Hazard, risk and safety

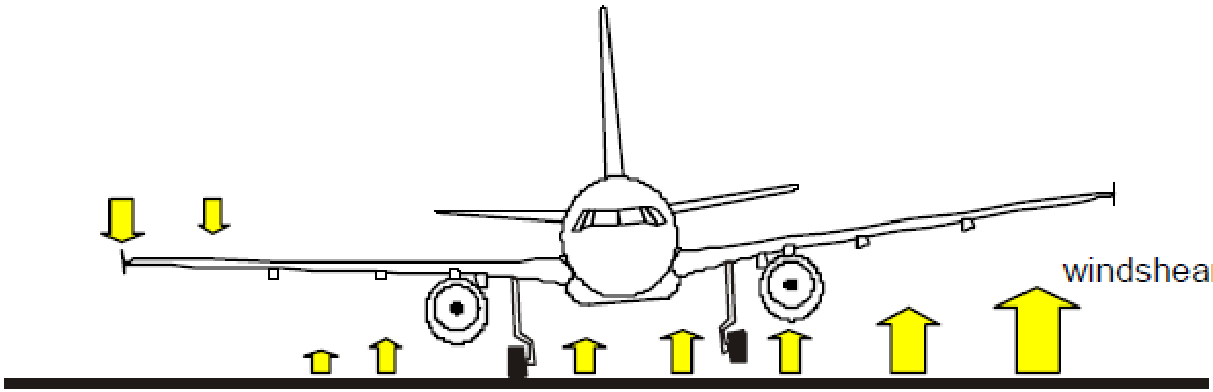


Part of the overall system safety

- depends on the correct functioning of the E/E/PE system: i.e., whether it operates correctly in response to its inputs
- depends on other technology safety-related systems
- depends on external risk reduction facilities

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!

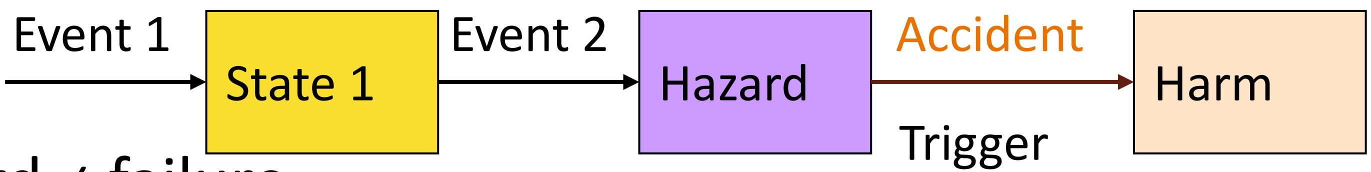


Accident examples

- 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)

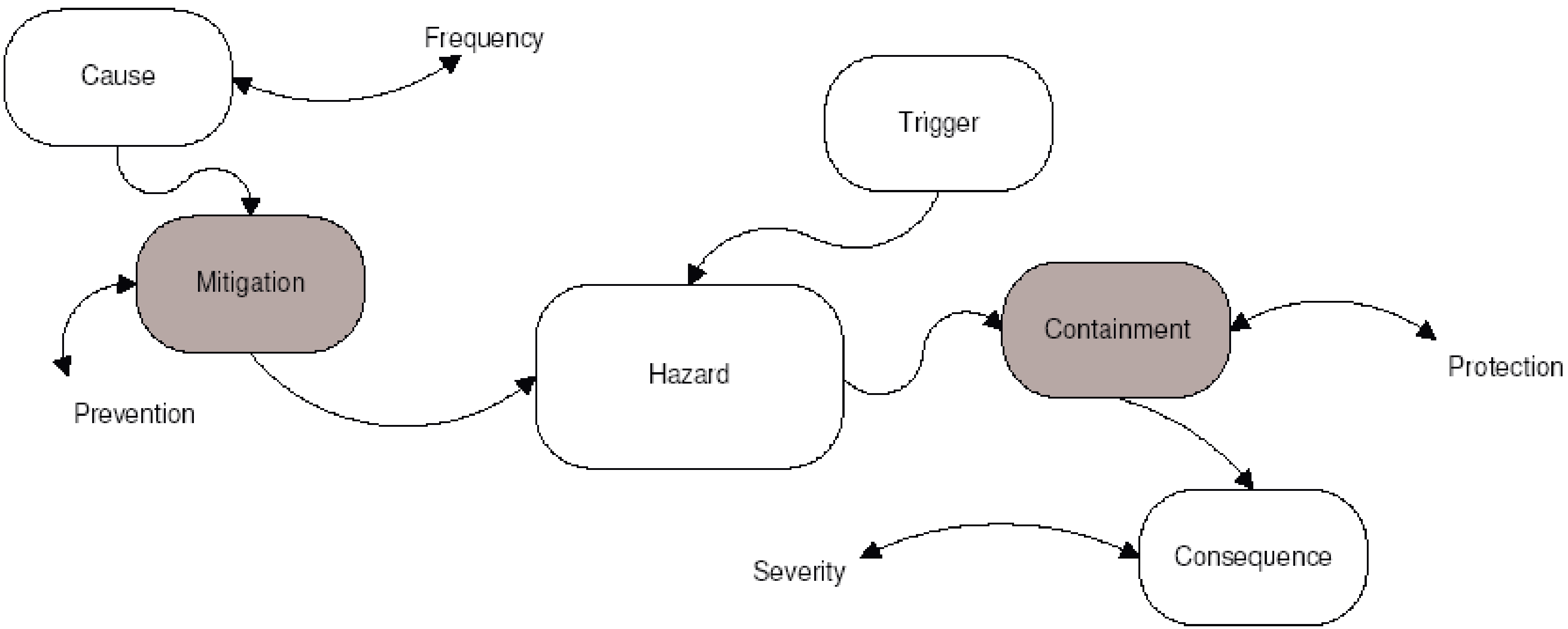
Experiences

- 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
  - Hazard may also be caused by (unexpected) combination of normal events
- Central problems in safety-critical systems:
  - Analysis of hazards
  - Assignment of functions to avoid hazards → accidents → harms

Hazard control



- Risk characteristics:
  - Frequency of occurrence
  - Severity of its consequence
- Mitigation: Eliminate or decrease the chance of a hazard
- Containment: Reduce the consequence of a hazard

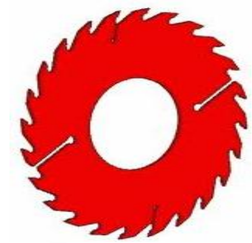
Safety-related system

- Safety function:
  - Function which is intended to achieve or maintain a safe state for the EUC
- Safety-related system:
  - Implements the required safety functions necessary to achieve or maintain a safe state for the EUC,
  - and is intended to achieve the necessary safety integrity for the required safety functions
- Requirements for a safety-related system:
  - What is the safety function: Safety function requirements
  - What is the likelihood of the correct operation of the safety function: Safety integrity requirements

Safety integrity

- Safety integrity:
  - Probability of a safety-related system satisfactorily performing the required safety functions (i.e., without failure)
    - under all stated conditions
    - within a stated period of time
- Types of safety integrity:
  - Random (hardware): Related to random hardware failures
    - Occur at a random time due to degradation mechanisms
  - Systematic: Related to systematic failures
    - Failures related in a deterministic way to faults that can only be eliminated by modification of the design / manufacturing process / operation procedure / documentation / other relevant factors
- Safety integrity level (SIL):
  - Discrete level for specifying safety integrity requirements of the safety functions (i.e., probabilities of failures)

Example: Safety function



- Machine with a rotating blade
  - Blade is protected by a hinged solid cover
- Cleaning of the blade: Lifting of the cover is needed
- Hazard analysis: Avoiding injury of the operator when cleaning the blade
  - If the cover is lifted more than 5 mm then the motor should be stopped
  - The motor should be stopped in less than 1 sec
- Safety function: Interlocking
  - When the cover is lifted to 4 mm, the motor is stopped and braked in 0,8 s
- Safety integrity:
  - The probability of failure of the interlocking (safety function) shall be less than 10^{-4} (one failure in 10.000 operation)
  - Failure of interlocking is not necessarily result in an injury since the operator may be careful

Safety and dependability

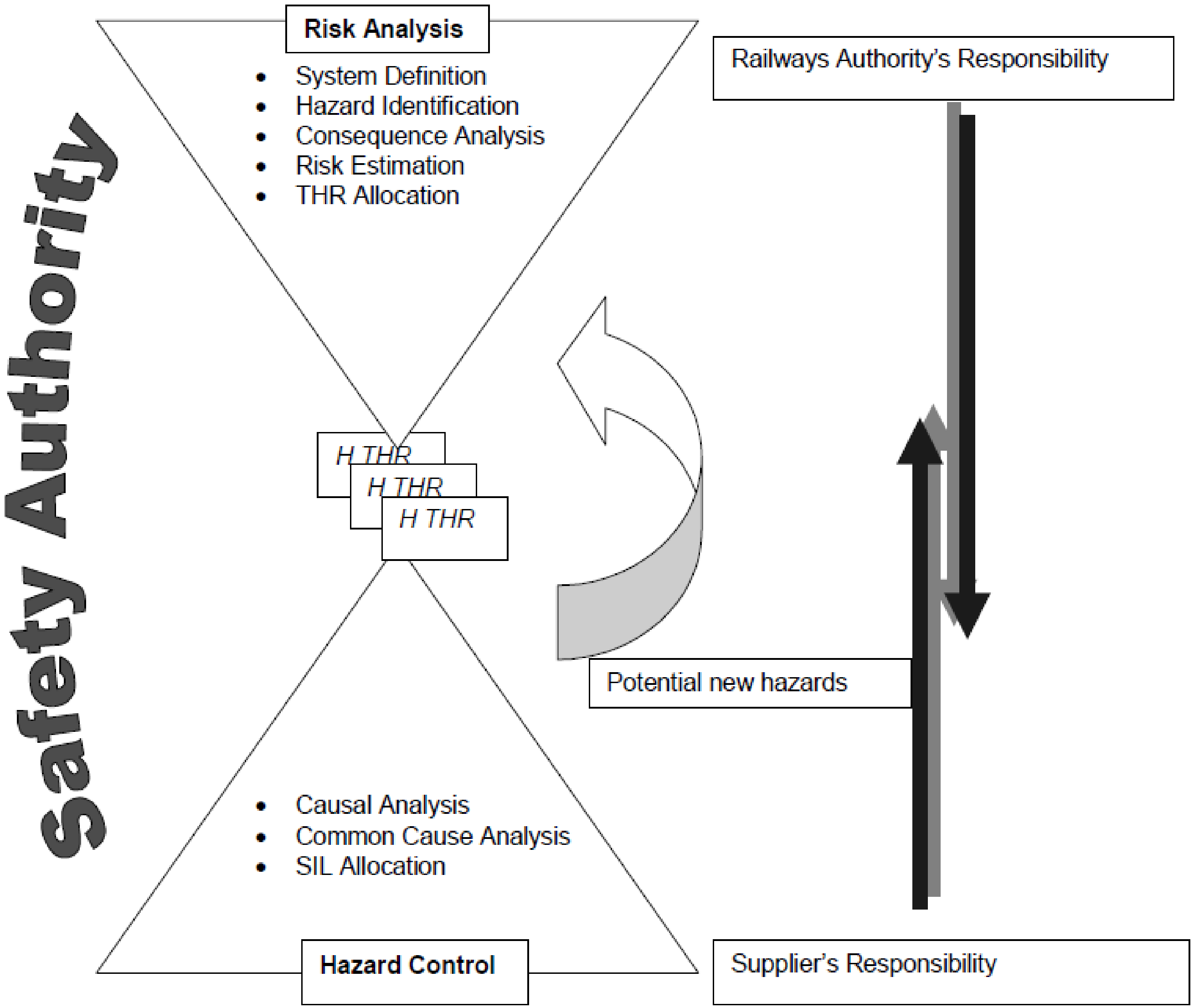
- Safety vs. reliability:
  - Fail-safe state: safe, but 0 reliability
    - Railway signaling, red state: Safety ≠ reliability
    - Airplane control: Safety = reliability
- Safety vs. availability:
  - Fail-stop state: safe, but 0 availability (and reliability)
  - High availability may result in (short) unsafe states

Safety requirements

- Requirements for a safety-related system:
  - Safety function requirements:
    - Derived from hazard identification
  - Safety integrity requirements:
    - Related to target failure measure of the safety function
    - Derived from risk estimation: Acceptable risk
- Safety standards: Risk based approach for determining target failure measure
  - Tolerable risk: Risk which is accepted in a given context based on the current values of society
  - It is the result of risk analysis
    - Performed typically by the customer
    - Considering the environment, scenarios, mode of operation, …

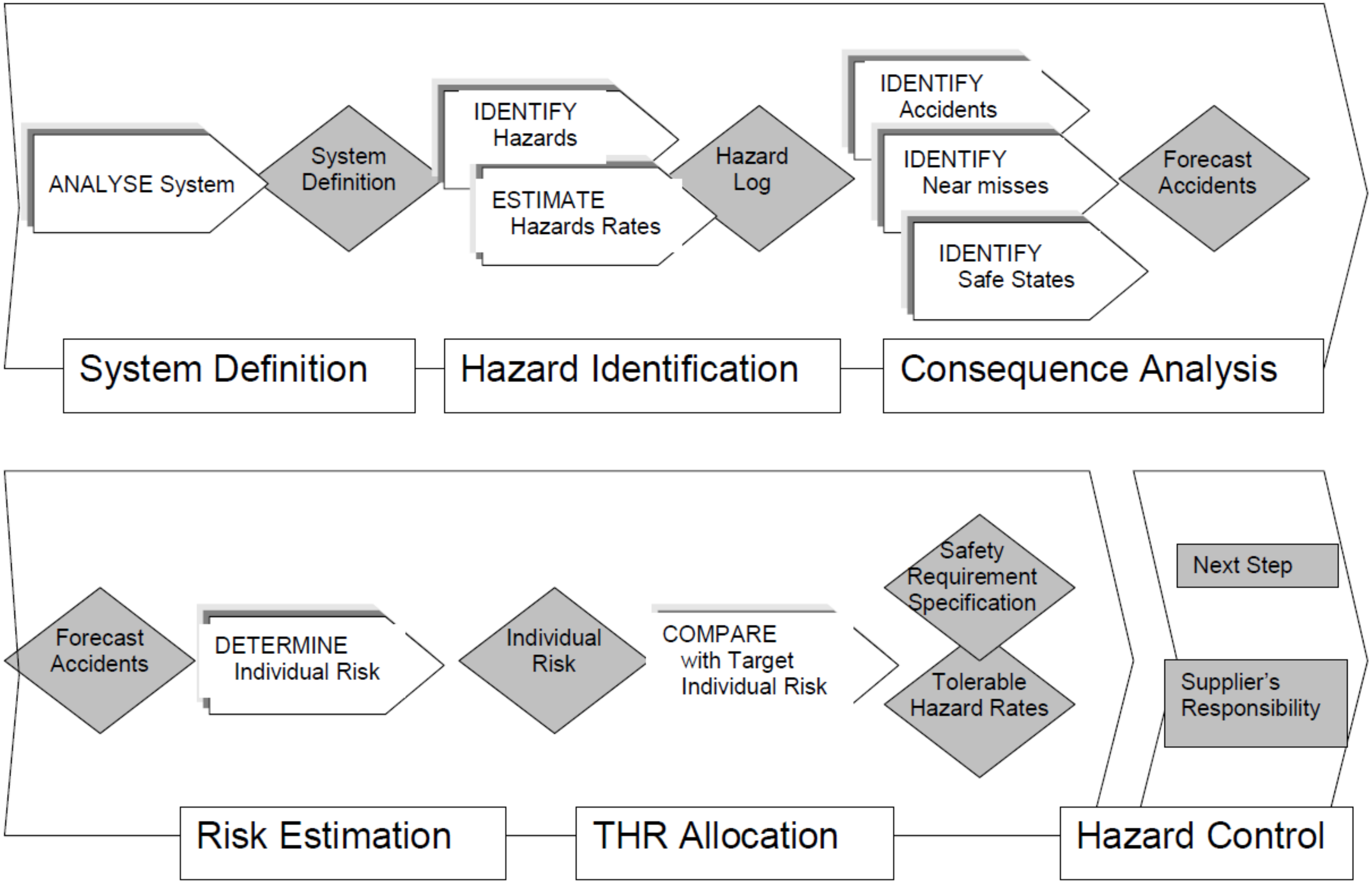
Risk based approach

- EN50129: Railway applications
- THR: Tolerable hazard rate (continuous operation)



Risk analysis

- EN50129 (railway applications)



Mode of operation

- Way in which a safety-related system is to be used:
  - Low demand mode: Frequency of demands for operation is
    - no greater than one per year and
    - no greater than twice the proof-test frequency
  - High demand (or continuous) mode: Frequency of demands for operation is
    - greater than one per year or
    - greater than twice the proof-test frequency
- Target failure measure:
  - Low demand mode: Average probability of failure to perform the desired function on demand
  - High demand mode: Probability of a dangerous failure per hour
    - Acceptable risk -> Tolerable hazard rate (THR)

Safety integrity requirements

- Low demand mode:

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

- High demand mode:

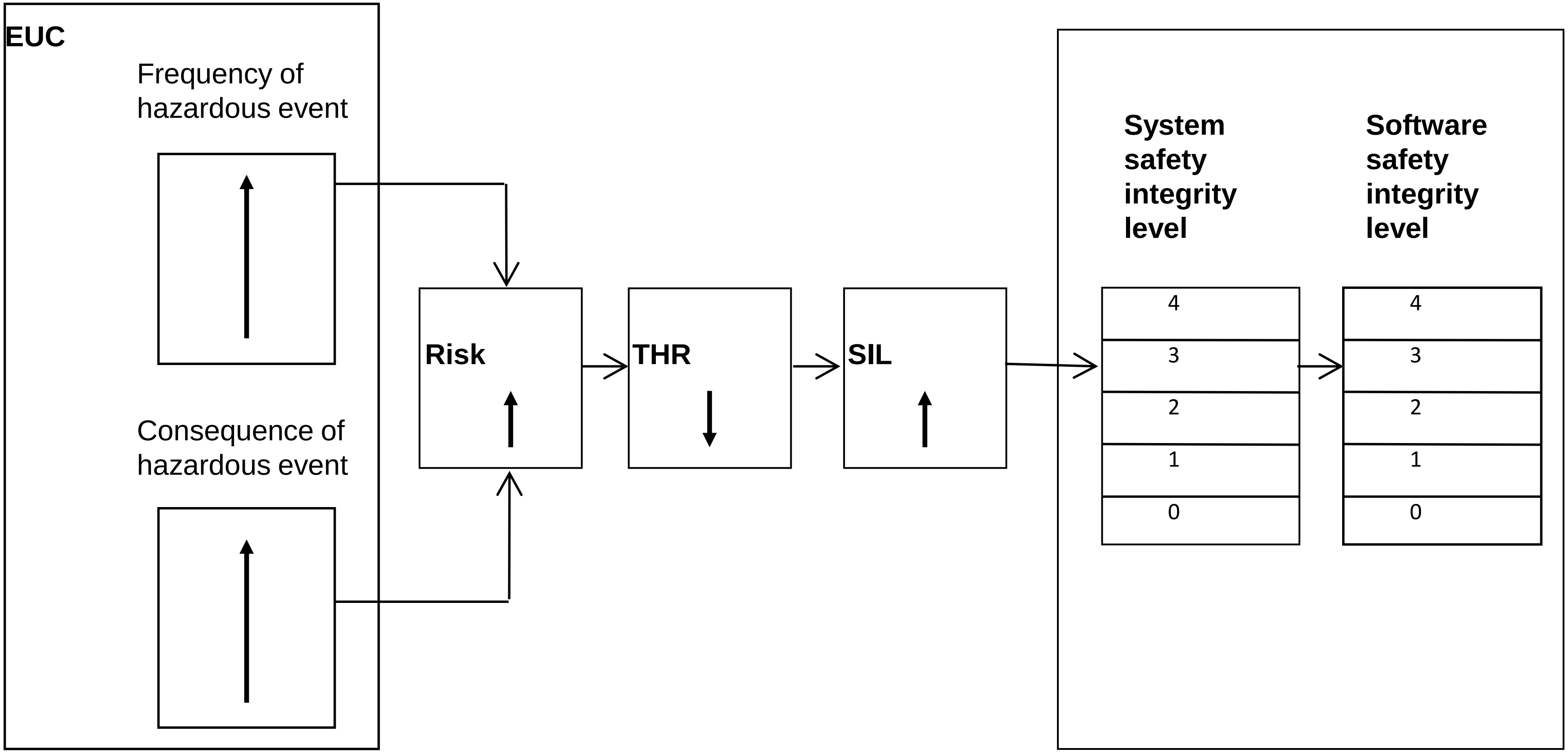
SIL	Probability of dangerous failure per hour per safety function
1	$10^{-6} \leq \text{PFH} < 10^{-5}$
2	$10^{-7} \leq \text{PFH} < 10^{-6}$
3	$10^{-8} \leq \text{PFH} < 10^{-7}$
4	$10^{-9} \leq \text{PFH} < 10^{-8}$

(PFH or THR)

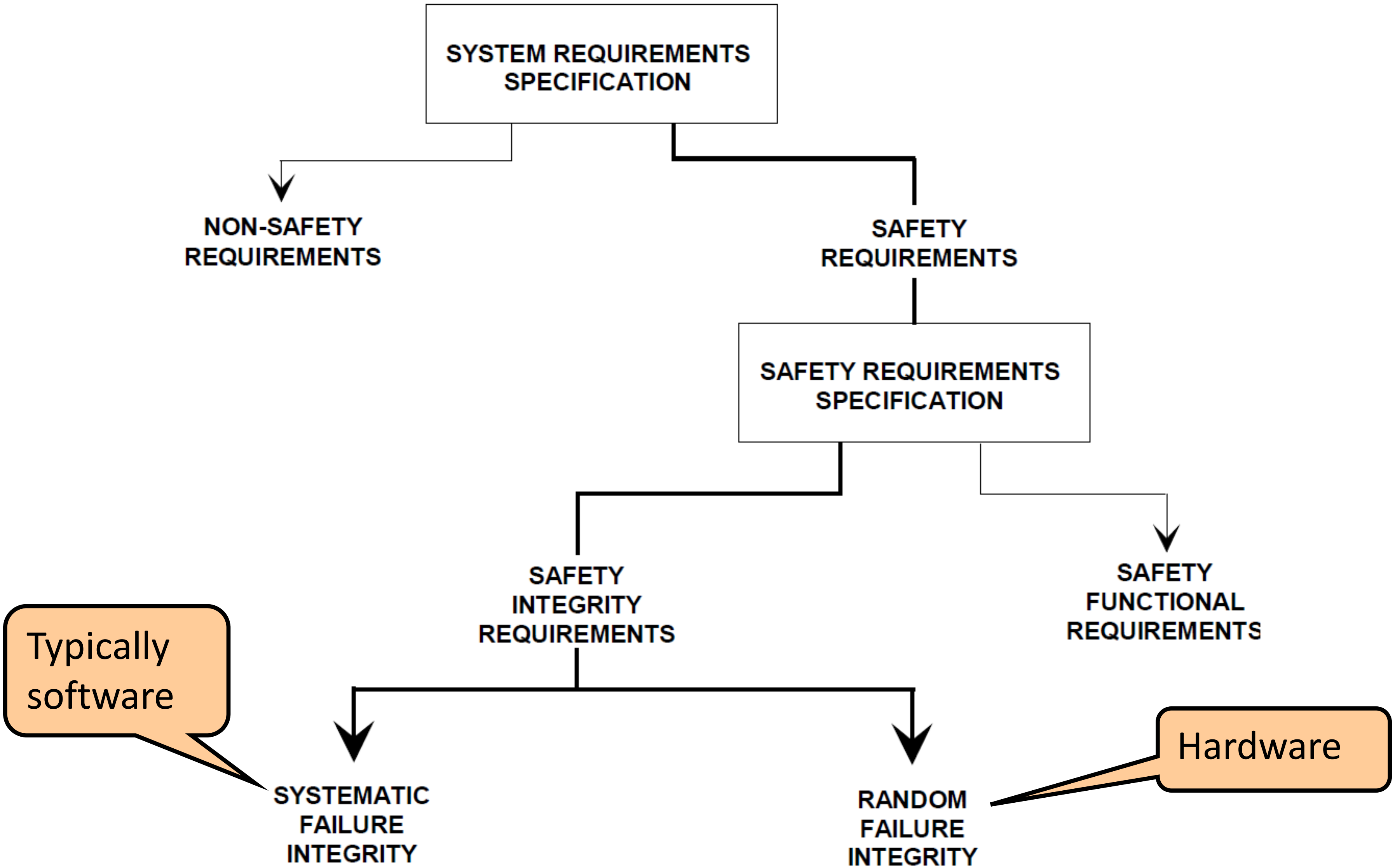
15 years lifetime:
1 failure in case of
750 equipment

Determining SIL: Overview

- Hazard identification and risk analysis -> Target failure measure



Structure of requirements



Challenges in achieving functional safety

- E/E/PE systems: Complexity
  - Impossible to determine every failure mode
  - Difficult to predict safety performance
- Preventing/controlling dangerous failures resulting from
  - Incorrect specification (system, HW, SW)
  - Omissions in safety requirement specification
  - Hardware failure mechanisms: Random or systematic
  - Software failure mechanisms: Systematic
  - Common cause failures
  - Human (operator) errors
  - Environmental influences (e.g., temperature, EM, mechanical)
  - Supply system disturbances (e.g., power supply)
  - …

Demonstrating SIL requirements

- Approaches:
  - Random failure integrity:
    - Quantitative approach: Based on statistics, experiments
  - Systematic failure integrity:
    - Qualitative approach: Rigor in the engineering
      - Development life cycle
      - Techniques and measures
      - Documentation
      - Independence of persons
- Safety case:
  - Documented demonstration that the product complies with the specified safety requirements
  - Systematic demonstration

Summary of the basic concepts

System safety

- emphasizes building in safety, not adding it to a completed design
- deals with systems as a whole rather than with subsystems or components
- takes a larger view of hazards than just failures
- emphasizes analysis rather than past experience and standards
- emphasizes qualitative rather than quantitative approaches

Dependability related requirements

(Safety is not enough)

Characterizing the system services

- Typical characteristics of services:
  - Reliability, availability, integrity, ...
  - These depend on the failures during the use of the services (the good quality of the production process is not enough)
- 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?

Fault effects

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Fault effects

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

Fault effects

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

Fault effects

Development process → **Product in operation**

- Design faults
- Implementation faults

- Hardware faults
- Configuration faults
- Operator faults

Verification during the development

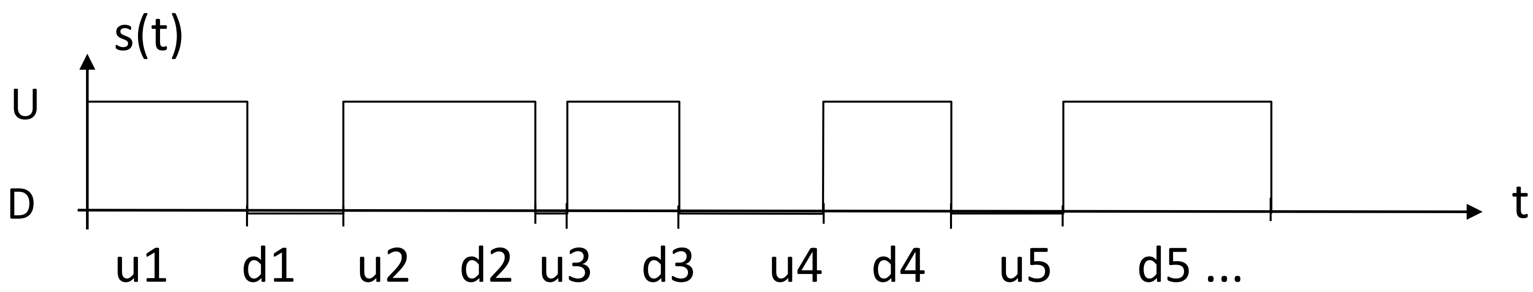
Fault tolerance during operation

Dependability and security

- Basic attributes of dependability:
  - Availability: Probability of correct service (considering repairs and maintenance)
  - Reliability: Probability of continuous correct service (until the first failure)
  - Safety: Freedom from unacceptable risk of harm
  - Integrity: Avoidance of erroneous changes or alterations
  - Maintainability: Possibility of repairs and improvements
- (Attributes of security:)
  - Availability
  - Integrity
  - Confidentiality: absence of unauthorized disclosure of information

Dependability metrics: Mean values

- Partitioning the state 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{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:

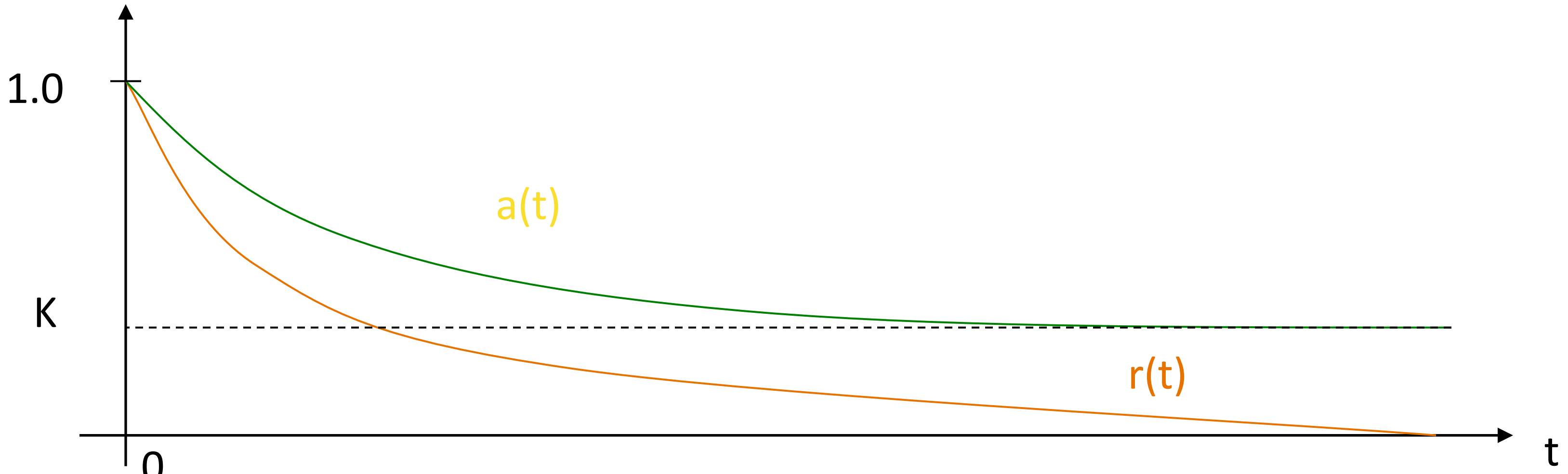
$$a(t) = P\{s(t) \in U\}$$ (failures may occur)

- Reliability:

$$r(t) = P\{s(t') \in U, \forall t' < t\}$$ (no failure until t)

- Asymptotic availability: $K = \lim_{t \to \infty} a(t)$ (regular repairs)

  In other way: $K = A = \mathrm{MTTF} / (\mathrm{MTTF} + \mathrm{MTTR})$



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 a component is 95%:

- 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 density that the component will fail at time point t given that it has been correct until t

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

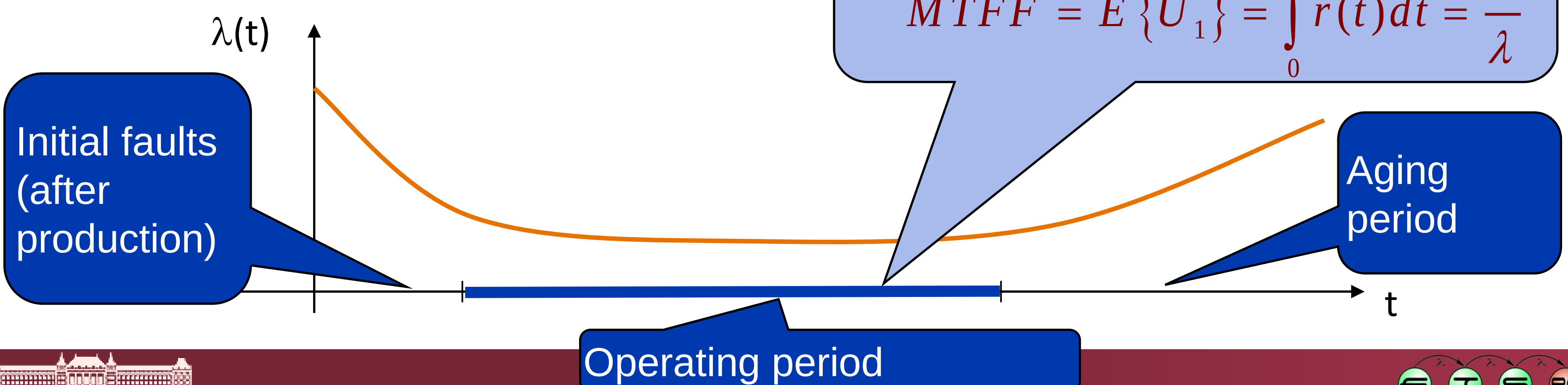
  - In other way (on the basis of the definition of reliability):

$$\lambda(t) = -\frac{1}{r(t)}\frac{dr(t)}{dt}, \quad \text{thus} \quad r(t) = e^{-\int_0^t \lambda(t)dt}$$

  - For electronic components:

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

$$MTFF = E\{U_1\} = \int_0^\infty r(t)dt = \frac{1}{\lambda}$$



Case study: development of a DMI

Driver

DMI

EVC

EVC: European Vital Computer (on board)

Maintenance centre

Characteristic:

- 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
  - CENELEC standards: EN 50129 and EN 50128
- 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
    MTTR < 24 hours if MTTF=5000 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

Fault

- Space
  - Internal
    - Physical (hardware)
    - Design (typ. software)
  - External
    - Physical (environment)
    - Data (input)
- Time
  - Intermittent (transient)
  - Permanent

Software fault:

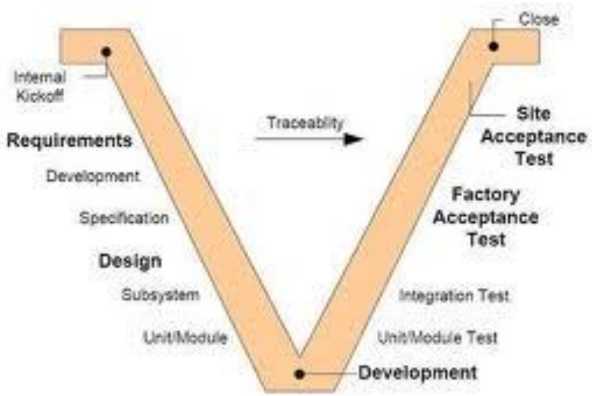
- Permanent design fault (systematic)
- Activation of the fault depends on the operational profile (inputs)

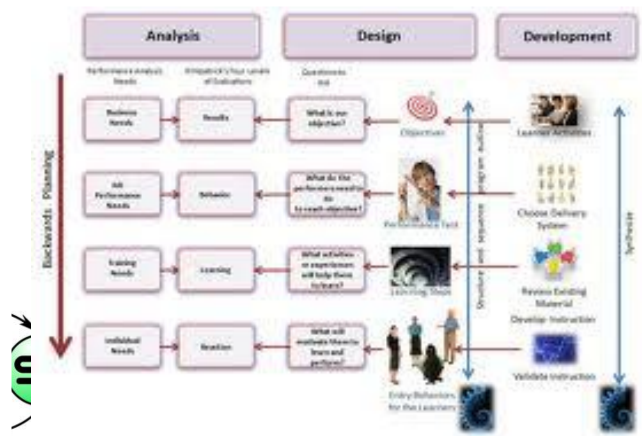
Means to improve dependability

- 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)

Overview of the development of safety-critical systems



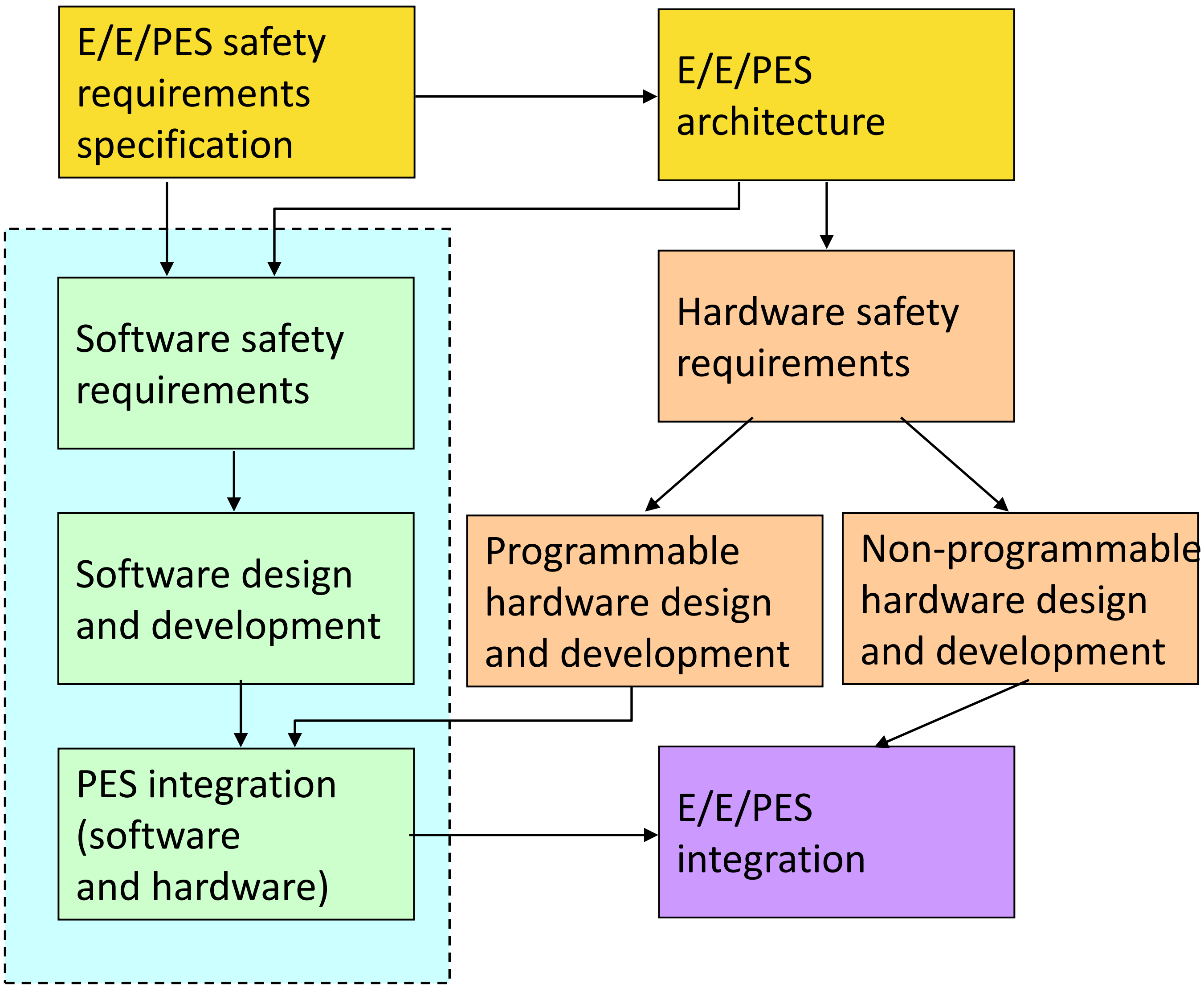


Overall safety lifecycle model: Goals

- Technical framework for the activities necessary for ensuring functional safety
- Covers all lifecycle activities
  - Initial concept
  - Hazard analysis and risk assessment
  - Specification, design, implementation
  - Operation and maintenance
  - Modification
  - Final decommissioning and/or disposal

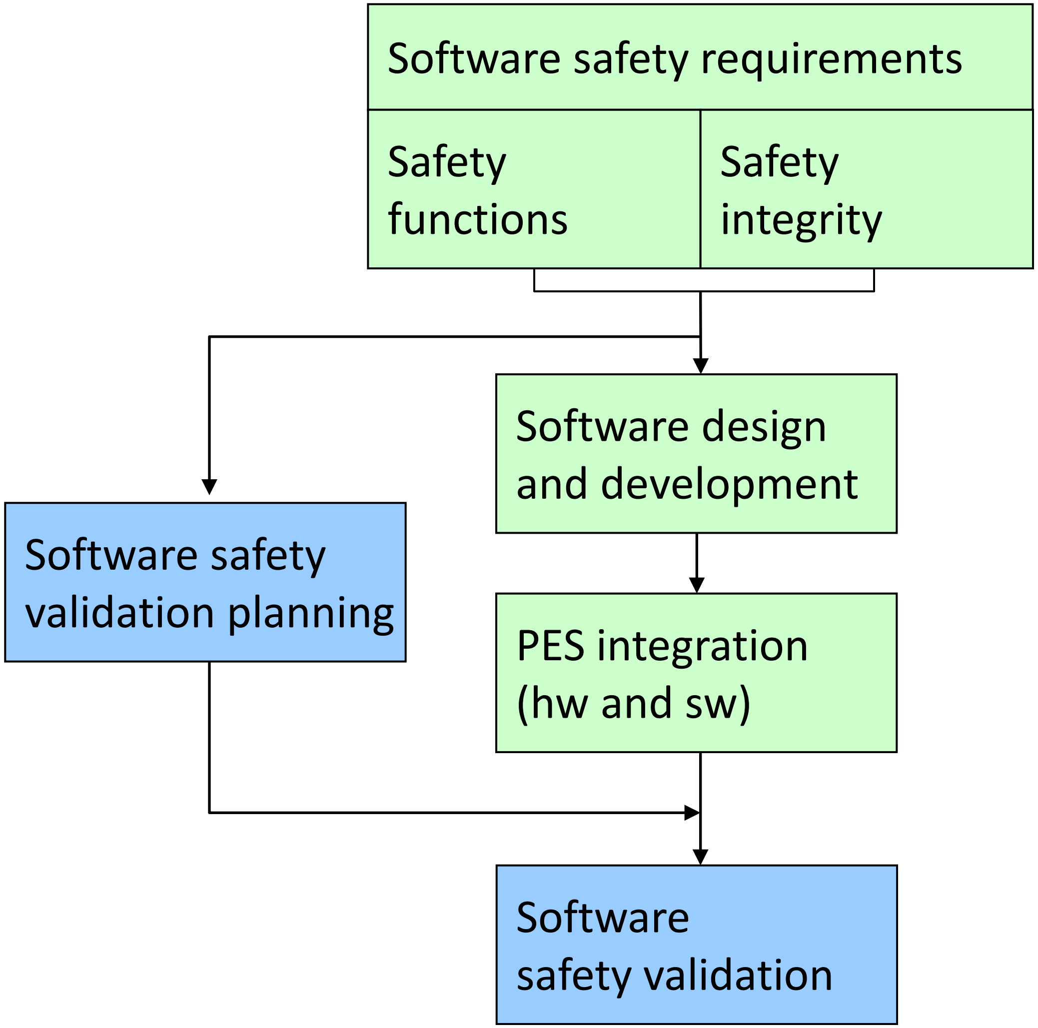
Hardware and software development

- PE system architecture (partitioning of functions) determines software requirements
- PES integration follows software development
- Final step: E/E/PES integration

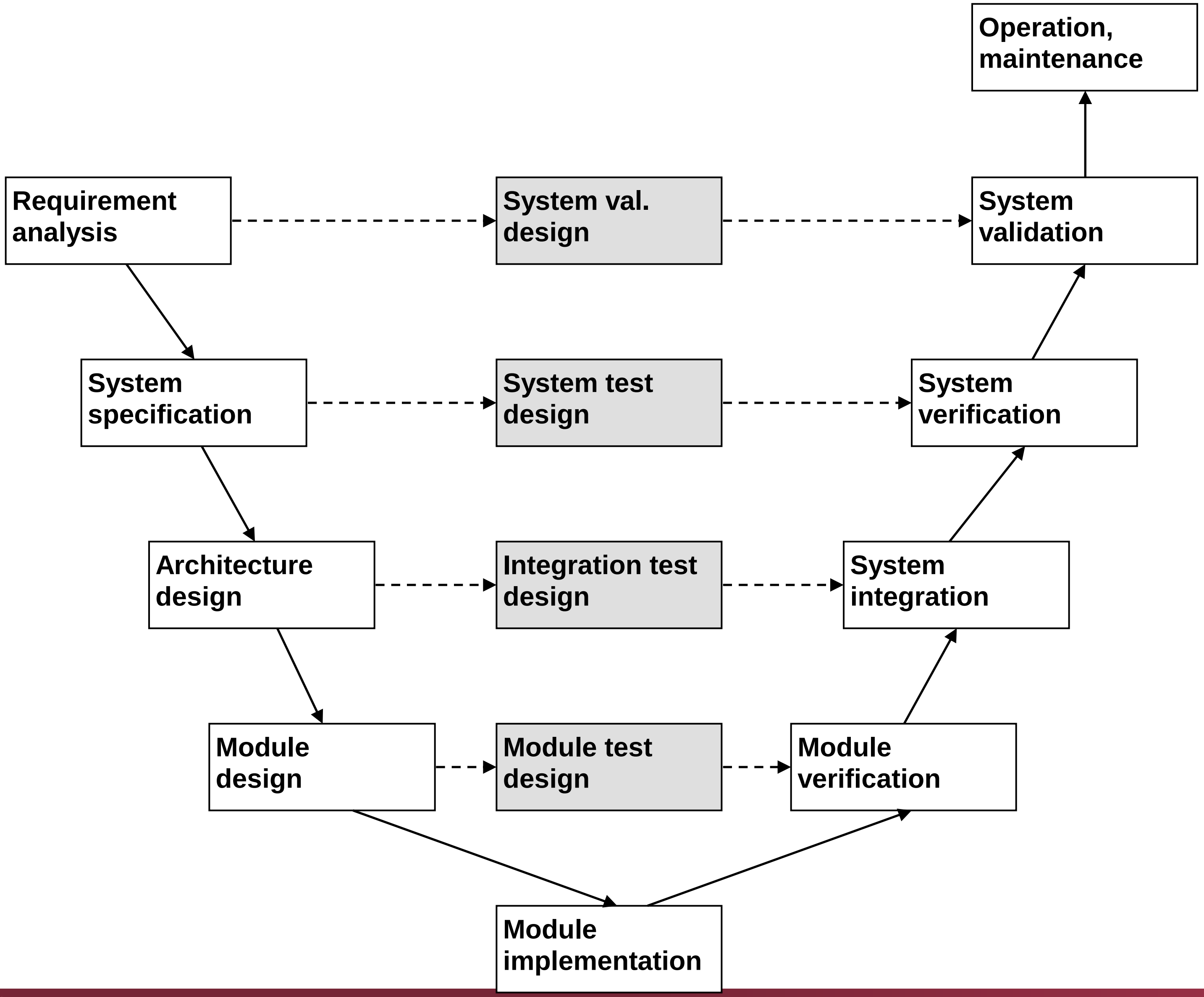


Software safety lifecycle

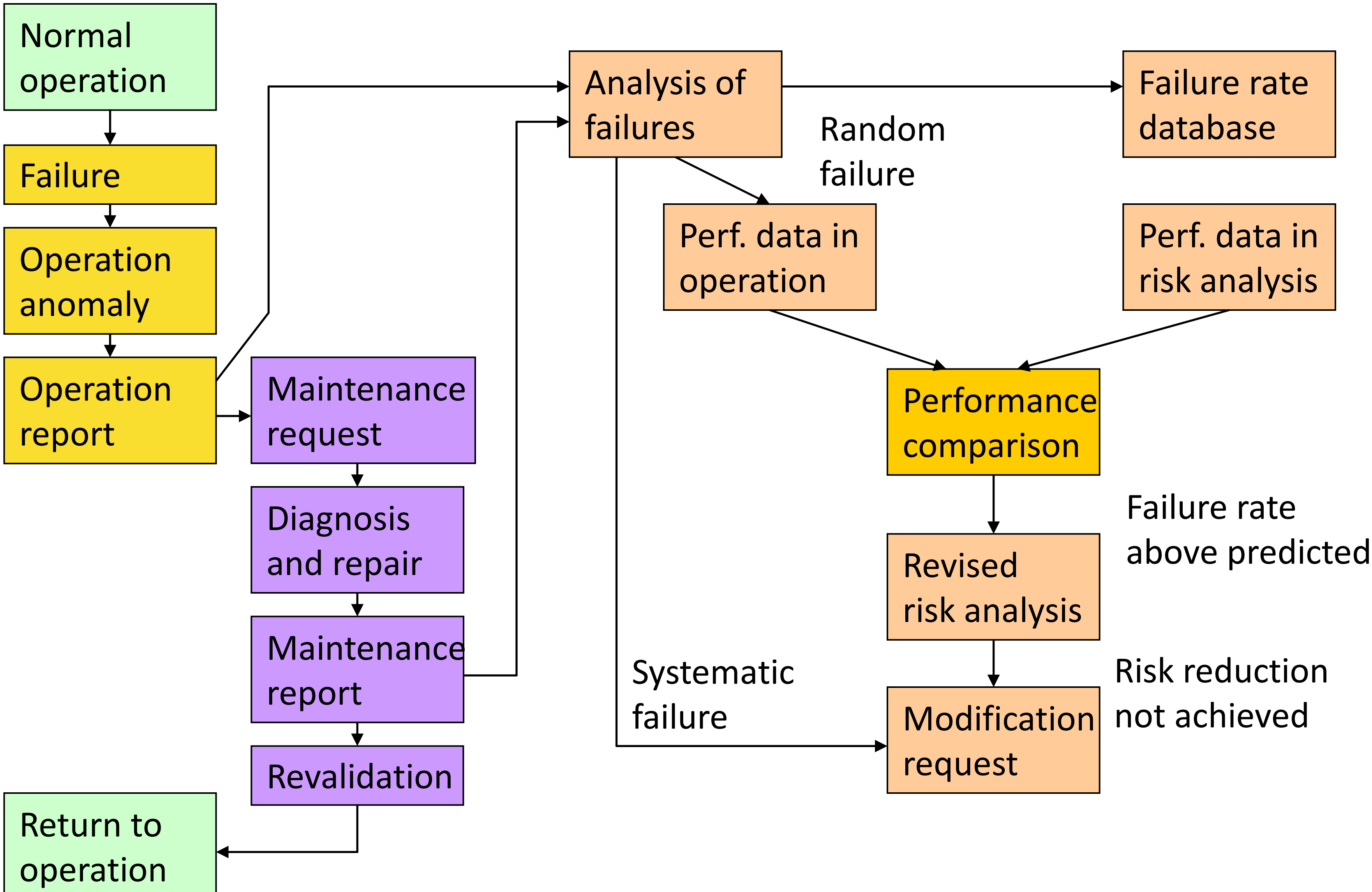
- Safety req. spec. has two parts:
  - Software safety functions
  - Software safety integrity levels
- Validation planning is required
- Integration with PE hardware is required
- Final step: Software safety validation



Example software lifecycle (V-model)



Maintenance activities



Techniques and measures: Basic approach

- Goal: Preventing the introduction of systematic faults and 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., alternate or equivalent techniques are marked
- Hierarchy of methods is formed (references to tables)

Example: Guide to selection of techniques

- Software safety requirements specification:
  - Techniques 2a and 2b are alternatives

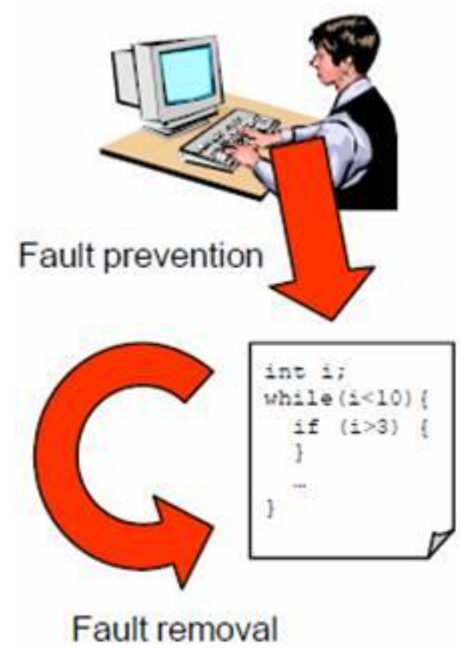
	Technique/Measure*	Ref.	SIL1	SIL2	SIL3	SIL4
1	Computer-aided specification tools	B.2.4	R	R	HR	HR
2a	Semi-formal methods	Table B.7	R	R	HR	HR
2b	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR

  - Referred table: Semi-formal methods (B.7)

	Technique/Measure*	Ref	SIL1	SIL2	SIL3	SIL4
1	Logic/function block diagrams	see note below	R	R	HR	HR
2	Sequence diagrams	see note below	R	R	HR	HR
3	Data flow diagrams	C.2.2	R	R	R	R
4	Finite state machines/state transition diagrams	B.2.3.2	R	R	HR	HR
5	Time Petri nets	B.2.3.3	R	R	HR	HR
6	Decision/truth tables	C.6.1	R	R	HR	HR

Application of tools in the lifecycle

- **Fault prevention**:
  - Program translation from high-level programming languages
  - MBD, CASE tools: High level modeling and code/configuration generators
- **Fault removal**:
  - Analysis, testing and diagnosis
  - Correction (code modification)



Management tools

- Contributing both to fault prevention and removal
- Includes project management, configuration management, issue tracking

Safety concerns of tools

- Types of tools
  - Tools potentially introducing faults
    - Modeling and programming tools
    - Program translation tools
  - Tools potentially failing to detect faults
    - Analysis and testing tools
    - Project management tools
- Requirements
  - Use certified or widely adopted tools
    - “Increased confidence from use” (no evidence of improper results yet)
  - Use the well-tested parts without altering the usage
  - Check the output of tools (analysis/diversity)
  - Control access and versions

Safety of programming languages

- Factors for selection of languages
  - Functional characteristics (probability of faults)
    - Logical soundness (unambiguous definition)
    - Complexity of definition (understandability)
    - Expressive power
    - Verifiability (consistency with specification)
    - Vulnerability (security aspects)
  - Availability and quality of tools
  - Expertise available in the design team

- Coding standards (subsets of languages) are defined
  - “Dangerous” constructs are excluded (e.g., function pointers)
  - Static checking can be used to verify the subset
- Specific (certified) compilers are available
  - Compiler verification kit for third-party compilers

Safety of programming languages

- Factors
  - Func
    - L
    - C
    - E
    - V
    - V
  - Avai
  - Expe
- Coding
  - "Dangerous ... are excluded (e.g., function pointers)
  - Static checking can be used to verify the subset
- Specific (certified) compilers are available
  - Compiler verification kit for third-party compilers

Constructs that make verification difficult (61508):
- Unconditional jumps excluding subroutine calls
- Recursion
- Pointers, heaps or any type of dynamic variables
- Interrupt handling at source code level
- Multiple entries and exits of loops and subprograms
- Implicit variable initialization or declaration
- Variant records and equivalence
- Procedural parameters

Language comparison

	Structured assembler	C	CORAL 66	ISO Pascal	Modula-2	Ada
Wild jumps	*	?	?	?	?	*
Overwrites	?	X	X	?	?	?
Semantics	?	X	?	?	*	?
Model of math	?	X	?	*	*	?
Operational arithmetic	?	X	X	?	?	?
Data typing	?	X	?	?	?	*
Exception handling	X	?	X	X	?	*
Safe subset?	?	X	X	X	?	X
Exhaustion of memory	*	?	?	?	?	X
Separate compilation	X	X	?	?	*	*
Well understood?	*	?	?	*	*	?

X – not provided (may lead to unsafe behaviour)
? – some protection, but risk remains
* - sound protection is provided

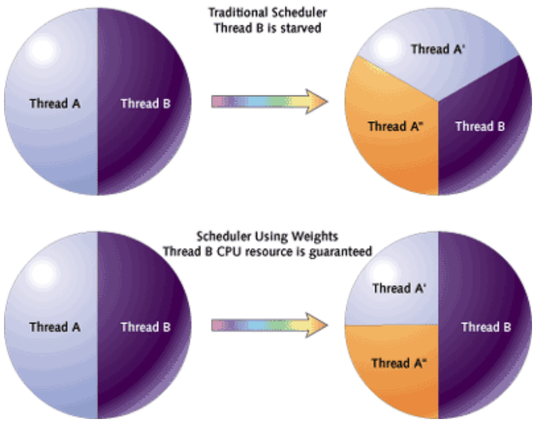
[Cullyer+1991]

Wild jumps: Jump to arbitrary address in memory
Overwrites: Overwriting arbitrary address in memory
Model of math: Well-defined data types
Separate compilation: Type checking across modules

Coding standards for C and C++

- MISRA C (Motor Industry Software Reliability Association)
  - Safe subset of C (2004): 141 rules (121 required, 20 advisory)
  - Examples:
    - Rule 33 (Required): The right hand side of a "&&" or "||" operator shall not contain side effects.
    - Rule 49 (Advisory): Tests of a value against zero should be made explicit, unless the operand is effectively Boolean.
    - Rule 59 (R): The statement forming the body of an "if", "else if", "else", "while", "do ... while", or "for" statement shall always be enclosed in braces.
    - Rule 104 (R): Non-constant pointers to functions shall not be used.
  - Tools to check “MISRA conformance” (LDRA, PolySpace, …)
    - Test cases to demonstrate adherence to MISRA rules
- MISRA C++ (2008): 228 rules
- US DoD, JSF C++: 221 rules (incl. metric guidelines)
  - “Joint Strike Fighter Air Vehicle C++ Coding Standard”

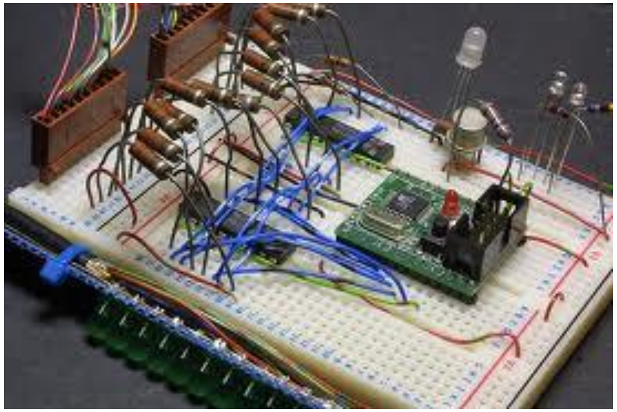
Safety-critical OS: Required properties



- Partitioning in space
  - Memory protection
  - Guaranteed resource availability
- Partitioning in time
  - Deterministic scheduling
  - Guaranteed resource availability in time
- Mandatory access control for critical objects
  - Not (only) discretionary
- Bounded execution time
  - Also for system functions
- Support for fault tolerance and high availability
  - Fault detection and recovery / failover
  - Redundancy control

Example: Safety and RTOS

- Compromise needed
  - Complex RTOS:
    - Difficult to test
  - “Bare machine”:
    - Less scheduling risks
    - High maintenance risks



- Example: Tornado® for Safety Critical Systems
  - Integrated software solution uses Wind River's securely partitioned VxWorks® AE653 RTOS
  - ARINC 653: Time and space partitioning (guaranteed isolation)
  - RTCA/DO-178B: Level A certification
  - POSIX, Ada, C support

Principles for documentation

- Type of documentation
  - Comprehensive (overall lifecycle)
    - E.g., Software Verification Plan
  - Specific (for a given lifecycle phase)
    - E.g., Software Source Code Verification Report
- Document Cross Reference Table
  - Determines documentation for a lifecycle phase
  - Determines relations among documents
- Traceability of documents is required
  - Relationships between documents are specified (input, output)
  - Terminology, references, abbreviations are consistent
- Merging documents is allowed
  - If responsible persons (authors) shall not be independent

Document cross reference table (EN50128)

- ■ creation of a document
- ◆ used document in a given phase (read vertically)

clause / title	8 SRS	9 SA	10 SDD	11 SVer	12 S/H I	13 SVal	14 Ass	15 Q	16 Ma	
PHASES *(*)=in parallel with other phases*										**DOCUMENTS**
SW REQUIREMENTS	■	◆	◆	◆	◆	◆	◆			Sw Requirements Specification
	■			◆	◆	◆	◆			Sw Requirements Test Specification
				■						Sw Requirements Verification Report
SW DESIGN		■	◆	◆	◆	◆	◆			Sw Architecture Specification
			■	◆	◆	◆	◆			Sw Design Specification
				■						Sw Arch. and Design Verification
SW MODULE DESIGN			■	◆	◆	◆	◆			Sw Module Design Specification
			■	◆	◆	◆	◆			Sw Module Test Specification
				■						Sw Module Verification Report
CODE			■	◆	◆	◆	◆			Sw Source Code
				■		◆	◆			Sw Source Code Verification Report
MODULE TESTING			■	◆						Sw Module Test Report
SW INTEGRATION				■						Sw Integration Test Report
										Data Test Report
SW/HW INTEGRATION					■					Sw/Hw Integration Test Report
VALIDATION (*)						■				Sw Validation Report

Example (EN50128)

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

Software Maintenance Phase
- Software Maintenance Records
- Software Change Records

Software Assessment Phase
- Software Assessment Report

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

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

Software Validation Phase
- Software Validation Report

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

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

Software Integration Phase
- Software Integration Test Report

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

Software Module Testing Phase
- Software Module Test Report

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

- Document structure in EN50128
- 30 documents in a systematic structure
  - Specification
  - Design
  - Verification

Human factors

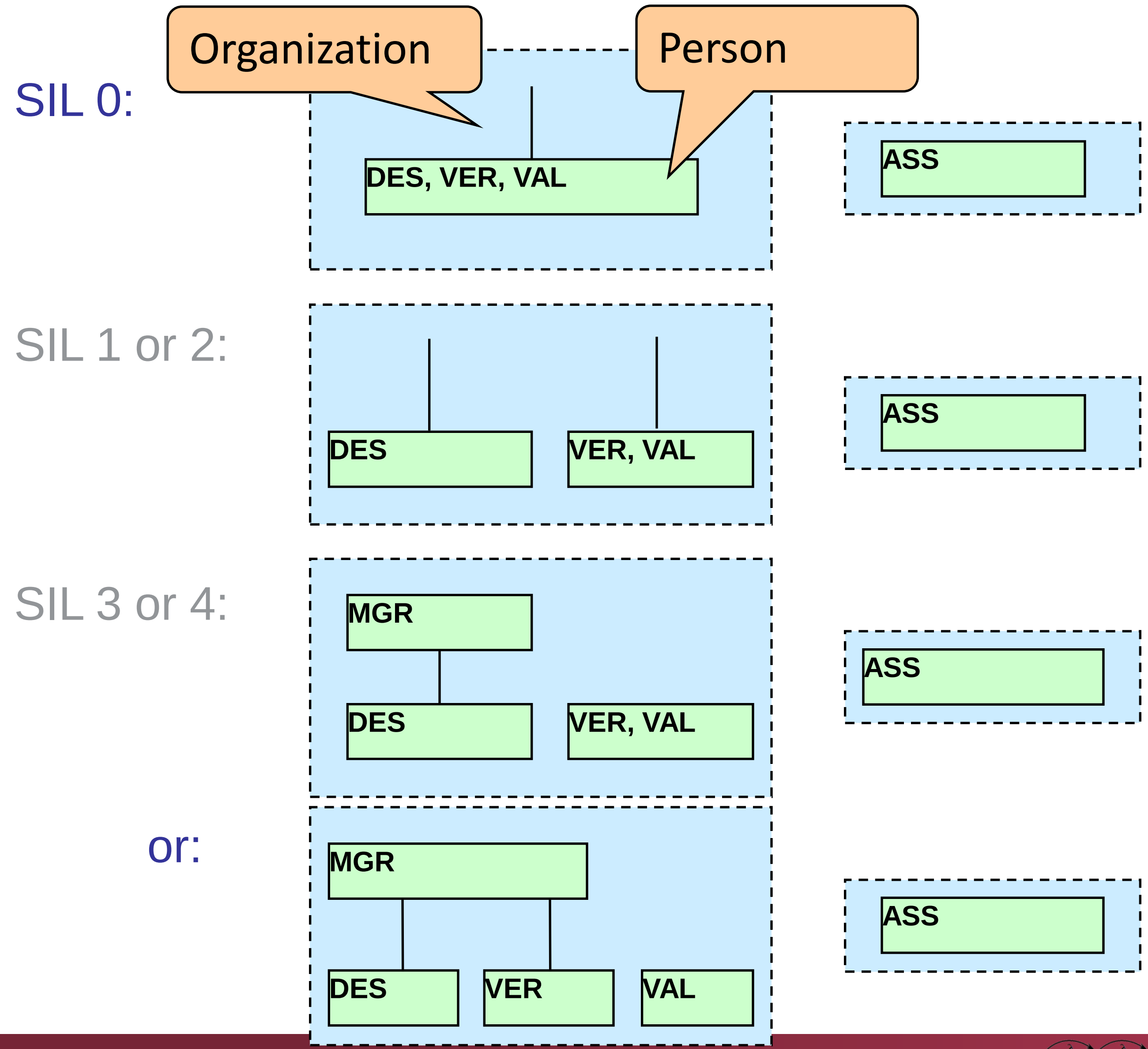
- In contrast to computers
  - Humans often fail in:
    - reacting in time
    - following a predefined set of instructions
  - Humans are good in:
    - handling unanticipated problems
- Human errors
  - Not all kind of human errors are equally likely
  - Hazard analysis (FMECA) is possible in a given context
  - Results shall be integrated into system safety analysis
- Reducing the errors of developers
  - Safe languages, tools, environments
  - Training, experience and redundancy (independence)
- Reducing operator errors:
  - Designing ergonomic HMI (patterns are available)
  - Designing to aid the operator rather than take over

Organization

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

Independence of personnel

EN 50128:



Summary

- Safety-critical systems
  - Hazard, risk
  - THR and Safety Integrity Level
- Dependability
  - Attributes of dependability
  - Fault -> Error -> Failure chain
  - Means to improve dependability
- Development process
  - Lifecycle activities
  - Methods and techniques
  - Documentation
  - Organization