Universität Bremen

dfki Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Systems of High Safety and Security

Lecture 2 from 21.10.26:
Legal Requirements - Norms and Standards

Winter term 2026/27



Serge Autexier, Christoph Lüth

Norms and Standards — why bother?

Norms vs. Standards

- **Norms** are collections of definitions and rules that have been published by one of the standardization bodies, like DIN, ISO, IEC, EN.
- **Standards** contain the same kinds of information as norms, but have not been published by a standardization body. Examples of standards that are not norms include
  - HTML (published by . . . ?)
  - UML Specifications, published by OMG https://www.omg.org/spec/UML/2.5.1/About-UML
  - C standard (published by ANSI/ISO)
- In practice both are used interchangeably.

Why bother with norms?

If you want (or need) to write safety-critical software
then you need to adhere to state-of-the-art practice
as specified by the relevant norms and standards.

- The **bad** news:
  - As a **qualified professional**, you may become personally liable if you deliberately and intentionally (**grob vorsätzlich**) disregard the state of the art or do not comply to the rules (norms, standards) that were to be applied.
- The **good** news:
  - Pay attention here and you will be delivered from these evils.
- Caution: applies to all kinds of software.

Because in case of failure...

- Whose fault is it?
- Who pays for it?
- Product liability — **Produkthaftung**
  - European practice: extensive regulation
  - American practice: litigation (lawsuits)
- Norms often put a lot of emphasis on process and traceability (auditable evidence): Who decided to do what, why, and how?
- Which are the relevant norms related to safety and security? $\longrightarrow$ Examples later.

What is Safety?

- Absolute definition:
  - "Safety is freedom from accidents or losses."
    - Nancy Leveson, "Safeware: System safety and computers"
- But is there such a thing as absolute safety?
- Technical definition:
  - German: "Sicherheit: Freiheit von **unvertretbaren** Risiken"
  - English: "Safety: Freedom from unjustifiable risks"
    - IEC 61508-4:2001, §3.1.8

Legal Grounds

- The **machinery directive** : *The Directive 2006/42/EC of the European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC (recast)*
- Scope:
  - Machineries (with a drive system and movable parts)
- Objective:
  - Market harmonization (not safety) – machinery rightfully carrying the CE label is allowed to be sold in all EU countries, regardless of the manufacturer's country
- Structure:
  - Sequence of whereas clauses (explanatory)
  - followed by 29 articles (main body)
  - and 12 subsequent annexes (detailed information about particular fields, e.g. health & safety)
- Some application areas have their own regulations:
  - Cars and motorcycles, railways, planes, nuclear plants ...

The Norms and Standards Landscape

- First-tier standards (**A-Normen**)
  - General, widely applicable, no specific area of application
  - Example: IEC 61508
- Second-tier standards (**B-Normen**)
  - Restriction to a particular area of application
  - Example: ISO 26262 (IEC 61508 for automotive domain)
- Third-tier standards (**C-Normen**)
  - Specific pieces of equipment
  - Example: IEC 61496-3 (“Berührungslos wirkende Schutzeinrichtungen”)
- Always use most specific norm.

Norms for the Working Programmer

- **IEC 61508:** *"Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)"*
  - Widely applicable, general, considered to be hard to understand
- **ISO 26262**
  - Specialisation of 61508 to automotive domain
- **ISO 21448**:"Road vehicles – Safety of the intended functionality"
  - Complements ISO 26262 for autonomous vehicles using AI algorithms
- **ANSI/UL 4600**: "Standard for Safety – Evaluation of Autonomous Products"
  - For assuring system-level safety of autonomous systems (cars, railways, aircrafts, robots,...)
- **DIN EN 50128:2012-03**
  - Specialisation of 61508 to software for railway industry
- **RTCA DO 178-C**: "*Software Considerations in Airborne Systems and Equipment Certification* "
  - Airplanes, NASA/ESA

- **ISO 15408:** "*Common Criteria for Information Technology Security Evaluation*"
  - Security, evolved from TCSEC ("Orange Book", US), ITSEC (EU), CTCPEC (Canada)
  - Must be applied for security-relevant systems or products used in military or public administration.
- **IEC 62443:** "*Industrial communication networks – IT security for networks and systems*"
  - Practical alternative to ISO 15408 for industrial applications

Functional Safety: IEC 61508 and friends

What is regulated by IEC 61508?

- Analyse the operation of the **unprotected system**.
  - What is the risk that could occur if the system is operated **without protection**?
- If this risk is too high, introduce a **safety controller** (also called **safety supervisor**) which may be realised as an electric, electronic or programmable electronic system (E/E/PES) and performs a safety function.
  - The system to be operated with the safety controller becomes the Equipment Under Control (EUC).
- Now the safety controller **could fail** when needed.
  - Therefore, the safety controller must be developed with **greater care** if its **failure** may result in **higher risks**.
  - Therefore, the safety controller needs to be developed according to a certain **safety integrity level** (SIL).

The Standard Approach to Safety of IEC 61508

- **Risk analysis** determines the safety integrity level (SIL).
- **Hazard analysis** leads to **safety requirement** specification.
- **Safety requirements** must be satisfied by product:
  - Need to **verify** that this is achieved.
  - SIL determines amount of development documentation and V&V effort
- **Life-cycle** needs to be managed and organised:
  - Planning: development plan, verification & validation plan ...
  - Note: personnel needs to be qualified.
- All of this needs to be **independently assessed** by verification and validation (V&V) teams, certification authorities whose personnel are called safety assessors.
  - SIL determines the required independence of assessment body.

Definition: Safety Integrity

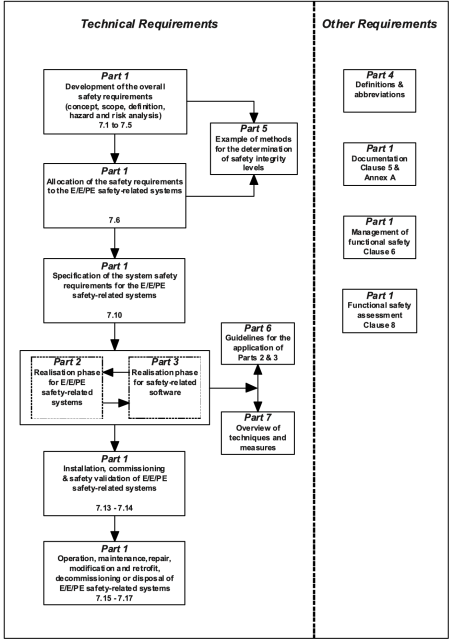
- Safety integrity is defined as the probability of a safety-related system to satisfactorily perform the required safety functions under all the stated conditions within a stated period of time.
- Safety integrity does not say anything about the availability or reliability of end-user functionality, if the latter is not safety-related.
- Definitions of reliability and availability:
  - Availability = probability of the system to be usable with its specified services when it is needed;
  - Reliability = probability of the system to perform its specified services while it is used.

The Seven Parts of IEC 61508

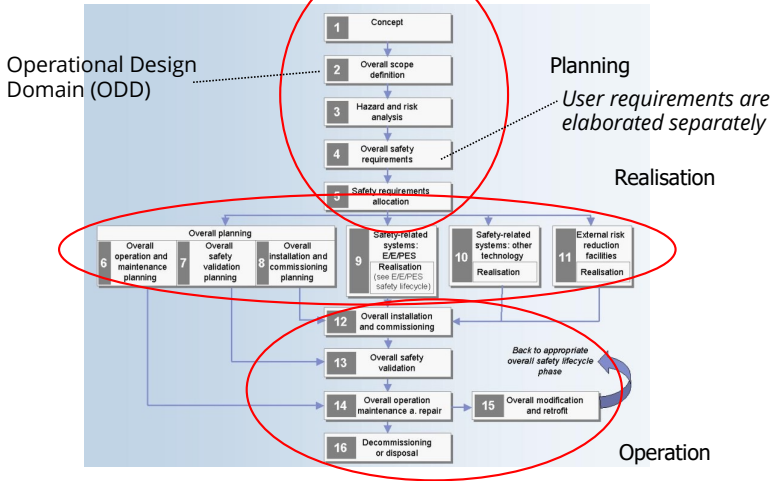
Electric / Electronic / Programmable Electronic Software

- General requirements
- Requirements for E/E/PES safety-related systems
  - Hardware rather than software
- **Software requirements**
- Definitions and abbreviations
- Examples of methods for the determination of safety-integrity levels
  - **Mostly informative**
- Guidelines on the application of Part 2 and 3
  - **Mostly informative**
- Overview of techniques and measures

The Seven Parts of IEC 61508



The Safety Life Cycle (IEC 61508)



E/E/PES: Electrical/Electronic/Programmable Electronic Safety-related Systems

Risk Definitions

- The most general definition of risk – also given in IEC 61508 – is as follows.
  - Risk is the combination of the probability of occurrence of harm and the severity of harm
- Two frequently applied formulas are
  - Risk = probability-of-occurrence-of-harm * cost-of-resulting-harm
  - Risk = frequency-of-occurrence-of-harm * cost-of-resulting-harm
- Probability and frequency are always expressed as **scalar value per time unit** , typically **value/hour** or **value/year**

The Safety Integrity Level

Safety Integrity Levels

Maximum tolerable risk exposure

- What is the risk of operating a system?
- Two factors:
  - How likely is a failure ?
  - What is the damage caused by a failure?



Numerical Characteristics of Safety Integrity Level

- The standard IEC 61508 defines the following numerical characteristics per safety integrity level:
  - **PFD**, average Probability of Failure to perform its designed function on Demand (average probability of dangerous failure on demand of the safety function), i.e. the probability of unavailability of the safety function leading to dangerous consequences.
  - **PFH**, the Probability of a dangerous Failure per Hour (average frequency of dangerous failure of the safety function).
- Failure on demand = "function fails when it is needed"

Safety Integrity Level (SIL)

- The SIL defines the **probability** that the safety control function **fails** to perform its function (on demand).
- The higher the SIL, the more strictly development, verification, and validation methods must be regulated and documented; and development personnel must be higher qualified.
- Maximum average probability of a **dangerous failure** (per hour/per demand) is determined depending on how often it is used:
  - **Low demand** of the safety function: at most once per year $\rightarrow$ max. PFD
    - Example: Airbag and its controller
  - **High demand**: more than once per year $\rightarrow$ max. PFH
    - Example 1: Car brakes (demand still occurs at discrete points in time)
    - Example 2: Safety function of autonomous vehicle (continuous demand)

Safety Integrity Level and PFD/PFH

- The following table correlates the SIL to the max. PFD/PFH:

SIL	High Demand (more than once a year)	Low Demand (once a year or less)
4	$10^{-9} < \text{PFH/h} < 10^{-8}$	$10^{-5} < \text{PFD}_{avg} < 10^{-4}$
3	$10^{-8} < \text{PFH/h} < 10^{-7}$	$10^{-4} < \text{PFD}_{avg} < 10^{-3}$
2	$10^{-7} < \text{PFH/h} < 10^{-6}$	$10^{-3} < \text{PFD}_{avg} < 10^{-2}$
1	$10^{-6} < \text{PFH/h} < 10^{-5}$	$10^{-2} < \text{PFD}_{avg} < 10^{-1}$

- How to read this table: e.g. if we develop a safety function with SIL 3, we can (reasonably) assume it fails once between 10000 and 1000 years.

Establishing target SIL (Quantitative)

- IEC 61508 does not describe a standard procedure to establish a SIL target, it allows for alternatives.
- Quantitative approach:
  - Start with target risk level (maximal tolerable risk)
  - Factor in fatality and frequency
  - Develop system with such a SIL that target risk level is not superceded.

Maximum tolerable risk of fatality	Individual risk (per annum)
Employee	10^{-4}
Public	10^{-5}
Broadly acceptable ("Negligible")	10^{-6}

Establishing target SIL (Quantitative)

- Establish maximal tolerable risk of fatality A
- Establish probability of hazardous events leading to fatality B
- Determine low or high demand, and establish risk of failure (i.e. a hazardous state is entered) of **unprotected** process C; determine
- Risk of fatality, unprotected is $B \cdot C$.
- If $B \cdot C > A$, then we need a **safety function**. This will have a probability of failure of E.
- The risk of fatality, protected, is $E \cdot B \cdot C$. E must be so low that $E \cdot B \cdot C \leq A$, so E can be computed as

$$E \leq \frac{A}{B \cdot C}$$

- Now lookup E in table on previous slide to determine SIL. E is PFH/h with high demand, or PFD_{avg} with low demand.

Example: Safety system for a chemical plant

- Max. tolerable risk exposure: $A = 10^{-6}$ (per annum)
- Probability of hazardous events leading to fatality: $B = 10^{-2}$
- Risk of failure of unprotected process: $C = \frac{1}{5}$ per annum (once in 5 years),
  - The probability that the safety-function needs to kick in is less than once per year: we can determine the SIL-level according to the low demand column.
- Risk of fatality, unprotected: $B \cdot C = 2 \cdot 10^{-3}$ (once in 500 years)
- Since $B \cdot C > A$, we need safety function with $E \leq \frac{A}{B \cdot C} = 5 \cdot 10^{-4}$
- Lookup E in table in low demand colum: **SIL 3.**

Example: Safety system for a hydraulic press

- Max. tolerable risk exposure: A=10^{-4} per annum, i.e. $A' = 10^{-8}$ per hour
- Ratio of hazardous events leading to serious injury: $B = \frac{1}{100}$.
  - Worker will not wilfully put his hands into the press.
- Risk of failure of unprotected process: $C = 50$ per hour.
  - Press operates and pushes down on work item 50 times/hour.
  - This means that protection mechanism is needed more often than once per year: high demand of safety function.
- Risk of fatality, unprotected: $B \cdot C = \frac{50}{100} = \frac{1}{2}$ per hour
- Clearly $B \cdot C > A'$, so we need safety function with $E \leq \frac{A'}{B \cdot C} = 2 \cdot 10^{-8}$.
- Lookup E in high demand column: **SIL 3.**

Example: Heating iron (or similar domestic appliance)

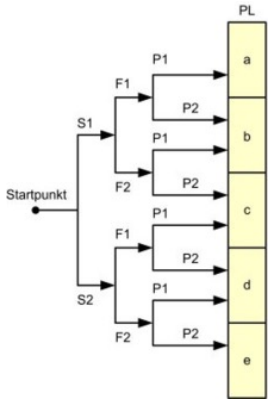
- Overheating may cause fire.
- Max. tolerable risk exposure: $A = 10^{-5}$ per annum, i.e. $A' = 10^{-9}$ per hour.
- Study suggests 1 in 400 incidents leads to fatality, i.e. $B \cdot C = \frac{1}{400}$.
- High demand for safety controller, because heating iron is used very often.
- Since $B \cdot C > A'$, we need safety function with $E \leq \frac{A'}{B \cdot C} = 10^{-9} \cdot 400 = 4 \cdot 10^{-7}$.
- Lookup E in table high demand column: **SIL 2.**

Establishing Target SIL (Qualitative)

- Qualitative method: risk graph analysis (e.g. DIN 13849).
- DIN EN ISO 13849:1 determines the **performance level (PL).**

PL	SIL
a	-
b	1
c	2
d	3
e	4

Relation PL to SIL



Source: Peter Wratil (Wikipedia)

Severity of injury:
S1 - slight (reversible) injury
S2 – severe (irreversible) injury

Occurrence:
F1 – rare occurrence
F2 – frequent occurrence

Possible avoidance:
P1 – possible
P2 – impossible

What are the implications of SIL for the development process?

- **In general**:
  - "Competent" personnel
  - Independent assessment ("four eyes")
- **SIL 1**:
  - Basic quality assurance (e.g. ISO 9001).
- **SIL 2**:
  - Safety-directed quality assurance, more tests.
- **SIL 3**:
  - Exhaustive testing, possibly formal methods.
  - Assessment by separate department.
- **SIL 4**:
  - State-of-the-art practices, formal methods.
  - Assessment by separate organization

Different Norms have differing notions of safety integrity

Approximate cross-domain mapping of ASIL

Domain	Domain-Specific Safety Levels					
Automotive (ISO 26262)	QM	ASIL A	ASIL B	ASIL C	ASIL D	-
General (IEC 61508)	-	SIL-1	SIL-2		SIL-3	SIL-4
Railway (CENELEC 50126/128/129)	-	SIL-1	SIL-2		SIL-3	SIL-4
Space (ECSS-Q-ST-80)	Category E	Category D	Category C		Category B	Category A
Aviation: airborne (ED-12/DO-178/DO-254)	DAL-E	DAL-D	DAL-C		DAL-B	DAL-A
Aviation: ground (ED-109/DO-278)	AL6	AL5	AL4	AL3	AL2	AL1
Medical (IEC 62304)	Class A	Class B			Class C	-
Household (IEC 60730)	Class A	Class B			Class C	-
Machinery (ISO 13849)	PL a	PL b	PL c	PL d	PL e	-

Source: https://en.wikipedia.org/wiki/Automotive_Safety_Integrity_Level

Some Terminology

- Error handling:
  - Fail-safe (or fail-stop): terminate in a safe state.
  - Fail-operational systems: continue operation, even if (some) controllers fail.
  - Fault-tolerant systems: continue with a potentially degraded service (more general than fail operational systems)
- Safety-critical, safety-relevant (German: *sicherheitskritisch*)
  - General term – failure may lead to risk
- Safety function (German: *Sicherheitsfunktion*)
  - Technical term, the part of the functionality that ensures safety
- Safety-related (German: *sicherheitsgerichtet*, *sicherheitsbezogen*)
  - Technical term, directly related to the safety function

Increasing SIL by redundancy

- One can achieve a higher SIL by combining **independent** systems with lower SIL (*Mehrkanalsysteme*).
- Given two systems A, B with failure probabilities P_A, P_B the chance for failure of both is (with P_{CC} probability of common-cause failures): $P_{AB} = P_{CC} + P_A P_B$
- Hence, combining two SIL 3 systems **may** give you a SIL 4 system.
- However, be aware of **systematic** errors (and note that IEC 61508 considers all software errors to be systematic) — these result in common-cause failures.
- Note also that for fail-operational systems you need three (not two) systems.
- The degree of independence can be increased by **software diversity**: channels are equipped with software following the same specification but developed by independent teams

The Software Development Process

- 61508 in principle allows any software lifecycle model, but:
  - No specific process model is given, illustrations use a V-model, and no other process model is mentioned.
- Appx A, B give normative guidance on measures to apply:
  - Error detection needs to be taken into account (e.g. runtime assertions, error detection codes, dynamic supervision of data/control flow)
  - Use of strongly typed programming languages (see table)
  - Discouraged use of certain features:
    - recursion(!), dynamic memory, unrestricted pointers, unconditional jumps
  - **Certified** (also called **qualified** or **validated**) tools and compilers must be used or tools "proven in use".

Proven in Use: Statistical Evaluation

- As an alternative to systematic development, statistics about usage may be employed. This is particularly relevant:
  - for development tools (compilers, verification tools etc),
  - and for re-used software (modules, libraries).
- The norm (61508-7 Appx. D) is quite brief about this subject. It states these methods should only be applied by those "competent in statistical analysis".
- The problem: proper statistical analysis is more than just "plugging in numbers".
  - Previous use needs to be to the same specification as intended use (e.g. compiler: same target platform).
  - Uniform distribution of test data, independent tests.
  - Perfect detection of failure.
- The rapid change of technology makes it frequently impossible to gather reliable statistic data about tools or HW components.
  - Therefore, "proven in use approach" is not often possible to be applied.

Table A.2 - Software Architecture

Table A.2 – Software design and development – software architecture design

(see 7.4.3)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
	Architecture and design feature					
1	Fault detection	C.3.1	---	R	HR	HR
2	Error detecting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	---	R	R	----
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	---	R	R	HR
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	---	---	R
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	---	---	R	HR
3f	Backward recovery	C.3.6	R	R	---	NR
3g	Stateless software design (or limited state design)	C.2.12	---	---	R	HR
4a	Re-try fault recovery mechanisms	C.3.7	R	R	---	---
4b	Graceful degradation	C.3.8	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.9	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.10	---	NR	NR	NR

Table A.2 - Software Architecture

7	Modular approach	Table B.9	HR	HR	HR	HR
8	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
11a	Structured diagrammatic methods **	C.2.1	HR	HR	HR	HR
11b	Semi-formal methods **	Table B.7	R	R	HR	HR
11c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
11d	Automatic software generation	C.4.6	R	R	R	R
12	Computer-aided specification and design tools	B.2.4	R	R	HR	HR
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	R	HR	HR	HR
13b	Time-triggered architecture	C.3.11	R	HR	HR	HR
13c	Event-driven, with guaranteed maximum response time	C.3.11	R	HR	HR	-
14	Static resource allocation	C.2.6.3	-	R	HR	HR
15	Static synchronisation of access to shared resources	C.2.6.3	-	-	R	HR

Table A.4 - Software Design & Development

Table A.4 – Software design and development – detailed design

(See 7.4.5 and 7.4.6)

(Includes software system design, software module design and coding)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Structured methods **	C.2.1	HR	HR	HR	HR
1b	Semi-formal methods **	Table B.7	R	HR	HR	HR
1c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
2	Computer-aided design tools	B.3.5	R	R	HR	HR
3	Defensive programming	C.2.5	---	R	HR	HR
4	Modular approach	Table B.9	HR	HR	HR	HR
5	Design and coding standards	C.2.6 Table B.1	R	HR	HR	HR
6	Structured programming	C.2.7	HR	HR	HR	HR
7	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
8	Forward traceability between the software safety requirements specification and software design	C.2.11	R	R	HR	HR

Table A.5 – SW Design and Development

Table A.5 – Software design and development – software module testing and integration

(See 7.4.7 and 7.4.8)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	R
2	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
3	Data recording and analysis	C.5.2	HR	HR	HR	HR
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
5	Performance testing	Table B.6	R	R	HR	HR
6	Model based testing	C.5.27	R	R	HR	HR
7	Interface testing	C.5.3	R	R	HR	HR
8	Test management and automation tools	C.4.7	R	HR	HR	HR
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	R	R	HR	HR
10	Formal verification	C.5.12	---	---	R	R

Table A.6 – Programmable Electronics Integration (HW/SW)

Table A.6 – Programmable electronics integration (hardware and software)

(See 7.5)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
2	Performance testing	Table B.6	R	R	HR	HR
3	Forward traceability between the system and software design requirements for hardware/software integration and the hardware/software integration test specifications	C.2.11	R	R	HR	HR

Table A.7 - Software Aspects of System Safety Validation

Table A.7 – Software aspects of system safety validation

(See 7.7)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	HR
2	Process simulation	C.5.18	R	R	HR	HR
3	Modelling	Table B.5	R	R	HR	HR
4	Functional and black-box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	HR
5	Forward traceability between the software safety requirements specification and the software safety validation plan	C.2.11	R	R	HR	HR
6	Backward traceability between the software safety validation plan and the software safety requirements specification	C.2.11	R	R	HR	HR

NOTE 1 See Table C.7

Table A.9 – Software Verification

Table A.9 – Software verification

(See 7.9)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Formal proof	C.5.12	---	R	R	HR
2	Animation of specification and design	C.5.26	R	R	R	R
3	Static analysis	B.6.4 Table B.8	R	HR	HR	HR
4	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	HR
5	Forward traceability between the software design specification and the software verification (including data verification) plan	C.2.11	R	R	HR	HR
6	Backward traceability between the software verification (including data verification) plan and the software design specification	C.2.11	R	R	HR	HR
7	Offline numerical analysis	C.2.13	R	R	HR	HR
Software module testing and integration		See Table A.5				
Programmable electronics integration testing		See Table A.6				
Software system testing (validation)		See Table A.7				

Table B.1 – Coding Guidelines

- Table C.1, programming languages, mentions:
  - ADA, Modula-2, Pascal, FORTRAN 77, C, PL/M, Assembler, ...
- Example for a guideline:
  - MISRA-C: 2023, Guidelines for the use of the C language in critical systems.

Tabelle B.1 – Entwurfs- und Codierungs-Richtlinien (Verweisungen aus Tabelle A.4)

	Verfahren/Maßnahme *	siehe	SIL1	SIL2	SIL3	SIL4
1	Verwendung von Codierungs-Richtlinien	C.2.6.2	++	++	++	++
2	Keine dynamischen Objekte	C.2.6.3	+	++	++	++
3a	Keine dynamischen Variablen	C.2.6.3	o	+	++	++
3b	Online-Test der Erzeugung von dynamischen Variablen	C.2.6.4	o	+	++	++
4	Eingeschränkte Verwendung von Interrupts	C.2.6.5	+	+	++	++
5	Eingeschränkte Verwendung von Pointern	C.2.6.6	o	+	++	++
6	Eingeschränkte Verwendung von Rekursionen	C.2.6.7	o	+	++	++
7	Keine unbedingten Sprünge in Programmen in höherer Programmiersprache	C.2.6.2	+	++	++	++

ANMERKUNG 1 Die Maßnahmen 2 und 3a brauchen nicht angewendet zu werden, wenn ein Compiler verwendet wird, der sicherstellt, dass genügend Speicherplatz für alle dynamischen Variablen und Objekte vor der Laufzeit zugeteilt wird, oder der Laufzeittests zur korrekten Online-Zuweisung von Speicherplatz einfügt.

* Es müssen dem Sicherheits-Integritätslevel angemessene Verfahren/Maßnahmen ausgewählt werden. Alternative oder gleichwertige Verfahren/Maßnahmen sind durch einen Buchstaben hinter der Nummer gekennzeichnet. Es muss nur eine(s) der alternativen oder gleichwertigen Verfahren/Maßnahmen erfüllt werden.

Table B.5 - Modelling

Tabelle B.5 – Modellierung
(Verweisung aus der Tabelle A.7)

Verfahren/Maßnahme *	siehe	SIL1	SIL2	SIL3	SIL4
1 Datenflussdiagramme	C.2.2	+	+	+	+
2 Zustandsübergangsdiagramme	B.2.3.2	o	+	++	++
3 Formale Methoden	C.2.4	o	+	+	++
4 Modellierung der Leistungsfähigkeit	C.5.20	+	++	++	++
5 Petri-Netze	B.2.3.3	o	+	++	++
6 Prototypenerstellung/Animation	C.5.17	+	+	+	+
7 Strukturdiagramme	C.2.3	+	+	+	++

ANMERKUNG Sollte eine spezielles Verfahren in dieser Tabelle nicht vorkommen, darf nicht angenommen werden, dass dieses nicht in Betracht gezogen werden darf. Es sollte zu dieser Norm in Einklang stehen.

* Es müssen dem Sicherheits-Integritätslevel angemessene Verfahren/Maßnahmen ausgewählt werden.

Certification

- **Certification** is the process of showing **conformance** to a **standard**.
- Conformance to IEC 61508 can be shown in two ways:
  - either that an organization (company) has in principle the ability to produce a product conforming to the standard,
  - or that a specific product (or system design) conforms to the standard.
- Certification can be done by the developing company (self-certification) but is typically done by a **notified body** ("benannte Stellen").
  - In Germany, e.g. the TÜVs or **Berufsgenossenschaften** ;
  - In Britain, professional role (ISA) supported by IET/BCS;
  - Aircraft certification in Europe: EASA (European Aviation Safety Agency)
  - Aircraft certification in US: FAA (Federal Aviation Administration)

An Important New Trend

- Until recently, software components contributing to a safety function had to be developed in a way ensuring that the probability of remaining software bugs could be neglected (formal verification, exhaustive testing). It was not allowed to leave a known bug inside the SW and argue that the risk induced by this bug is small enough to be neglected.
- With the advent of autonomous systems (in particular, road vehicles and robots), AI-related methods were implemented in software, such as neural networks for traffic sign recognition. This type of software components has two new characteristics:
  1. It performs correctly only with a certain probability.
  2. It may change its behaviour over time, due to effects of runtime machine learning.
- The standards in the automotive domain are already being adapted: ISO 21448; other standards will follow. This trend comes with a considerable risk: In the future, the probabilistic argument might also be applied to buggy software which should better be corrected.

Conclusion

Summary

- Norms and standards enforce the application of the state-of-the-art when developing software which is **safety-critical** or **security-critical**.
- Wanton disregard of these norms may lead to **personal liability**.
- Norms typically place a lot of emphasis on **process**.
- Key question are **traceability** of decisions and design, and **verification** and **validation**.
- Different application fields have different norms:
  - **Safety**: IEC 61508 and its specializations, e.g. ISO 26262, DO-178 B/C.
  - **Security**: IEC 15408 (“Common Criteria”)

Further Reading

- Terminology for dependable systems:
  - J. C. Laprie *et al.*: Dependability: Basic Concepts and Terminology. Springer-Verlag, Berlin Heidelberg New York (1992).
- Literature on safety-critical systems:
  - Storey, Neil: Safety-Critical Computer Systems. Addison Wesley Longman (1996).
  - Nancy Levenson: Safeware – System Safety and Computers. Addison-Wesley (1995).
- A readable introduction to IEC 61508:
  - David Smith and Kenneth Simpson: Functional Safety. 2nd Edition, Elsevier (2004).