Universität Bremen

Systeme hoher Sicherheit und Qualität

WS 2019/2020

Lecture 02:
Legal Requirements - Norms and Standards

Christoph Lüth, Dieter Hutter, Jan Peleska

Where are we?

Why Bother with Norms?

Why bother with norms?

> **If** you want (or need to) to write safety-criticial software
> **then** you need to adhere to state-of-the-art practice
> **as** encoded by the relevant norms & 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? ("Produkthaftung")
  - European practice: extensive regulation
  - American practice: judicial mitigation (lawsuits)
- Standards often put a lot of emphasis on process and traceability (**auditable evidence**). Who decided to do what, why, and how?
- What are norms relevant to safety and security?
  Examples:
    - Safety: IEC 61508 – Functional safety
      - specialised norms for special domains
    - Security: IEC 15408 – Common criteria
      - In this context: "cybersecurity", not "guns and gates"
- What is regulated by such norms?

Emergent Properties

- An **emergent property** of a system is one that cannot be attributed to a single system component, but results from the overall effect of system components inter-operating with each other and the environment
- **Safety and Security are emergent properties.**
  - They can only be analyzed in the context of the complete system and its environment
  - Safety and security can never be derived from the properties of a single component, in particular, never from that of a software component alone

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:
  - „Sicherheit: Freiheit von *unvertretbaren* Risiken"
    - 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)
- 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)
- Third-tier standards (C-Normen)
  - Specific pieces of equipment
  - Example: IEC 61496-3 ("Berührungslos wirkende Schutzeinrichtungen")
- Always use most specific norm.

The standards quagmire ?

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 hard to understand
- ISO 26262
  - Specialisation of 61508 to cars (automotive industry)
- DIN EN 50128:2011
  - Specialisation of 61508 to software for railway industry
- RTCA DO 178-B and C (new developments require 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 (US), ITSEC (EU), CTCPEC (Canada)

Functional Safety: IEC 61508 and friends

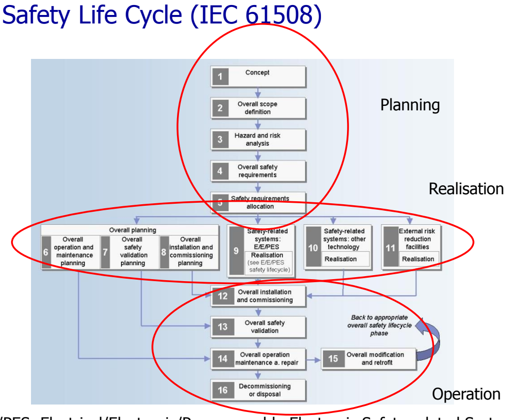
What is regulated by IEC 61508?

1. Risk analysis determines the safety integrity level (SIL).
2. Hazard analysis leads to safety requirement specification.
3. Safety requirements must be satisfied by product:
   - Need to verify that this is achieved.
   - SIL determines amount of testing/proving etc.
4. Life-cycle needs to be managed and organised:
   - Planning: verification & validation plan.
   - Note: personnel needs to be qualified.
5. All of this needs to be independently assessed.
   - SIL determines independence of assessment body.

The Seven Parts of IEC 61508

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

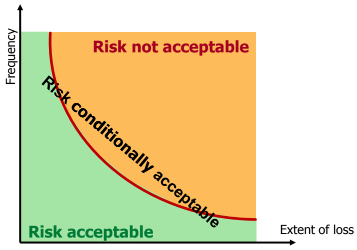
The Safety Life Cycle (IEC 61508)



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

Safety Integrity Levels

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



Safety Integrity Levels

- Maximum average probabilty of a **dangerous failure** (per hour/per demand) depending on how often it is used:

SIL	High Demand (more than once a year)	Low Demand (once a year or less)
4	$10^{-9} < P/hr < 10^{-8}$	$10^{-5} < P < 10^{-4}$
3	$10^{-8} < P/hr < 10^{-7}$	$10^{-4} < P < 10^{-3}$
2	$10^{-7} < P/hr < 10^{-6}$	$10^{-3} < P < 10^{-2}$
1	$10^{-6} < P/hr < 10^{-5}$	$10^{-2} < P < 10^{-1}$

- Examples:
  - High demand: car brakes
  - Low demand: airbag control
- Note: SIL only meaningful for **specific safety functions**.

Establishing target SIL (Quantitative)

- IEC 61508 does not describe standard procedure to establish a SIL target, it allows for alternatives.
- Quantitative approach
  - Start with target risk level
  - Factor in fatality and frequency

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

- Example: Safety system for a chemical plant
  - Max. tolerable risk exposure: $A=10^{-6}$ (per annum)
  - Ratio of hazardous events leading to fatality: $B= 10^{-2}$
  - Risk of failure of unprotected process: C= 1/5 per annum (ie. 1 in 5 years)
  - Risk of hazardous event, unprotected: $B*C= 2*10^{-3}$ (ie. 1 in 5000 years)
  - Risk of hazardous event, protected A = E*B*C (with E *failure on demand*)
  - Calculate E as $E = A/(B*C) = 5*10^{-4}$, so SIL 3
- More examples: airbag, safety system for a hydraulic press

Establishing target SIL (Quantitative)

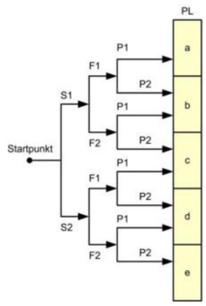
- Example: Safety system for a hydraulic press
  - Max. tolerable risk exposure: $A=10^{-4}$ per annum, i.e. $A'= 10^{-8}$ per hour
  - Ratio of hazardous events leading to serious injury: B= 1/100
    - Worker will not willfully put his hands into the press
  - Risk of failure of unprotected process: C= 50 per hour
    - Press operates
  - Risk of hazardous event, unprotected: B*C= 1/2 per hour
  - $E = A'/(B*C) = 2*10^{-8}$, so SIL 3
- Example: Domestic appliance, e.g. heating iron
  - 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*C= 1/400
  - Then $E= A'/B*C = 10^{-9}*400 = 4*10^{-7}$, so SIL 3

Establishing Target SIL (Qualitative)

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

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

Relation PL to SIL



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

Occurrence:
F1 – rare occurrence
F2 – frequent occurrence

Possible avoidance:
P1 – possible
P2 – impossible

Source: Peter Wratil (Wikipedia)

Numerical Characteristics

- The standard IEC 61508 defines the following numerical characteristics per safety integrity level:
  - PFD, average probability of failure to perform its design 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"

What does the SIL mean 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

Some Terminology

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

Increasing SIL by redudancy

- 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} probablity 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).
- 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 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 (eg. compiler: same target platform).
  - Uniform distribution of test data, indendent tests.
  - Perfect detection of failure.

Proven in Use: Statistical Evaluation

- Statistical statements can only be given with respect to a confidence level ($\lambda = 1 - p$), usually $\lambda = 0.99$ or $\lambda = 0.9$.
- With this and all other assumptions satisfied, we get the following numbers from the norm:
  - For on-demand: observed demands without failure (P_1: accepted probability of failure to perform per demand)
  - For continuously-operated: observed hours w/o failure (P_2: accepted probability of failure to perform per hour of operation)

SIL	On-Demand			Continuously Operated		
	P_1	$\lambda = 99\%$	$\lambda = 90\%$	P_2	$\lambda = 99\%$	$\lambda = 90\%$
1	$< 10^{-1}$	46	3	$< 10^{-5}$	$4.6 \cdot 10^5$	$3 \cdot 10^5$
2	$< 10^{-2}$	460	30	$< 10^{-6}$	$4.6 \cdot 10^6$	$3 \cdot 10^6$
3	$< 10^{-3}$	4600	3000	$< 10^{-7}$	$4.6 \cdot 10^7$	$3 \cdot 10^7$
4	$< 10^{-4}$	46000	30000	$< 10^{-8}$	$4.6 \cdot 10^8$	$3 \cdot 10^8$

Source: Ladkin, Littlewood: Practical Statistical Evaluation of Critical Software.

Table A.2 - Software Architecture

Tabelle A.2 – Softwareentwurf und Softwareentwicklung: Entwurf der Software-Architektur (siehe 7.4.3)

Verfahren/Maßnahme *	siehe	SIL1	SIL2	SIL3	SIL4
1 Fehlererkennung und Diagnose	C.3.1	o	+	++	++
2 Fehlererkennende und -korrigierende Codes	C.3.2	+	+	+	++
3a Plausibilitätskontrollen (Failure assertion programming)	C.3.3	+	+	+	++
3b Externe Überwachungseinrichtungen	C.3.4	o	+	+	+
3c Diversitäre Programmierung	C.3.5	+	+	+	++
3d Regenerationsblöcke	C.3.6	+	+	+	+
3e Rückwärtsregeneration	C.3.7	+	+	+	+
3f Vorwärtsregeneration	C.3.8	+	+	+	+
3g Regeneration durch Wiederholung	C.3.9	+	+	+	++
3h Aufzeichnung ausgeführter Abschnitte	C.3.10	o	+	+	++
4 Abgestufte Funktionseinschränkungen	C.3.11	+	+	++	++
5 Künstliche Intelligenz – Fehlerkorrektur	C.3.12	o	--	--	--
6 Dynamische Rekonfiguration	C.3.13	o	--	--	--
7a Strukturierte Methoden mit z. B. JSD, MASCOT, SADT und Yourdon.	C.2.1	++	++	++	++
7b Semi-formale Methoden	Tabelle B.7	+	+	++	++
7c Formale Methoden z. B. CCS, CSP, HOL, LOTOS, OBJ, temporäre Logik, VDM und Z	C.2.4	o	+	+	++

Table A.4 - Software Design & Development

Tabelle A.4 – Softwareentwurf und Softwareentwicklung: detaillierter Entwurf (siehe 7.4.5 and 7.4.6)
(Dies beinhaltet Software-Systementwurf, Entwurf der Softwaremodule und Codierung)

Verfahren/Maßnahme *	siehe	SIL1	SIL2	SIL3	SIL4
1a Strukturierte Methoden wie z. B. JSD, MASCOT, SADT und Yourdon	C.2.1	++	++	++	++
1b Semi-formale Methoden	Tabelle B.7	+	++	++	++
1c Formale Methoden wie z. B. CCS, CSP, HOL, LOTOS, OBJ, temporäre Logik, VDM und Z	C.2.4	o	+	+	++
2 Rechnergestützte Entwurfswerkzeuge	B.3.5	+	+	++	++
3 Defensive Programmierung	C.2.5	o	+	++	++
4 Modularisierung	Tabelle B.9	++	++	++	++
5 Entwurfs- und Codierungs-Richtlinien	Tabelle B.1	+	++	++	++
6 Strukturierte Programmierung	C.2.7	++	++	++	++

Table A.9 – Software Verification

Tabelle A.9 – Software-Verifikation (siehe 7.9)

Verfahren/Maßnahme *	siehe	SIL1	SIL2	SIL3	SIL4
1 Formaler Beweis	C.5.13	o	+	+	++
2 Statistische Tests	C.5.1	o	+	+	++
3 Statische Analyse	B.6.4 Tabelle B.8	+	++	++	++
4 Dynamische Analyse und Test	B.6.5 Tabelle B.2	+	++	++	++
5 Software-Komplexitätsmetriken	C.5.14	+	+	+	+

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: 2004, 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**.
  - Also sometimes (e.g. DO-178B) called `qualification'.
- 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 an **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)

Security: IEC 15408 - The Common Criteria

Recall: Security Criteria

- Confidentiality
- Integrity
- Availability
- Authenticity
- Accountability
- Non-repudiation

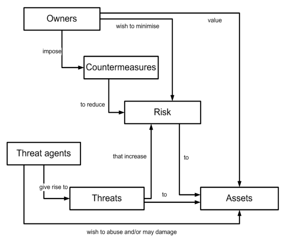
Common Criteria (IEC 15408)

- Established in 1996 as a harmonization of various norms to evaluate security properties of IT products and systems (e.g. ITSEC (Europe), TCSEC (US, “orange book”), CTCPEC (Canada))
- Basis for evaluation of **security properties** of IT products (or parts of) and systems (the **Target of Evaluation** TOE).
- The CC is useful as a guide for the development of products or systems with IT security functions and for the procurement of commercial products and systems with such functions.

Common Criteria

General Model

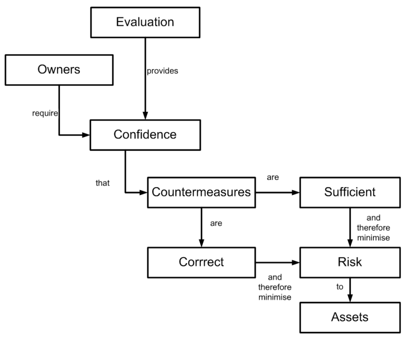
- Security is concerned with the protection of assets. Assets are entities that someone places value upon.
- Threats give rise to risks to the assets, based on the likelihood of a threat being realized and its impact on the assets
- (IT and non-IT) Counter-measures are imposed to reduce the risks to assets.



Security Goals

- Protection of information from unauthorized disclosure, modification, or loss of use:
  - **confidentiality**, **integrity**, and **availability**
  - may also be applicable to aspects
- Focus on **threats** to that information arising from human activities, whether malicious or otherwise, but may be applicable to some non-human threats as well.
- In addition, the CC may be applied in other areas of IT, but makes no claim of competence outside the strict domain of IT security.

Concept of Evaluation



Security Environment

- Laws, organizational security policies, customs, expertise and knowledge relevant for TOE
  - Context in which the TOE is intended to be used.
  - Threats to security that are, or are held to be, present in the environment.
- A statement of applicable organizational security policies would identify relevant policies and rules.
- Assumptions about the environment of the TOE are considered as axiomatic for the TOE evaluation.

Security Objectives

- Identification of all of the security concerns
  - Aspects addressed directly by the TOE or by its environment.
  - Incorporating engineering judgment, security policy, economic factors and risk acceptance decisions.
- Analysis of the security environment results in security objectives that counter the identified threats and address identified organizational security policies and assumptions.
- The security objectives for the environment would be implemented within the IT domain, and by non-technical or procedural means.
- Only the security objectives for the TOE and its IT environment are addressed by IT security requirements

Threats and Their Risks

- **Threats** to security of the assets relevant to the TOE.
  - in terms of a threat agent,
  - a presumed attack method,
  - any vulnerabilities that are the foundation for the attack, and
  - identification of the asset under attack.
- **Risks** to security. Assess each threat
  - by its likelihood developing into an actual attack,
  - its likelihood proving successful, and
  - the consequences of any damage that may result.

Security Requirements

- Refinement of security objectives into
  - Requirements for TOE and
  - Requirements for the environment
- **Functional requirements**
  - Functions in support for security of IT-system
  - E.g. identification & authentication, cryptography,…
- **Assurance Requirements**
  - Establishing confidence in security functions
  - Correctness of implementation
  - E.g. development, life cycle support, testing, …

Security Functions

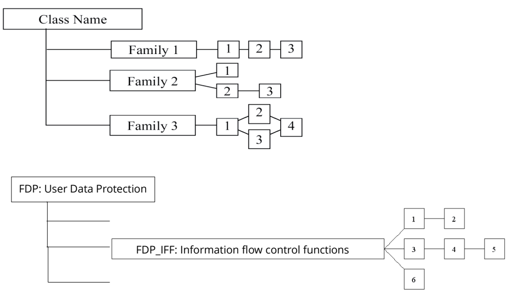
- The **statement of TOE security functions** shall cover the IT security functions and shall specify how these functions satisfy the TOE security functional requirements. This statement shall include a bi-directional mapping between functions and requirements that clearly shows which functions satisfy which requirements and that all requirements are met.
- Starting point for **design process**.

Security Functional Components

- Class FAU: Security audit
- Class FCO: Communication
- Class FCS: Cryptographic support
- **Class FDP: User data protection**
- Class FIA: Identification and authentication
- Class FMT: Security management
- Class FPR: Privacy
- Class FPT: Protection of the TSF
- Class FRU: Resource utilisation
- Class FTA: TOE access
- Class FTP: Trusted path/channels

Security Functional Components

- Content and presentation of the functional requirements



FDP – Information Flow Control

FDP_IFC.1 Subset information flow control

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1 The TSF shall enforce the [assignment: *information flow control SFP*] on [assignment: *list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

FDP_IFC.2 Complete information flow control

Hierarchical to: FDP_IFC.1 Subset information flow control

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.2.1 The TSF shall enforce the [assignment: *information flow control SFP*] on [assignment: *list of subjects and information*] and all operations that cause that information to flow to and from subjects covered by the SFP**.**

FDP_IFC.2.2 The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Assurance Requirements

Assurance Approach

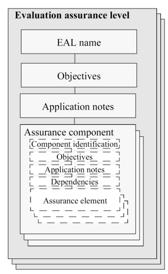
"The CC philosophy is to provide assurance based upon an evaluation (active investigation) of the IT product that is to be trusted. Evaluation has been the traditional means of providing assurance and is the basis for prior evaluation criteria documents. "

CC, Part 3, p.15

Assurance Requirements

- Concerning actions of the developer, evidence produced and actions of the evaluator.
- Examples:
  - Rigor of the **development process**
  - Search for and analysis of the impact of potential security vulnerabilities.
- **Degree of assurance**
  - varies for a given set of functional requirements
  - typically expressed in terms of increasing levels of rigor built with assurance components.
- **Evaluation assurance levels** (EALs) constructed using these components.

Part 3 Assurance levels



Assurance Components

- Class APE: Protection Profile evaluation
- Class ASE: Security Target evaluation
- Class ADV: Development
- Class AGD: Guidance documents
- Class ALC: Life-cycle support
- Class ATE: Tests
- Class AVA: Vulnerability assessment
- Class ACO: Composition

Evaluation Assurance Level

- EALs define levels of assurance (no guarantees)

1. Functionally tested
2. Structurally tested
3. Methodically tested and checked
4. Methodically designed, tested, and reviewed
5. Semi-formally designed and tested
6. Semi-formally verified design and tested
7. Formally verified design and tested

EAL5 – EAL7 require formal methods

Assurance class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Development	ADV_ARC		**1**	1	1	1	1	1
	ADV_FSP	**1**	**2**	**3**	**4**	**5**	5	**6**
	ADV_IMP				**1**	1	**2**	2
	ADV_INT					**2**	**3**	3
	ADV_SPM						**1**	1
	ADV_TDS		**1**	**2**	**3**	**4**	**5**	**6**
Guidance documents	AGD_OPE	**1**	1	1	1	1	1	1
	AGD_PRE	**1**	1	1	1	1	1	1
Life-cycle support	ALC_CMC	**1**	**2**	**3**	**4**	4	**5**	5
	ALC_CMS	**1**	**2**	**3**	**4**	**5**	5	5
	ALC_DEL		**1**	1	1	1	1	1
	ALC_DVS			**1**	1	1	**2**	2
	ALC_FLR							
	ALC_LCD			**1**	1	1	1	**2**
	ALC_TAT				**1**	**2**	**3**	3
Security Target evaluation	ASE_CCL	**1**	1	1	1	1	1	1
	ASE_ECD	**1**	1	1	1	1	1	1
	ASE_INT	**1**	1	1	1	1	1	1
	ASE_OBJ	**1**	**2**	2	2	2	2	2
	ASE_REQ	**1**	**2**	2	2	2	2	2
	ASE_SPD		**1**	1	1	1	1	1
	ASE_TSS	**1**	1	1	1	1	1	1
Tests	ATE_COV		**1**	**2**	2	2	**3**	3
	ATE_DPT			**1**	1	**3**	3	**4**
	ATE_FUN		**1**	1	1	1	**2**	2
	ATE_IND	**1**	**2**	2	2	2	2	**3**
Vulnerability assessment	AVA_VAN	**1**	**2**	2	**3**	**4**	**5**	5

Assurance Components Example: Development

ADV_FSP.1 Basic functional specification

EAL-1: … The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.

EAL-2: … The functional specification shall completely represent the TSF.

EAL-3: + … The functional specification shall summarize the SFR-supporting and SFR-non-interfering actions associated with each TSFI.

EAL-4: + … The functional specification shall describe all direct error messages that may result from an invocation of each TSFI.

EAL-5: … The functional specification shall describe the TSFI using a semi-formal style.

EAL-6: … The developer shall provide a formal presentation of the functional specification of the TSF. The formal presentation of the functional specification of the TSF shall describe the TSFI using a formal style, supported by informal, explanatory text where appropriate.

Degree of Assurance

(TSFI : Interface of the TOE Security Functionality (TSF), SFR : Security Functional Requirement)

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:
  - IEC 61508 and its specializations, e.g. DO-178B.
  - 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).