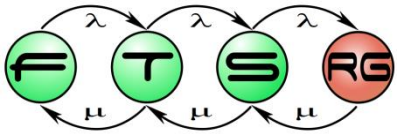
Overview of V&V techniques

Istvan Majzik, Zoltan Micskei

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group



Administration

- LAB0 sessions:
  - G1: 2019-09-19 14:15-16:00
  - GA: 2019-09-26 14:15-16:00
  - G2: 2019-09-26 16:15-18:00
- LAB0 exercises: https://github.com/FTSRG/swsv-labs/wiki/0a-Home-assignment-infrastructure

- Home assignment teams:
  - Form published (Github classroom)

Main topics of the course

- Overview (1.5)
  - Introduction, **V&V techniques**
- Static techniques (1.5)
  - Specification, Verifying source code
- Dynamic techniques: Testing (7)
  - Testing overview, Test design techniques
  - Test generation, Automation
- System-level verification (3)
  - Verifying architecture, Dependability analysis
  - Runtime verification

Learning outcomes

- **List typical V&V activities (K1)**
- **Classify the different verification techniques according to their place in the lifecycle (K2)**

RECAP: V&V TECHNIQUES

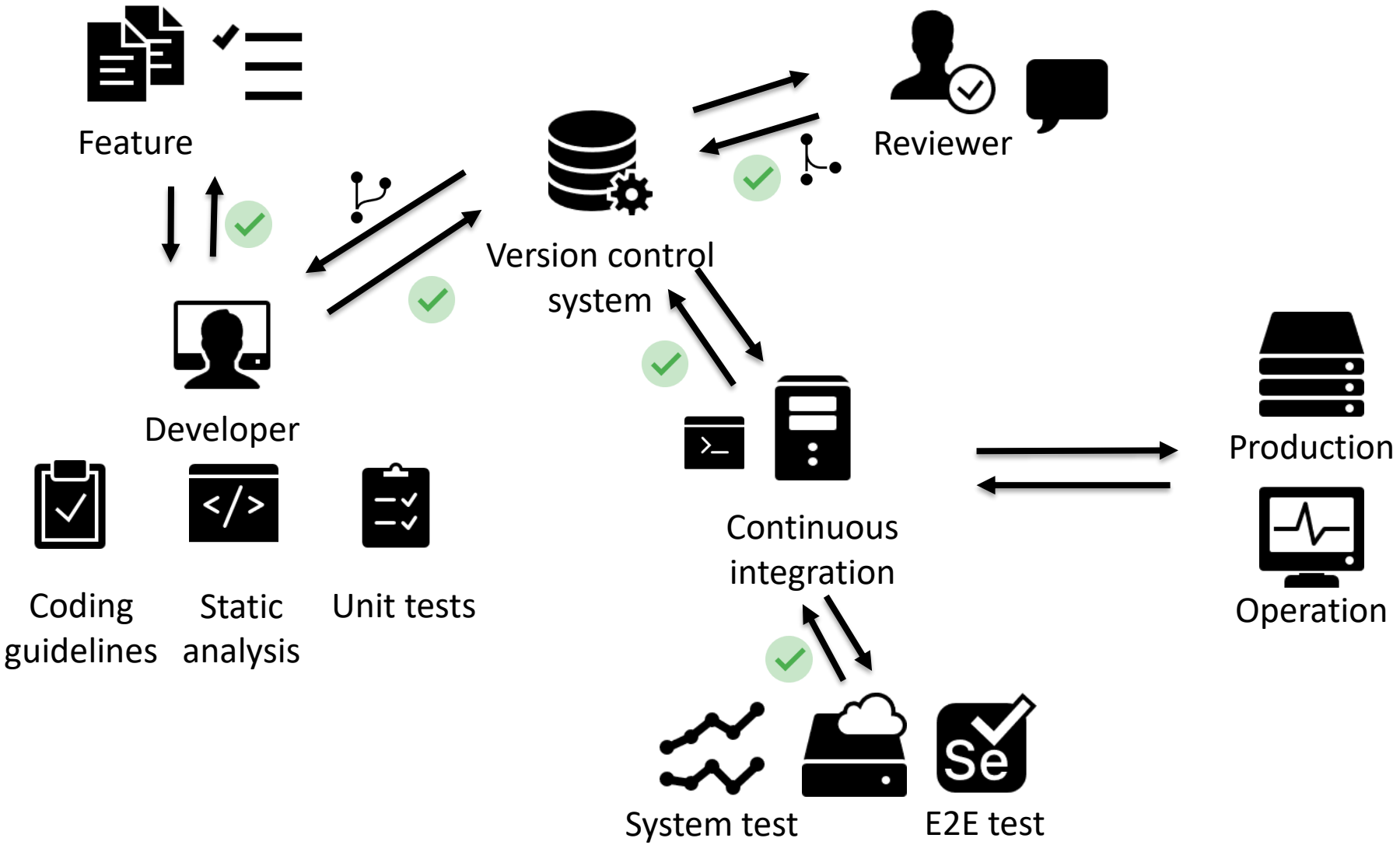
EXERCISE: Collect V&V techniques

What V&V techniques do you know?

(Tell: Why? Who? When?)

How can we categorize these techniques?

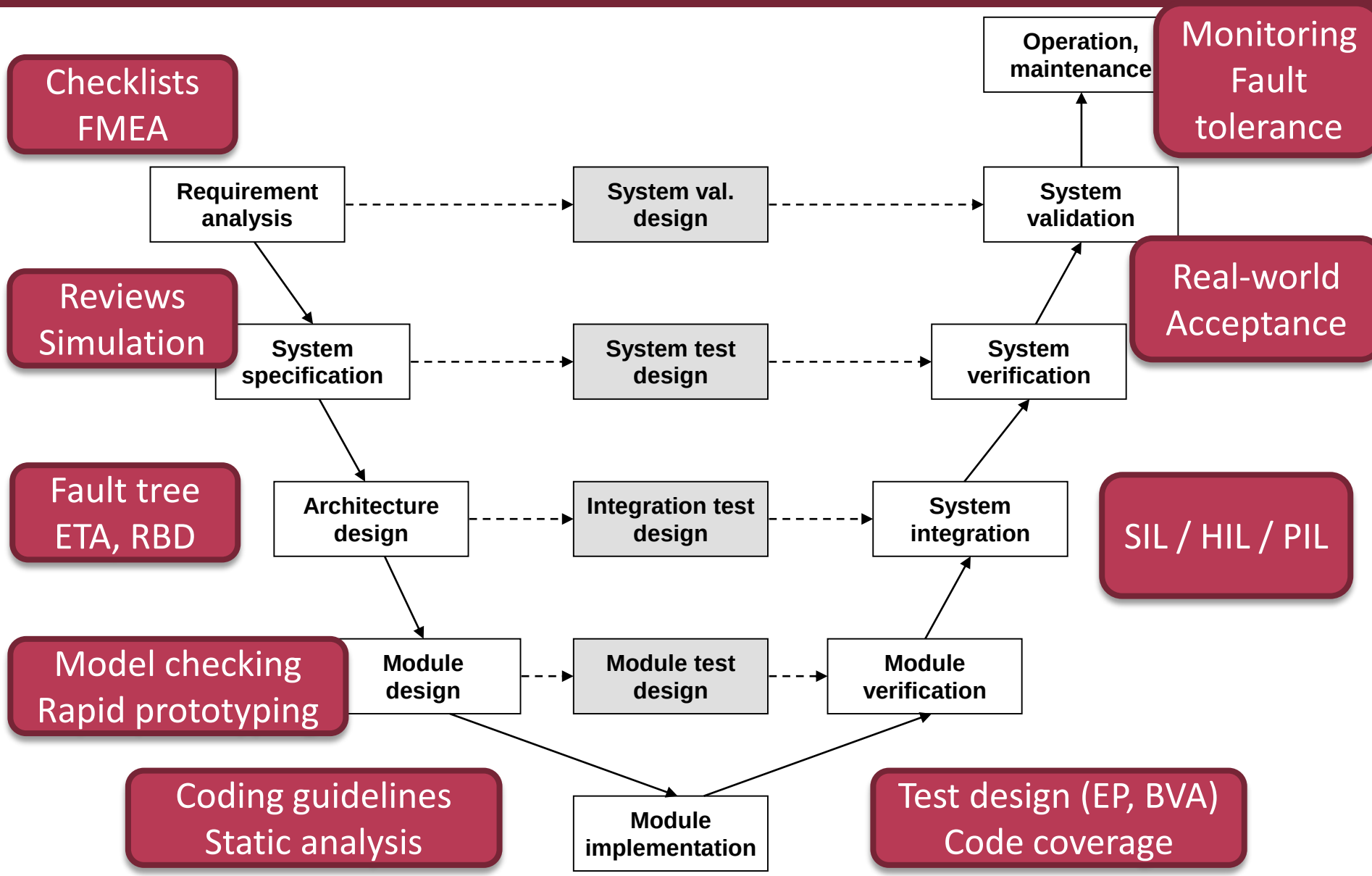
Continuous Verification and Feedback



Icons: icons8.com

See: https://www.mit.bme.hu/eng/eng/node/9675/lectures-0

V&V in the V-model (examples)



RECAP: MOTIVATION

Different kinds of faults

Development phase

- Specification faults
- Design faults
- Implementation faults

V&V during design

Operational phase

- Hardware faults
- Configuration faults
- Operator faults

Fault tolerance (e.g. redundancy)

How many bugs do we have to expect?

DB Mobility Networks Logistics

How many „Bugs“ do we have to expect?

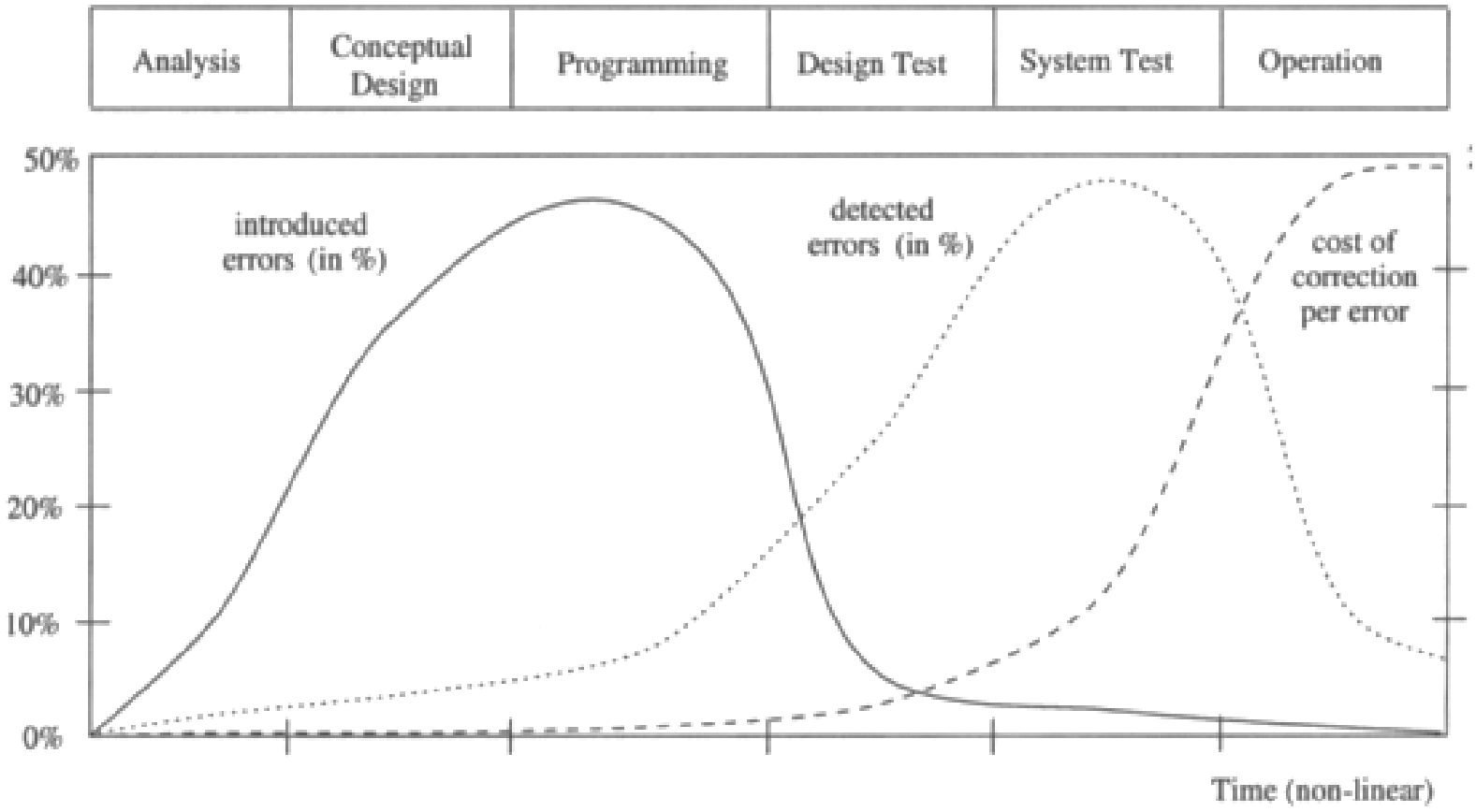
- Typical production type SW has 1 … 10 bugs per 1.000 lines of code (LOC).
- Very mature, long-term, well proven software: 0,5 bugs per 1.000 LOC
- Highest software quality ever reported :
  - *Less than 1 bug per 10.000 LOC*
  - At cost of more than 1.000 US$ per LoC *(1977)*
  - *US Space Shuttle with 3 m LOC* costing 3b US$ (out of 12b$ total R&D)

➔ Cost level not typical for the railway sector (< 100€/LoC)

- Typical ETCS OBU kernel software size is about 100.000 LOC or more
  - That means: 100 … 1.000 undisclosed defects per ETCS OBU
  - Disclosure time of defects can vary between a few days …. thousands of years

Source: K-R. Hase: „Open Proof in Railway Safety Software”, FORMS/FORMAT Conference, December 2-3, 2010, Braunschweig, Germany

Distribution and cost of bugs



Early V&V reduces cost!

V&V: Verification and Validation

Verification	Validation
„Am I building the system right?”	„Am I building the right system?”
Check consistency of development phases	Check the result of the development
Conformance of designs/models and their specification	Conformance of the finished system and the user requirements
Objective; can be automated	Subjective; checking acceptance
Fault model: Design and implementation faults	Fault model: problems in the requirements
Not needed if implementation is automatically generated from specification	Not needed if the specification is correct (very simple)

V&V techniques

Static

- What: any artefact (documentation, model, code)
- How: without execution
- E.g.: review, static analysis

Dynamic

- What: executable artefacts (model, code…)
- How: with execution
- E.g.: simulation, testing

RECAP: REQUIREMENTS

Learning outcomes

- Explain the properties and good practices of textual requirements (K2)

Requirement and specification

Requirement

- Vision, request, expectation from
  - Users
  - Stakeholders (authority, management, operator…)
- Basis for validation

Specification

- Request transformed for designer and developers
- Result of analysis (abstraction, structuring)
- Basis for verification

Definition of a requirement

"A condition or capability needed by a user to solve a problem or achieve an objective" (IEEE)

"A condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents" (IEEE)

Properties of good requirements

- **Identifiable + Unique** (unique IDs)
- **Consistent** (no contradiction)
- **Unambiguous** (one interpretation)
- **Verifiable** (e.g. testable to decide if met)

Captured with special statements and vocabulary

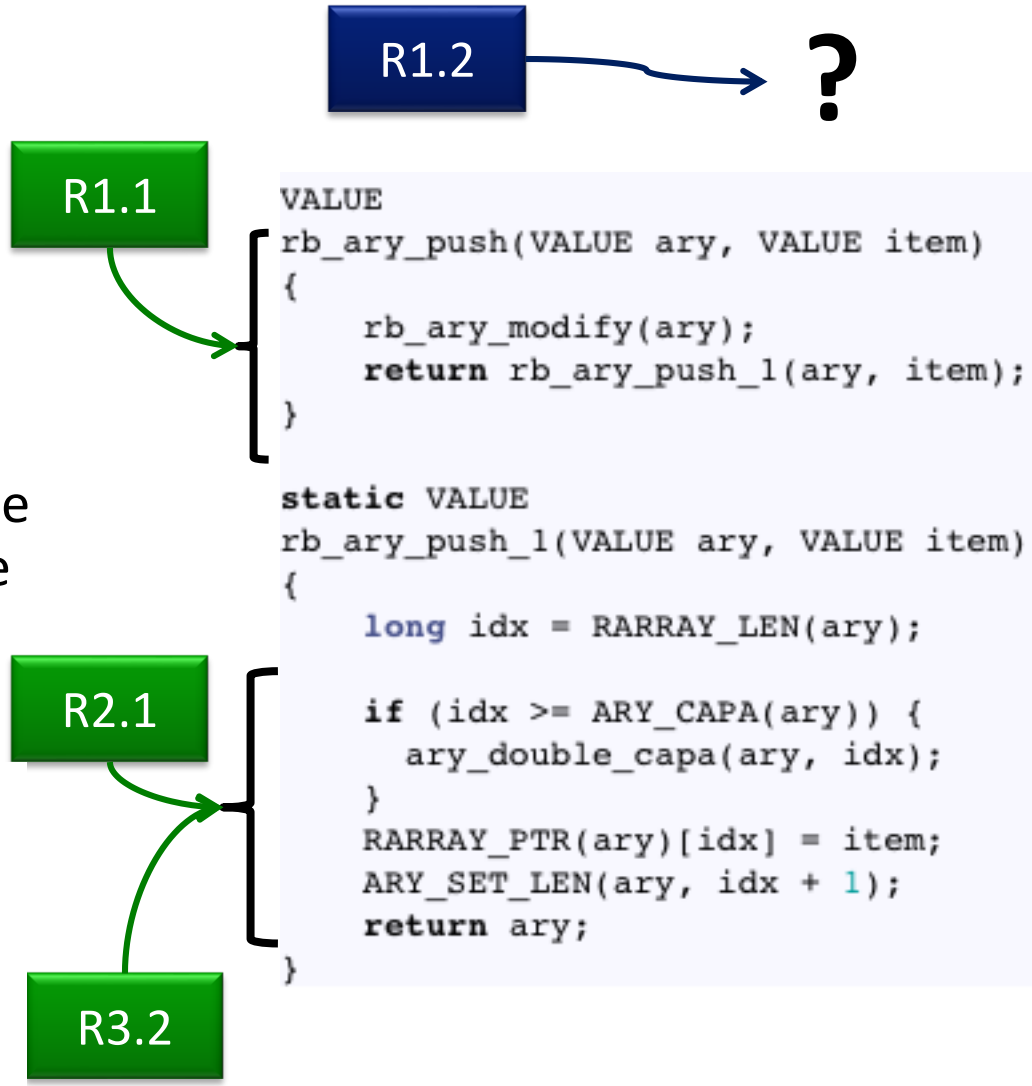
Good practices for writing textual requirements

a short description (stand-alone sentence / paragraph) of the problem and not the solution

- English phrasing:
  - Pattern: Subject Auxiliary Verb Object Conditions
  - E.g.: The system shall monitor the room's temperature when turned on.
- Use of auxiliaries (see RFC 2119)
  - Positive: SHALL / MUST > SHOULD > MAY
  - Negative: MUST NOT > SHOULD NOT
  - They specify priorities!

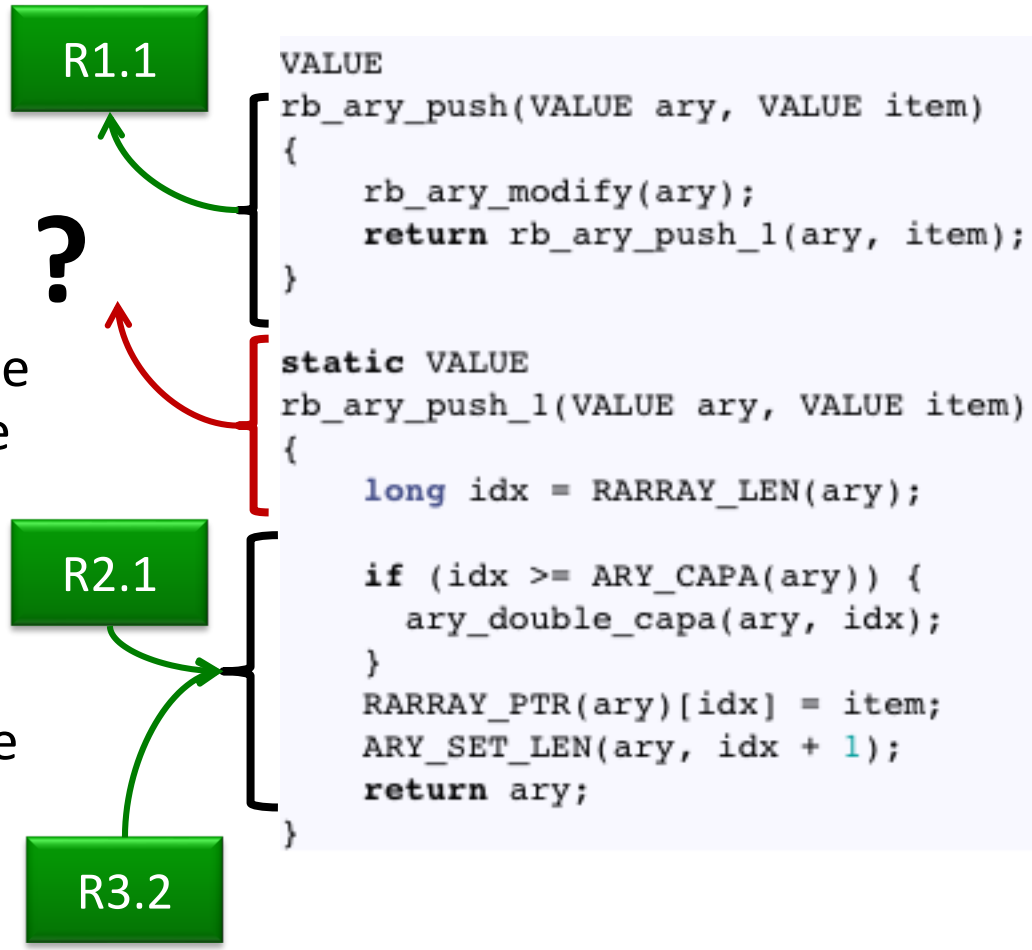
The Concept of Traceability

- Traceability is a core **certification concept**
  - For safety-critical systems
  - See safety standards (DO-178C, ISO 26262, EN 50126)
- **Forward traceability**:
  - From each requirement to the corresponding lines of source code (and object code)
  - Show responsibility



The Concept of Traceability

- Traceability is a core **certification concept**
  - For safety-critical systems
  - See safety standards (DO-178C, ISO 26262, EN 50126)
- **Forward traceability**:
  - From each requirement to the corresponding lines of source code (and object code)
  - Show responsibility
- **Backward traceability**:
  - From any lines of source code to one ore more corresponding requirements
  - No extra functionality



Anti-patterns

1. The system should be safe
2. The system shall use Fast Fourier Transformation to calculate signal value.
3. The system shall continue normal operation soon after a failure.
4. Sensor data shall be logged by a timestamp
5. Unauthorized personnel could not access the system

Too general / high-level

Describes a solution (and not only the problem)

Imprecise (how to verify „soon”?)

Passive should be avoided!

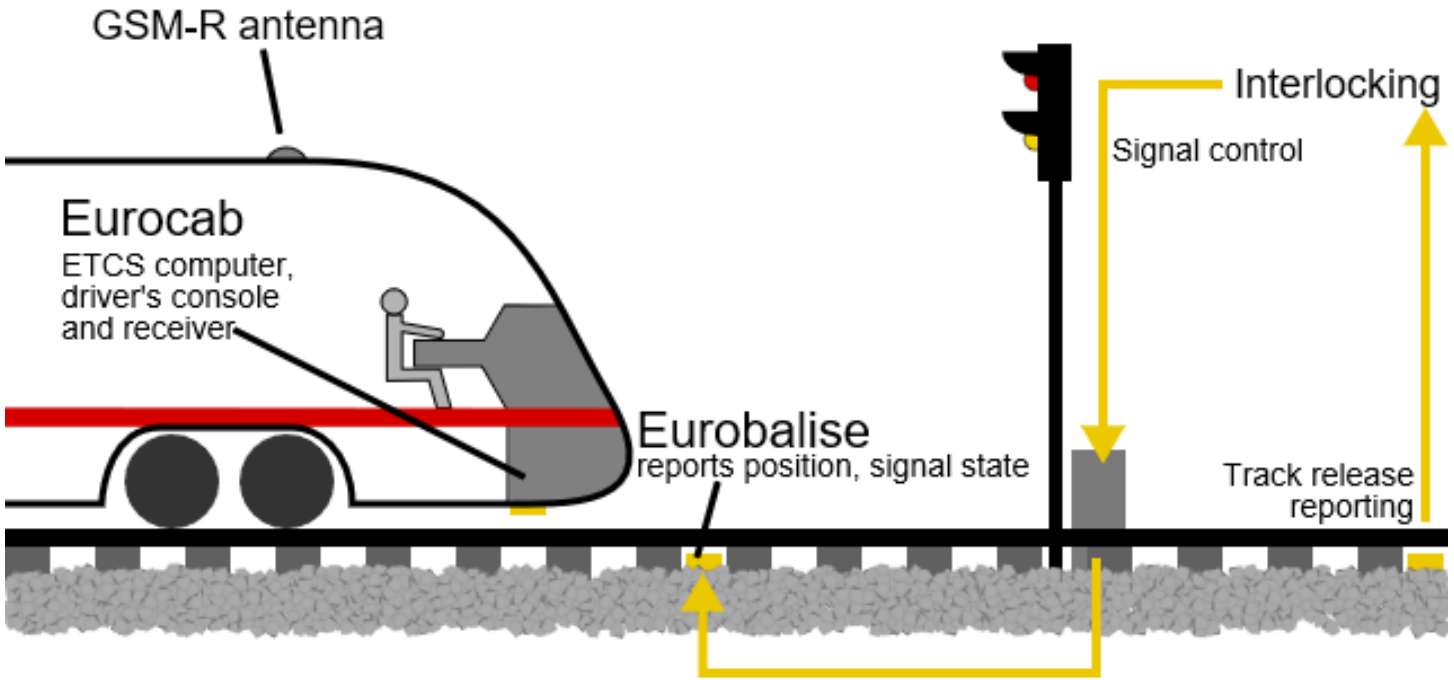
Use specific auxiliaries!

How to identify missing or inconsistent requirements?

Example requirements: ETCS

- European Rail Traffic Management System (ERTMS)
  - European Train Control System (ETCS) + GSM-R

http://www.era.europa.eu/Core-Activities/ERTMS/Pages/Set-of-specifications-3.aspx



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

Example requirements: ETCS

3.4.1 Balise Configurations – Balise Group Definition

3.4.1.1 A balise group shall consist of between one and eight balises.

3.4.1.2 In every balise shall at least be stored:

a) The internal number (from 1 to 8) of the balise

b) The number of balises inside the group

c) The balise group identity.

3.4.3.2 A balise may contain directional information, i.e. valid either for nominal or for reverse direction, or may contain information valid for both directions. In level 1, this information can be of the following type (please refer to section 3.8.5):

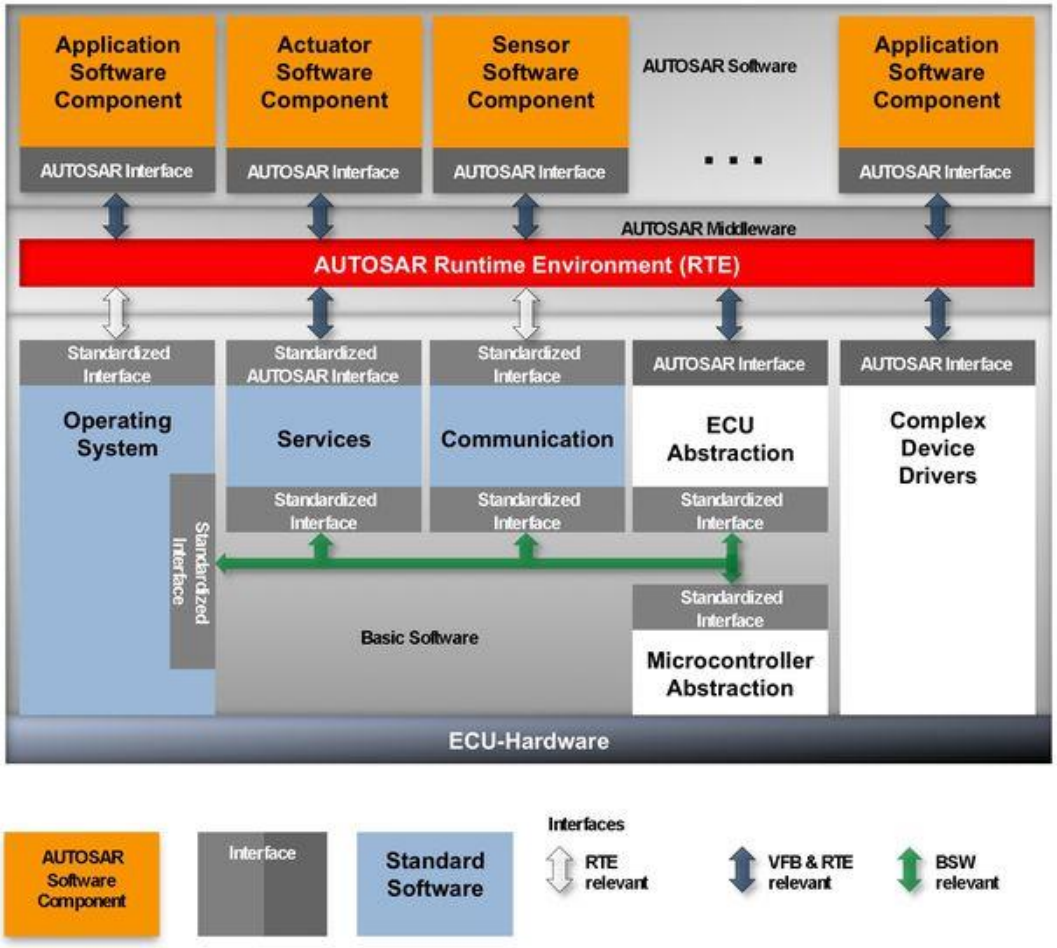
a) Non-infill

b) Intentionally deleted

c) Infill.

Example requirements: AUTOSAR

AUTomotive Open System Architecture



https://www.autosar.org/specifications/

Example requirements: AUTOSAR

3.1 [RS_PO_00001] AUTOSAR shall support the transferability of software.

⌜

Type:	Valid
Description:	AUTOSAR shall enable OEMs and suppliers to transfer software across the vehicle network and to reuse software.
Rationale:	Transferring software across the vehicle network allows overall system scaling and optimization. Redevelopment of software is expensive and error prone.
Use Case:	Application software is reusable across different product lines and OEMs. Scaling and optimizing of vehicle networks by transferring application software. Basic software is reusable across different ECUs and domains.
Dependencies:	RS_PO_00003, RS_PO_00004, RS_PO_00007, RS_PO_00008
Supporting Material:	--

High-level requirement

3 Requirements Tracing

The following table references the requirements specified in **[RS_ProjectObjectives]** and links to the fulfilments of these.

Requirement	Description	Satisfied by
RS_PO_00001	AUTOSAR shall support the transferability of software.	RS_Main_00060, RS_Main_00100, RS_Main_00130, RS_Main_00140, RS_Main_00150, RS_Main_00270, RS_Main_00310, RS_Main_00400, RS_Main_00410, RS_Main_00440, RS_Main_00450, RS_Main_00460, RS_Main_00480

Traceability

[SWS_EcuM_03022]⌈The SHUTDOWN phase handles the controlled shutdown of basic software modules and finally results in the selected shutdown target OFF or RESET.⌋(SRS_ModeMgm_09072)

Low-level requirement

Agile requirements: User stories

"As a <type of user>, I want <some goal> so that <some reason>."

- (Many different templates)
- Index card format
- “Just-in-time requirements”
- Connected to acceptance tests (→BDD)

RECAP: REVIEW PROCESS

Based on ISTQB Foundation Level Syllabus

Learning outcomes

- Recall the different types of review processes (K1)

Levels of formality in review

Informal review

- No formal process
- Peer or technical lead reviewing

Walkthrough

- Meeting led by author
- May be quite informal

Technical review

- Documented process
- Review meeting with experts
- Pre-meeting preparations for reviewers

Inspection

- Formal process
- Led by a trained moderator

Source: ISTQB CTFL

Activities of a formal review

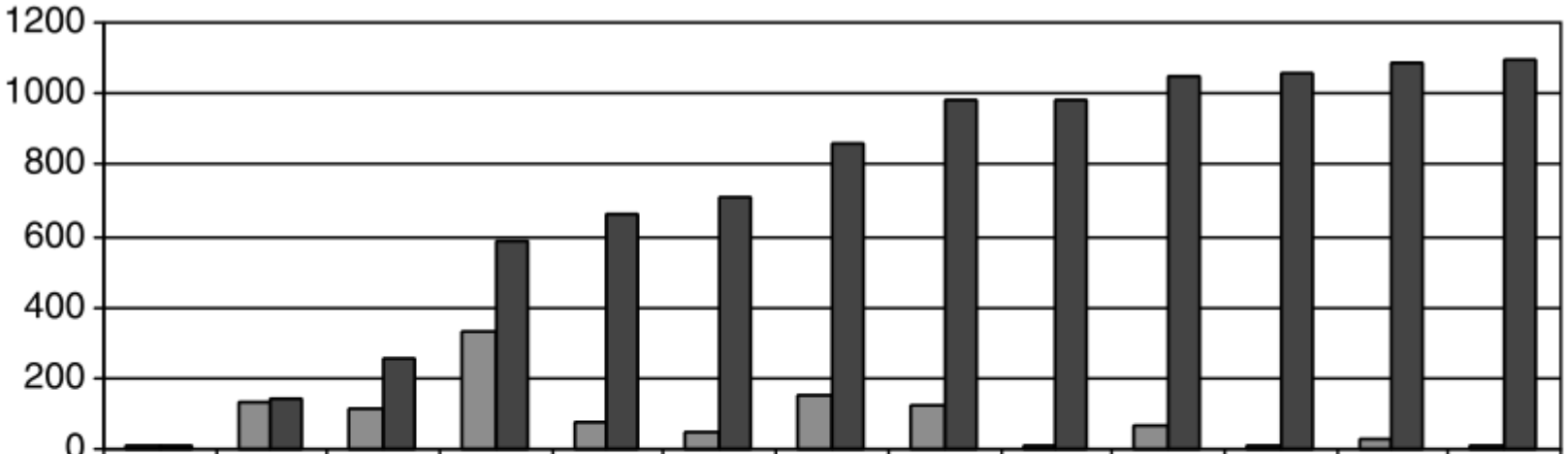
Planning	• Defining review criteria • Allocating roles
Kick-off	• Distributing documents • Explaining objectives
Individual preparation	• Reviewing artefacts • Noting potential defects, questions and comments
Review meeting	• Discussing and logging results • Noting defects, making decisions
Rework	• Fixing defects • Recording updated status
Follow-up	• Checking fixes • Checking on exit criteria

Source: ISTQB CTFL

Recommendations for reviews

- Thorough review is time consuming
  - Usually 5-10 pages / hour
  - Can be 1 page / hour
- Increasing the number of pages to review can greatly reduce the defects found
  - Practical limits: meeting is 2 hours, max 40 pages

Data on safety-critical projects



	fs	fs rev	des	des rev	code	code rev	ut des	ut run	int des	int run	sys des	sys run	engin e
correction	10	133	109	332	78	46	149	123	7	64	7	24	13
cumulative	10	143	252	584	662	708	857	980	987	1051	1058	1082	1095

Fig. 2 Corrections found at each phase and cumulative totals

fs – functional specification
fs rev – fs review
des – design
des rev – review
ut des – unit test design
ut run – ut execution
int – integration test
sys – system test

Source: The Economics of Unit Testing, ESE 11: 5–31, 2006

REVIEW CRITERIA

Learning outcomes

- List typical review criteria for requirements and specifications (K1)
- Perform review of requirements and specifications (K3)

Typical review criteria

Completeness	• Functions • References
Consistency	• Internal and external • Traceability
Implement-ability	• Resources • Usability, Maintainability • Risks: budget, technical, environmental
Verifiability	• Specific • Unambiguous • Measurable

Criteria from IEEE Std 830-1998

Correct

- Every requirement stated therein is one that the software shall meet
- Consistent with external sources (e.g. standards)

Unambiguous

- Every requirement has only one interpretation
- Formal or semi-formal specification languages can help

Complete

- For every (valid, invalid) input there is specifies behavior
- TBD only possible resolution

Consistent

- No internal contradiction, terminology

Ranked for importance and/or stability

- Necessity of requirements

Verifiable

- Can be checked whether the requirement is met

Modifiable

- Not redundant, structured

Traceable

- Source is clear, effect can be referenced

Criteria from IEEE Std 29148-2011

Necessary

- If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities

Implementation Free

- Avoids placing unnecessary constraints on the design

Unambiguous

- It can be interpreted in only one way; is simple and easy to understand

Consistent

- Is free of conflicts with other requirements

Complete

- Needs no further amplification (measurable and sufficiently describes the capability)

Singular

- Includes only one requirement with no use of conjunctions

Feasible

- Technically achievable, fits within system constraints (cost, schedule, regulatory…)

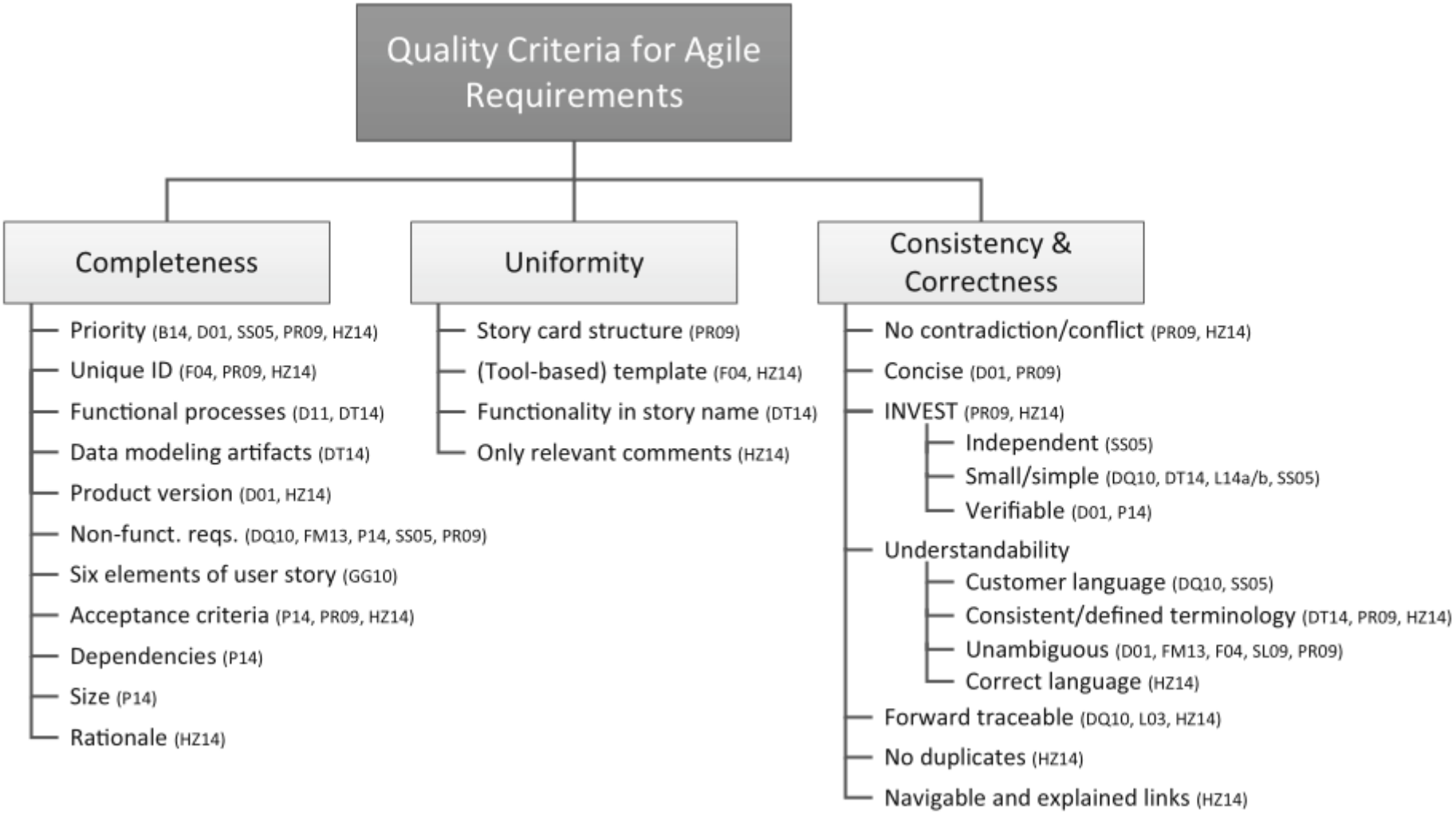
Traceable

- Upwards traceable to the stakeholder statements; downwards traceable to other documents

Verifiable

- Has the means to prove that the system satisfies the specified requirement

Quality criteria for agile requirements



Source: Heck, P. & Zaidman, A. A systematic literature review on quality criteria for agile requirements specifications. Software Qual J (2016). DOI: [10.1007/s11219-016-9336-4](https://doi.org/10.1007/s11219-016-9336-4)

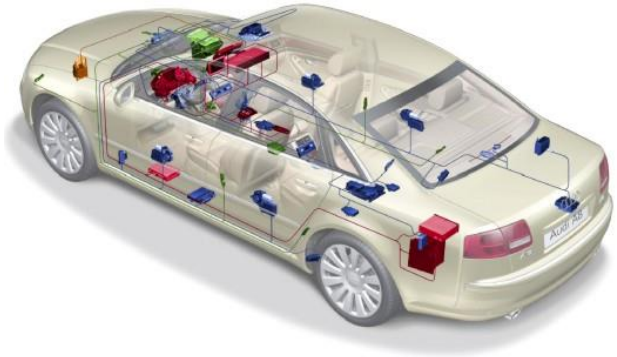
V&V TECHNIQUES IN CRITICAL SYSTEMS

Learning outcomes

- **Recall the safety concepts of critical systems (K1)**
- **List typical activities required by standards (K1)**

Safety-critical systems

Safety: “The expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered.” [IEEE]



Certification

- Certification by safety authorities

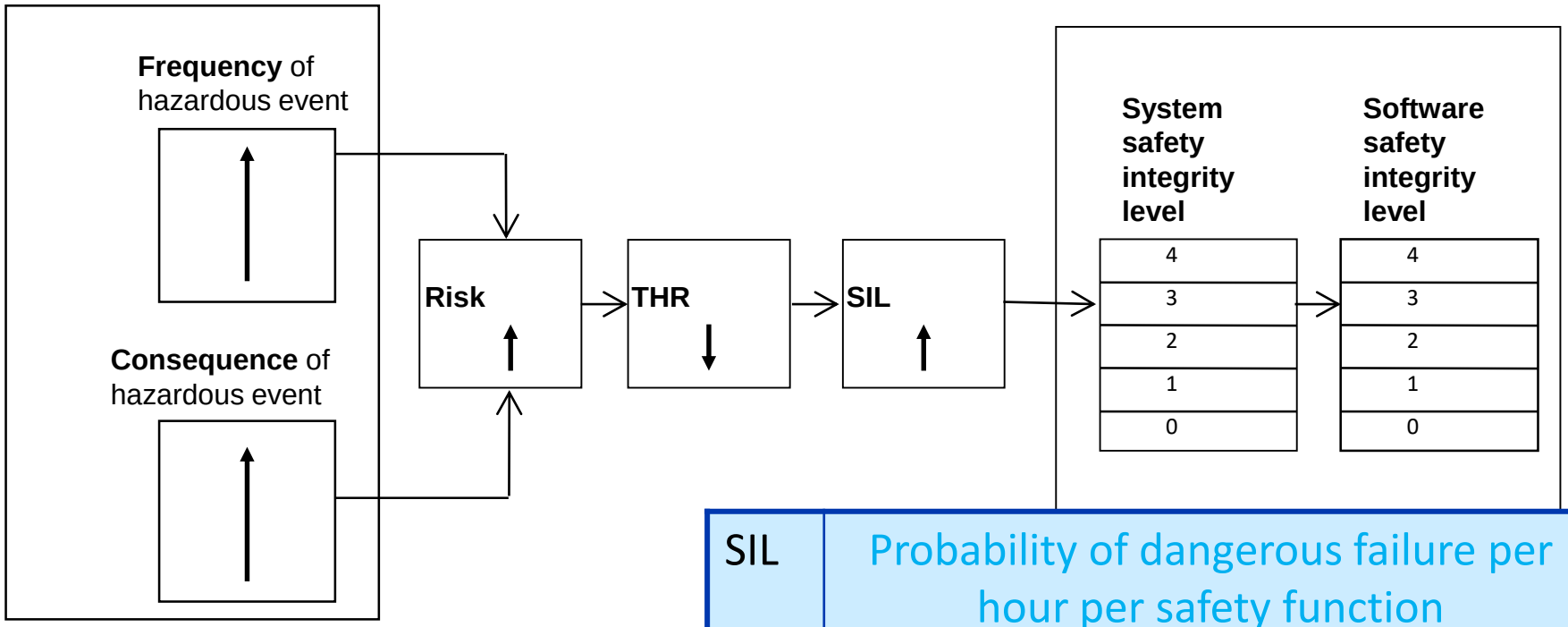
- Basis of certification: Standards
  - IEC 61508: Generic standard (for electrical, electronic or programmable electronic systems)
  - DO178B/C: Software in airborne systems
  - EN50128: Railway (software)
  - ISO26262: Automotive

Safety concepts

- Safety function
  - Intended to achieve or maintain a safe state
- Safety integrity
  - Probability of a safety-related system satisfactorily performing the required safety functions under all stated conditions and within a stated period of time
- Safety Integrity Level (SIL)
  - Based on risk analysis
  - Tolerable Hazard Rate (THR)

Basics of determining SIL

Risk analysis -> THR -> SIL



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

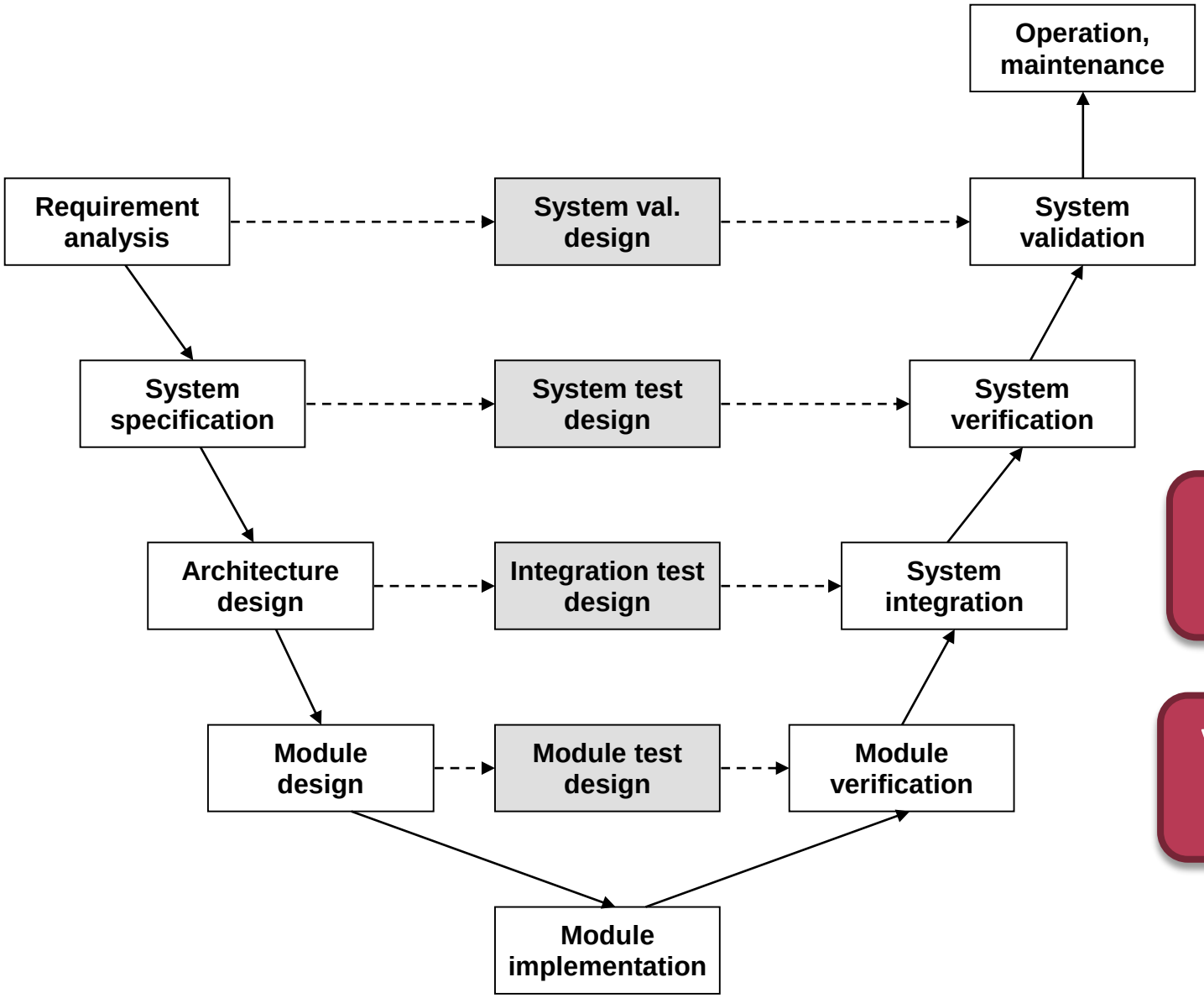
15 years lifetime:
1 failure in case
of 750 equipment

Demonstrating SIL requirements

Different approaches for types of failures

- Random failures (e.g. HW)
  - Qualitative analysis (statistics, experiments…)
- Systematic failures (e.g. SW)
  - Rigor in the engineering
  - Recommendations for each SIL
  - Process, techniques, documentation, responsibilities

Example: Process (V model)



Well-defined phases

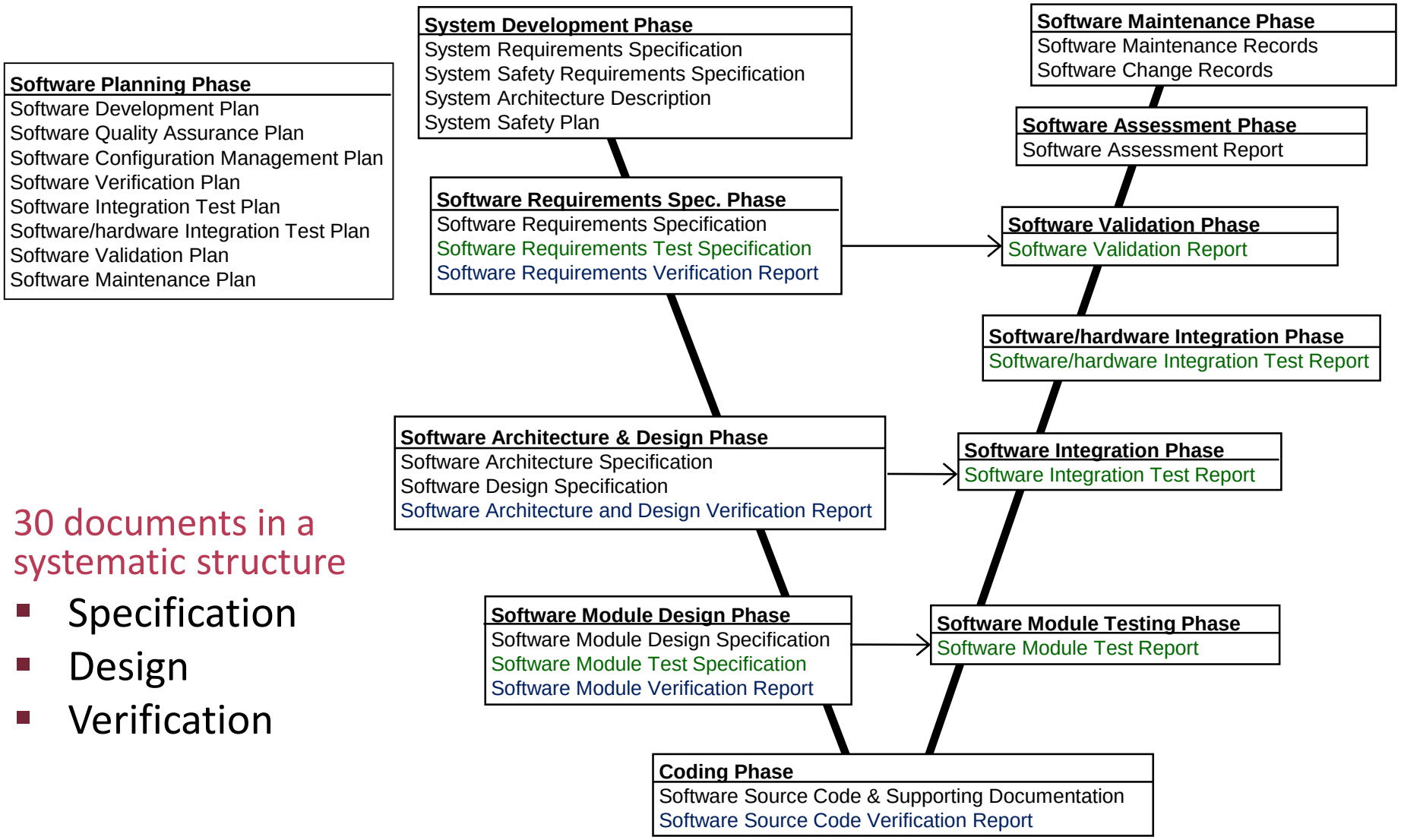
Verification of each step

Example: Techniques (EN 50128)

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

- 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

Example: Document structure (EN50128)



30 documents in a systematic structure

- Specification
- Design
- Verification

Example: Responsibilities (EN 50128)