The role of development standards in software V&V

Istvan Majzik
majzik@mit.bme.hu

Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Synopsis

- Introduction
- Verification in the requirement phase
- Architecture verification and evaluation
- Verification of the detailed design
  - Classic techniques
  - Formal methods: model checking, equivalence checking
  - Advanced methods: formal verification of extra-functional properties and timed behavior, handling complex designs (large state spaces)
- Verification of the source code
  - Code review, abstract interpretation, symbolic execution
  - Classic techniques of proving program correctness
- Testing and test case generation
  - Test design at unit level
  - Integration and system testing
  - Model based testing and test case generation
- Validation and assessment
- V&V in the maintenance phases
- Integrated approaches

The role of development standards

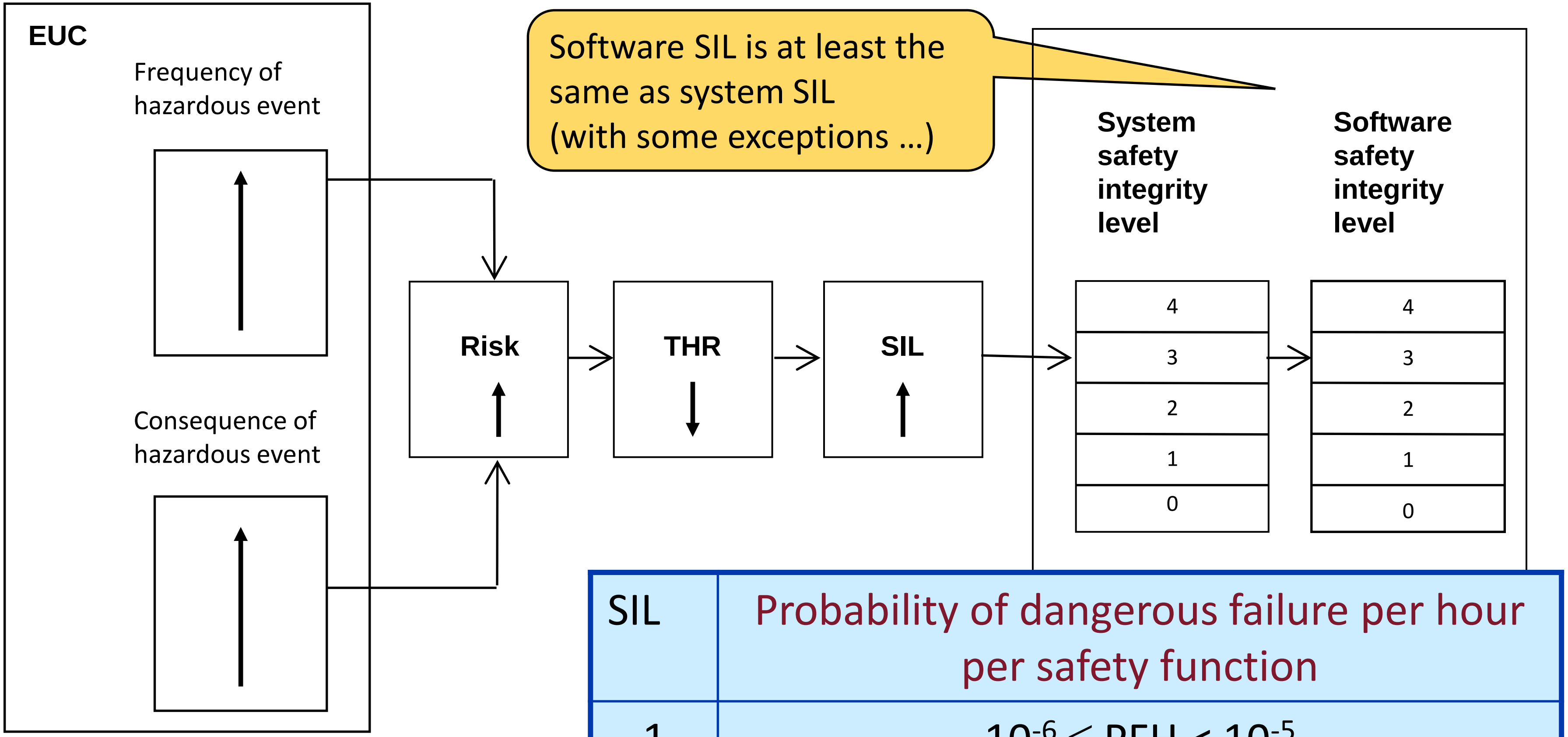
How systematic V&V is realized?

Use of standards: Safety critical systems

- Standards for development
  - IEC 61508: Functional safety in electrical / programmable electronic systems
  - EN 50128: Railway control software
  - ISO 26262: Automotive software
  - DO 178B: Airborne software
- Specification of safety functions
  - Functionality: Intended to achieve or maintain a safe state
  - Safety integrity: Probability that a safety-related system satisfactorily performs the required safety functions
    (under all stated conditions and within a stated period of time)
- Safety integrity levels
  - Safety integrity assignment to functions: Based on risk analysis (of failures)
    - Continuous operation: Tolerable rate of failures
    - On demand operation: Tolerable probability of failure
  - Tolerable Hazard Rate:
    - Categories based on numerical ranges: SIL 1, 2, 3, 4

Determining SIL

- Hazard identification and risk analysis -> Target failure measure


EUC
Frequency of hazardous event
Consequence of hazardous event
Risk
THR
SIL
System safety integrity level
Software safety integrity level
4
3
2
1
0
4
3
2
1
0

Software SIL is at least the same as system SIL (with some exceptions …)

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	$10^{-9} \leq \text{PFH} < 10^{-8}$

Risk analysis
-> Function THR
-> Function SIL
-> (Sub)system SIL

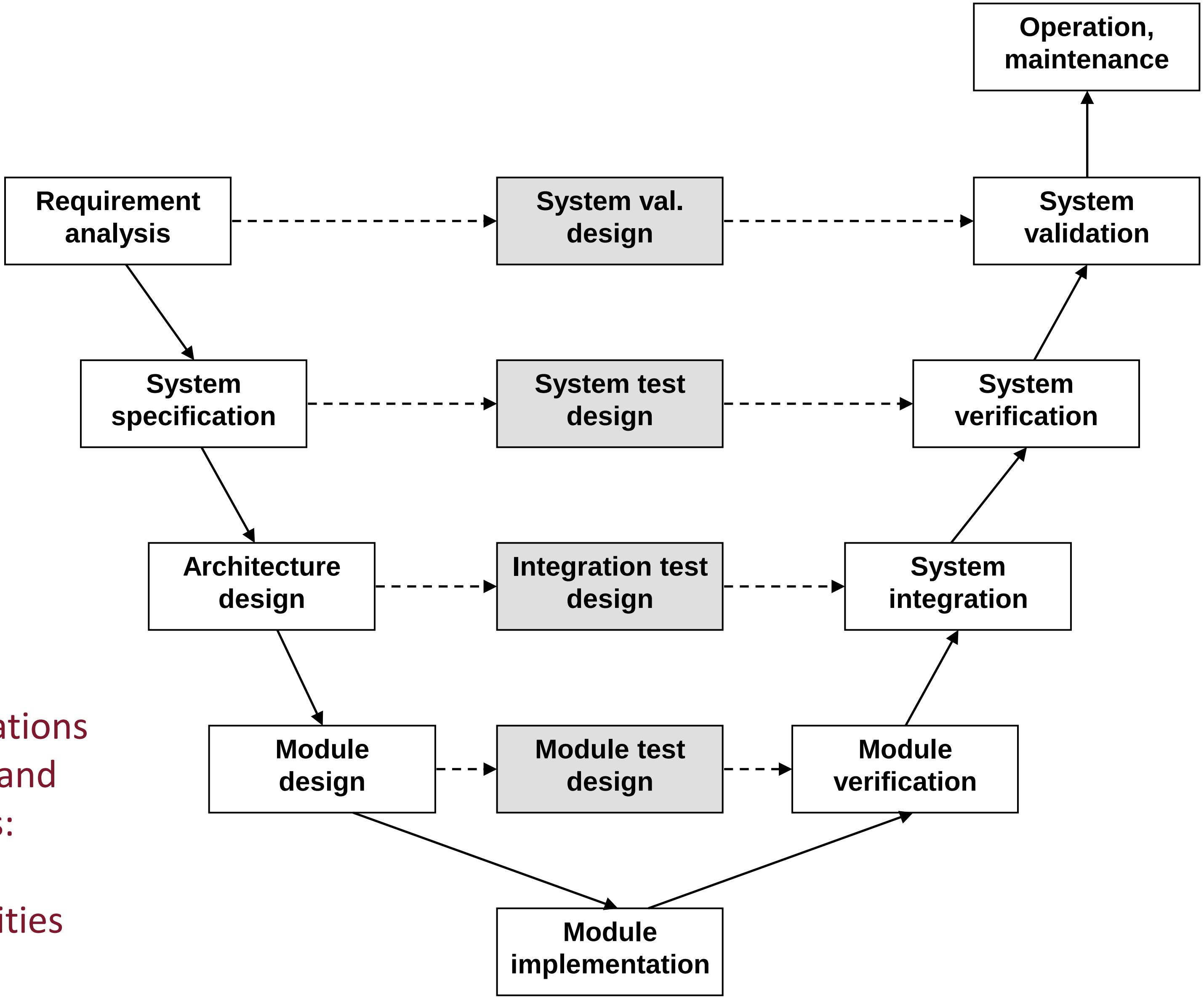
Demonstrating SIL requirements

- Safety case:
  - Documented demonstration that the product complies with the specified safety requirements (functional + safety integrity)
  - Evidence is based on verification and validation
- Random failure integrity (for hardware):
  - Quantitative approach: Based on statistics, experiments
  - Computation of system failure rate using component fault rate data from reliability handbooks
- Systematic failure integrity (for software):
  - Quantitative approach is not possible (missing reliability data)
  - Qualitative approach: Prescribing rigor in the development
    1. Well-defined development process (life cycle)
    2. Mandatory / recommended techniques and measures
    3. Organizational structure: Independence of persons / roles
    4. Precise documentation

1. The development process (life cycle)

- Strict rules for proceeding to the next step: Important to verify the results of development
  - High costs of late corrections (esp. during operation)
  - The risk caused by remaining failures may be high
- Typically results in a static process (e.g., V-model)
  - Well-defined steps
  - Requirements and environment known in advance
- Other characteristics:
  - Evidences collected for the safety case
  - Assessment (independent review)
  - Certification and supervision by safety authorities, based on the development standard

Typical life-cycle model: V-model



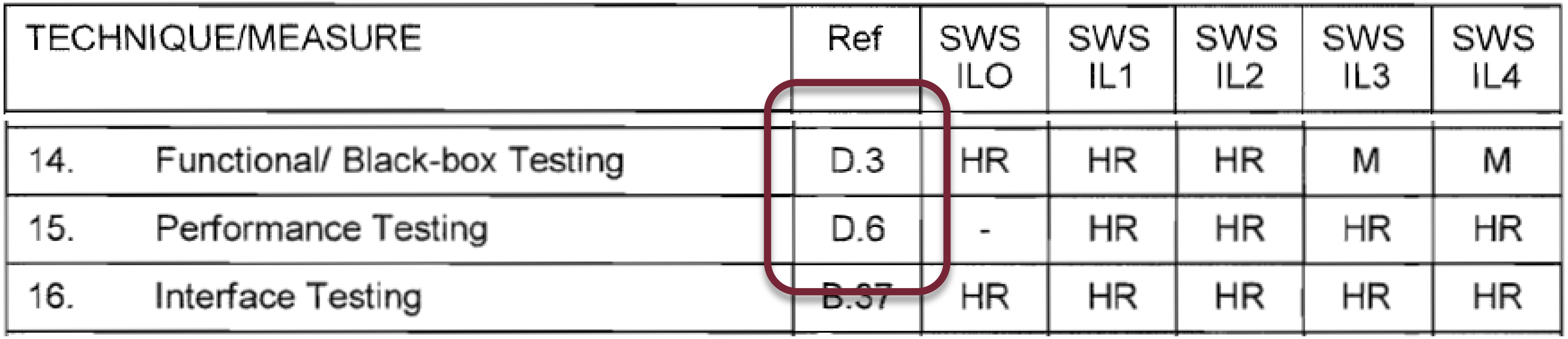
Well-defined relations between design and verification steps: Planning of the verification activities

2. Techniques and measures

- 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 is allowed
  - E.g., alternative or equivalent techniques are marked
- Hierarchy of techniques (references to sub-tables)

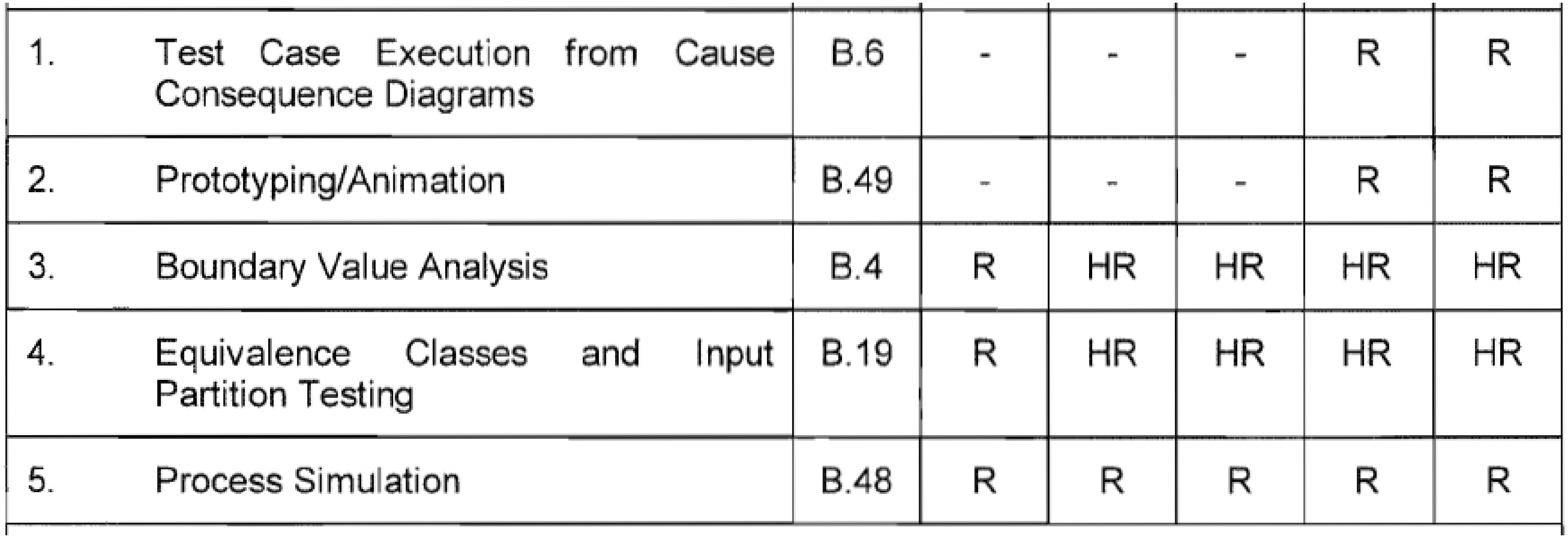
Example: Testing techniques (EN 50128)

- Software design and implementation:



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

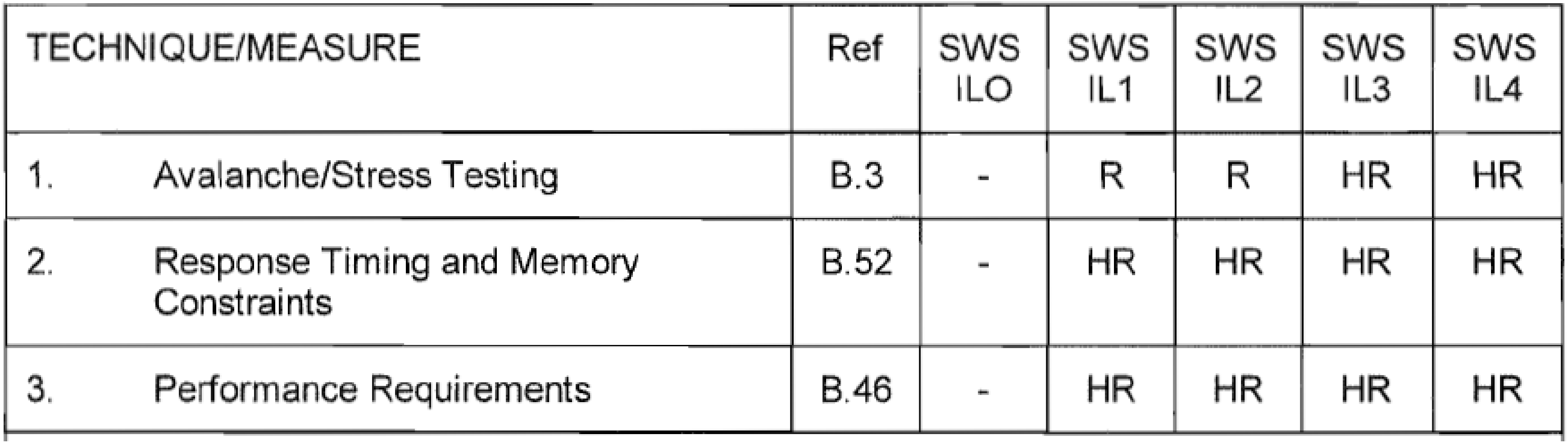
- Functional / black box testing (D3):



1.	Test Case Execution from Cause Consequence Diagrams	B.6	-	-	-	R	R
2.	Prototyping/Animation	B.49	-	-	-	R	R
3.	Boundary Value Analysis	B.4	R	HR	HR	HR	HR
4.	Equivalence Classes and Input Partition Testing	B.19	R	HR	HR	HR	HR
5.	Process Simulation	B.48	R	R	R	R	R

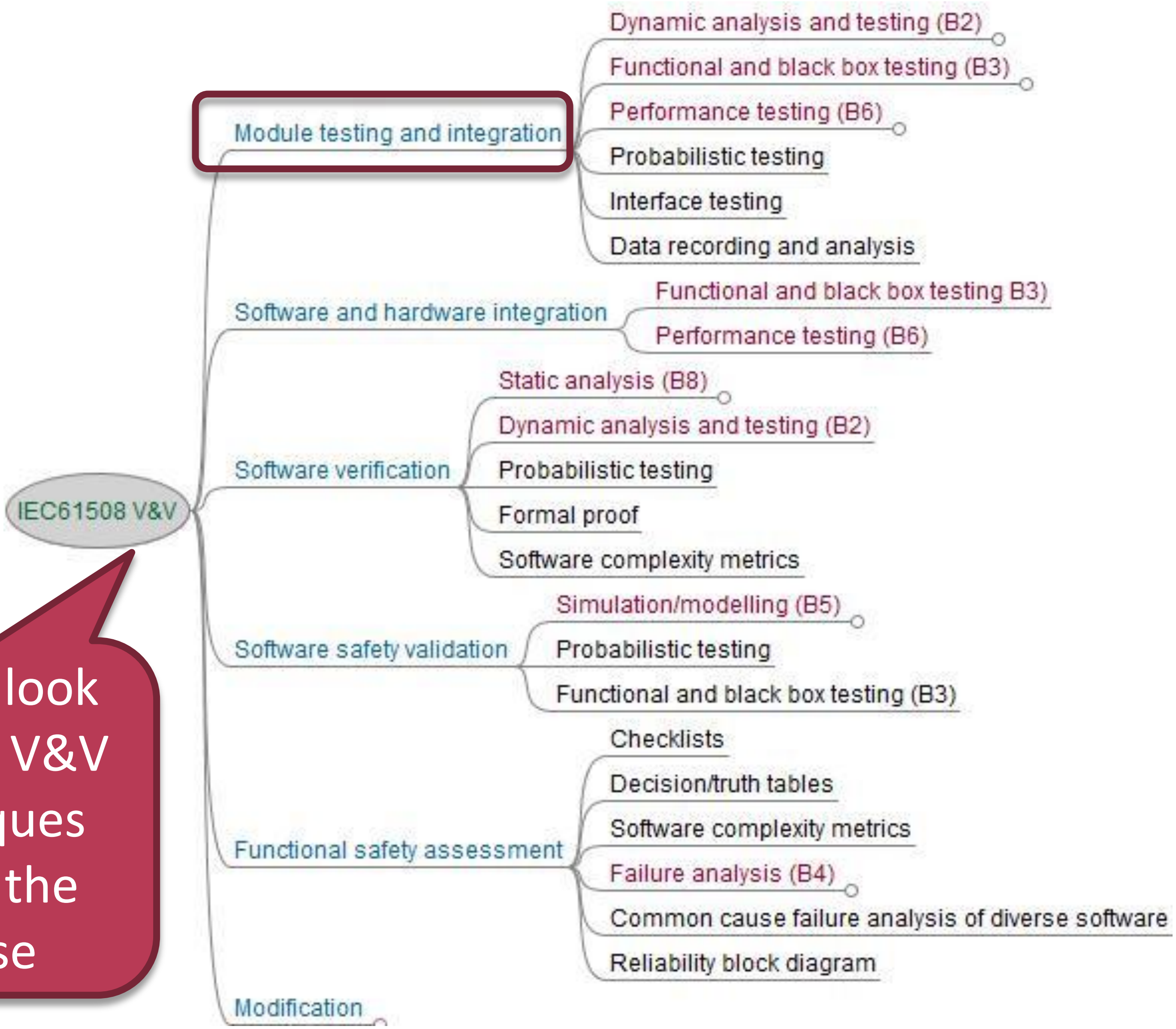
Example: Testing techniques (EN 50128)

- Performance testing (D6):



TECHNIQUE/MEASURE		Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Avalanche/Stress Testing	B.3	-	R	R	HR	HR
2.	Response Timing and Memory Constraints	B.52	-	HR	HR	HR	HR
3.	Performance Requirements	B.46	-	HR	HR	HR	HR

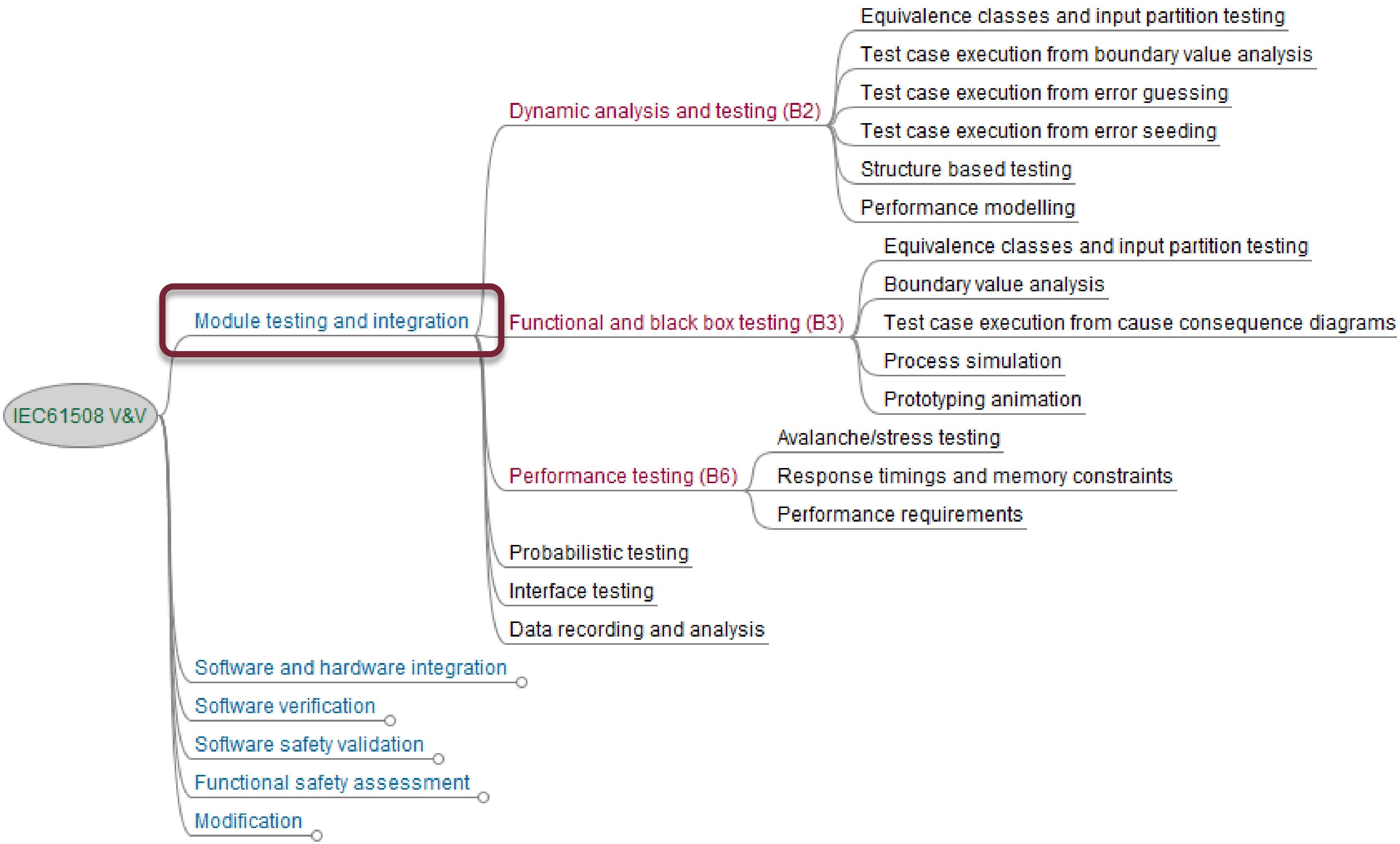
Example: Hierarchy of V&V methods (IEC 61508)



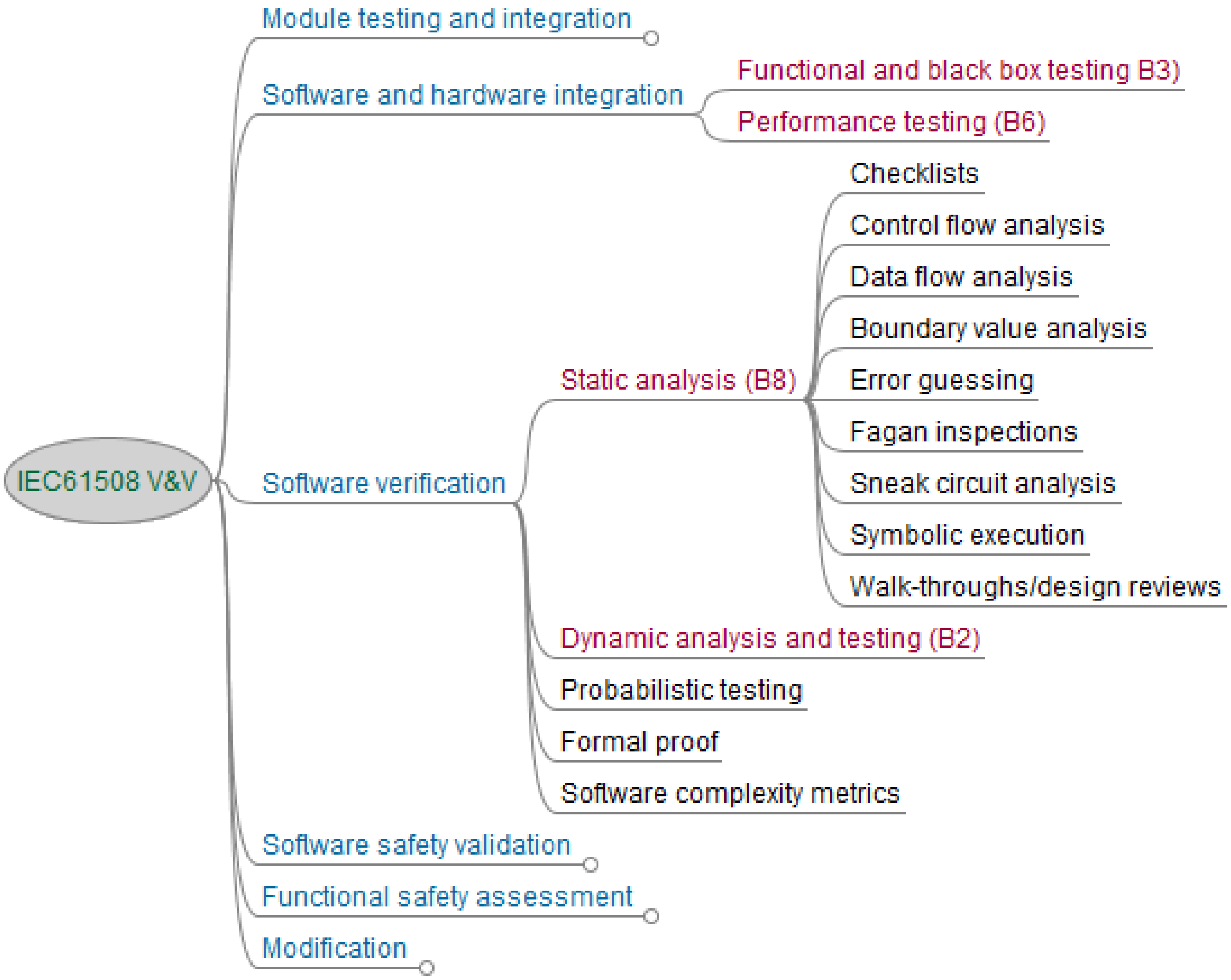
- IEC61508 V&V
  - Module testing and integration
    - Dynamic analysis and testing (B2)
    - Functional and black box testing (B3)
    - Performance testing (B6)
    - Probabilistic testing
    - Interface testing
    - Data recording and analysis
  - Software and hardware integration
    - Functional and black box testing B3)
    - Performance testing (B6)
  - Software verification
    - Static analysis (B8)
    - Dynamic analysis and testing (B2)
    - Probabilistic testing
    - Formal proof
    - Software complexity metrics
  - Software safety validation
    - Simulation/modelling (B5)
    - Probabilistic testing
    - Functional and black box testing (B3)
  - Functional safety assessment
    - Checklists
    - Decision/truth tables
    - Software complexity metrics
    - Failure analysis (B4)
    - Common cause failure analysis of diverse software
    - Reliability block diagram
  - Modification

We will look at these V&V techniques during the course

Example: Hierarchy of V&V methods (IEC 61508)

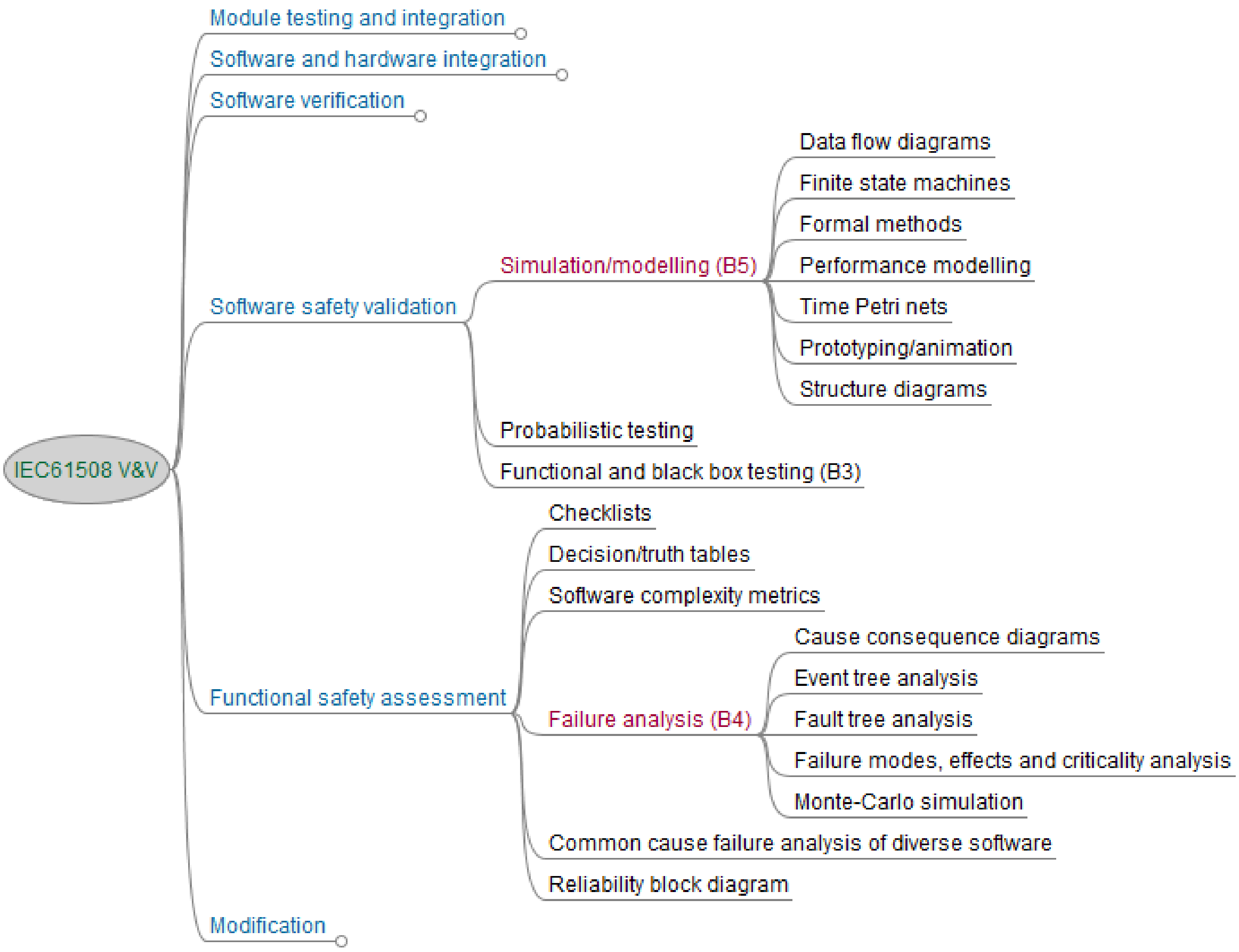

IEC61508 V&V
Module testing and integration
Dynamic analysis and testing (B2)
Equivalence classes and input partition testing
Test case execution from boundary value analysis
Test case execution from error guessing
Test case execution from error seeding
Structure based testing
Performance modelling
Functional and black box testing (B3)
Equivalence classes and input partition testing
Boundary value analysis
Test case execution from cause consequence diagrams
Process simulation
Prototyping animation
Performance testing (B6)
Avalanche/stress testing
Response timings and memory constraints
Performance requirements
Probabilistic testing
Interface testing
Data recording and analysis
Software and hardware integration
Software verification
Software safety validation
Functional safety assessment
Modification

Example: Hierarchy of V&V methods (IEC 61508)



- IEC61508 V&V
  - Module testing and integration
  - Software and hardware integration
    - Functional and black box testing B3)
    - Performance testing (B6)
  - Software verification
    - Static analysis (B8)
      - Checklists
      - Control flow analysis
      - Data flow analysis
      - Boundary value analysis
      - Error guessing
      - Fagan inspections
      - Sneak circuit analysis
      - Symbolic execution
      - Walk-throughs/design reviews
    - Dynamic analysis and testing (B2)
    - Probabilistic testing
    - Formal proof
    - Software complexity metrics
  - Software safety validation
  - Functional safety assessment
  - Modification

Example: Hierarchy of V&V methods (IEC 61508)

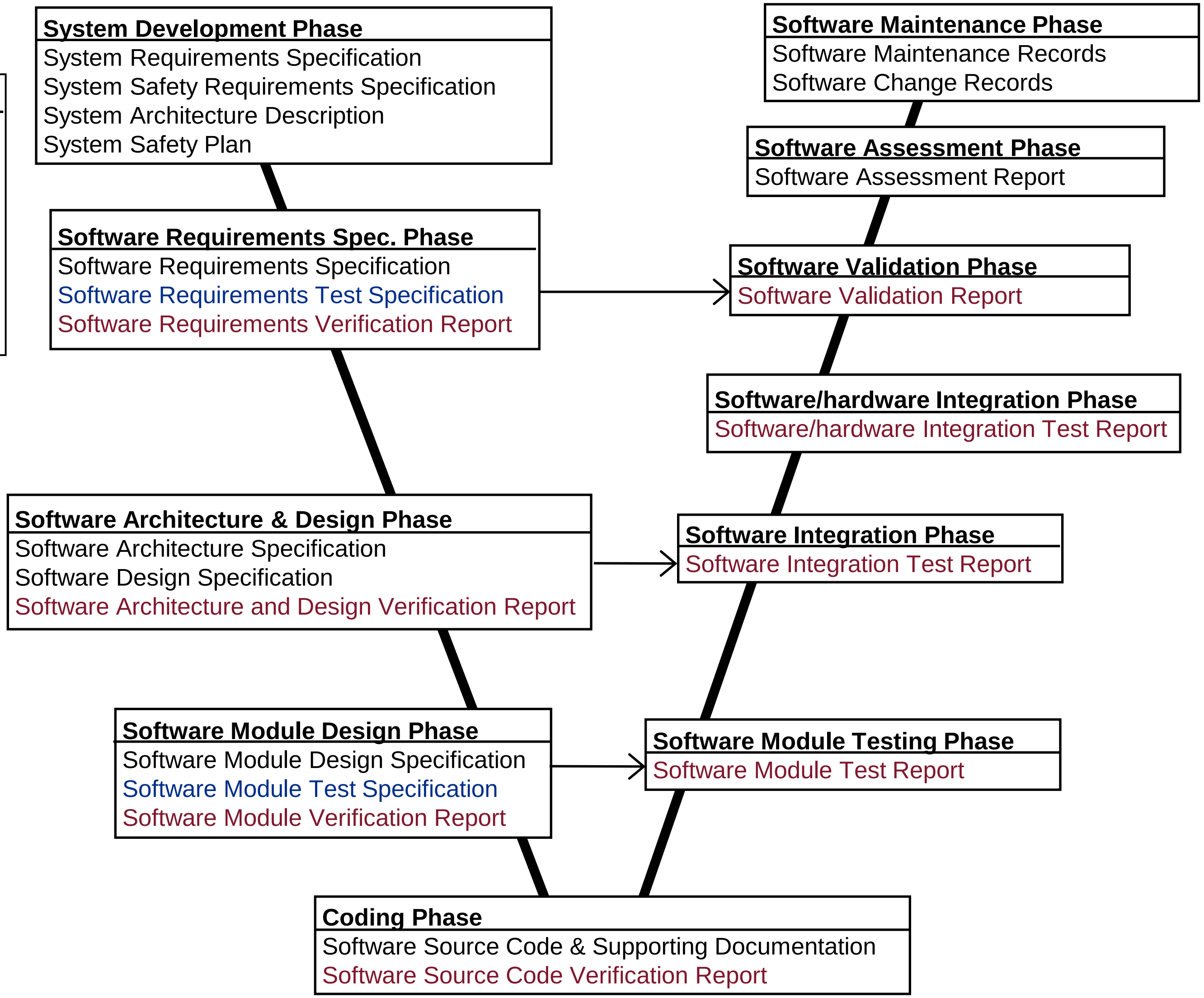

IEC61508 V&V
Module testing and integration
Software and hardware integration
Software verification
Software safety validation
Simulation/modelling (B5)
Data flow diagrams
Finite state machines
Formal methods
Performance modelling
Time Petri nets
Prototyping/animation
Structure diagrams
Probabilistic testing
Functional and black box testing (B3)
Functional safety assessment
Checklists
Decision/truth tables
Software complexity metrics
Failure analysis (B4)
Cause consequence diagrams
Event tree analysis
Fault tree analysis
Failure modes, effects and criticality analysis
Monte-Carlo simulation
Common cause failure analysis of diverse software
Reliability block diagram
Modification

3. Precise 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
  - Relationship between documents is specified ("based on", "includes")
  - Terminology, references, abbreviations are consistent
- Merging documents is allowed
  - If responsible persons (authors) shall not be independent

Example: Document structure (EN50128)

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



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

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

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

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

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

Software Module Testing Phase
Software Module Test Report

Software Integration Phase
Software Integration Test Report

Software/hardware Integration Phase
Software/hardware Integration Test Report

Software Validation Phase
Software Validation Report

Software Assessment Phase
Software Assessment Report

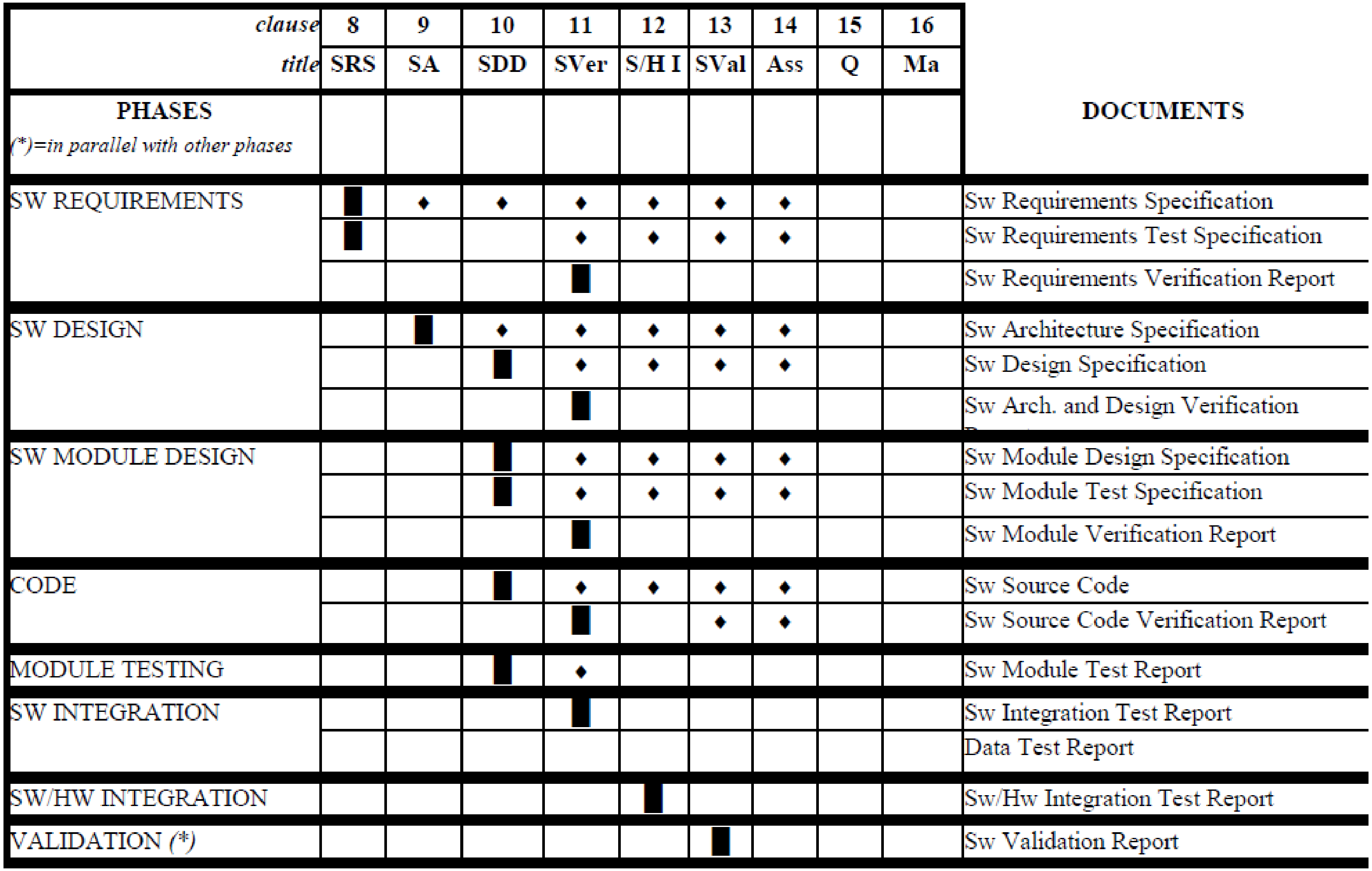
Software Maintenance Phase
Software Maintenance Records
Software Change Records

30 documents in a systematic structure

- Specification
- Design
- Verification

Example: Document cross reference table (EN50128)

- ■ Creation of a document
- ◆ Use of a document in a given phase

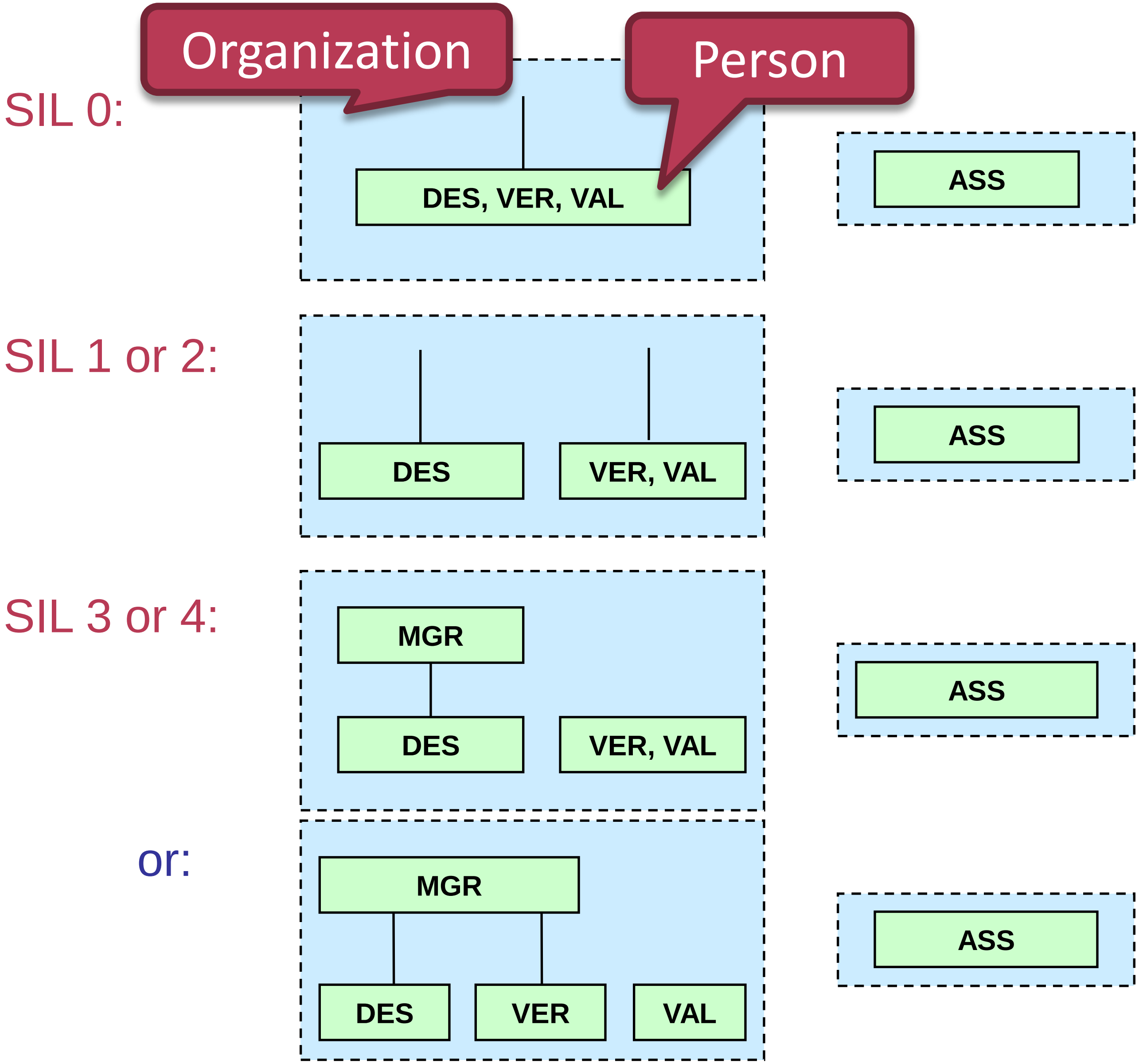


clause	8	9	10	11	12	13	14	15	16	
title	SRS	SA	SDD	SVer	S/H I	SVal	Ass	Q	Ma	
PHASES *(*)=in parallel with other phases*										**DOCUMENTS**
SW REQUIREMENTS	■	◆	◆	◆	◆	◆	◆			Sw Requirements Specification
	■			◆	◆	◆	◆			Sw Requirements Test Specification
				■						Sw Requirements Verification Report
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

4. Organization and independence of roles

- Safety management
  - Quality assurance
  - Safety Organization (responsible persons)
- 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

Example: Responsibilities (EN 50128)



Organization

Person

SIL 0:

DES, VER, VAL

ASS

SIL 1 or 2:

DES

VER, VAL

ASS

SIL 3 or 4:

MGR

DES

VER, VAL

ASS

or:

MGR

DES

VER

VAL

ASS

DES: Designer
VER: Verifier
VAL: Validator
ASS: Assessor
MAN: Project manager