Software and Systems Verification (VIMIMA01)

Introduction
Overview of V&V techniques

Istvan Majzik, Zoltan Micskei

Budapest University of Technology and Economics
Fault Tolerant Systems Research Group

Main topics of the course

- Overview (1)
  - V&V techniques, Critical systems
- Static techniques (2)
  - Verifying specifications
  - Verifying source code
- Dynamic techniques: Testing (7)
  - Developer testing, Test design techniques
  - Testing process and levels, Test generation, Automation
- System-level verification (3)
  - Verifying architecture, Dependability analysis
  - Runtime verification

Who is this course for?

Systems Engineer	• Requirements, verifying specification
Architect, Designer	• Modeling and verifying designs
Developer, Coder	• Verifying source code, unit testing
Test Designer	• Test processes and techniques
Test Engineer	• Test automation, integration and system tests
Safety Engineer	• Certification, development standards

Stereotypes

“Testing is destructive.”

“Testing is just pushing buttons and supplying values randomly.”

“If your are not good for a developer, you can be a tester.”

“Testing is boring.”

“I tested in the debugger…”

V&V (and testing) in reality

V&V (and testing) is creative!

How is this working?

How can I prove it works?

How should it work?

How can it fail?

V&V (and testing) is constructive!

Testers are not breaking the SW (it was broken)

Passion for quality

Testers help make the system better

V&V (and testing) requires a different mindset

Intuition

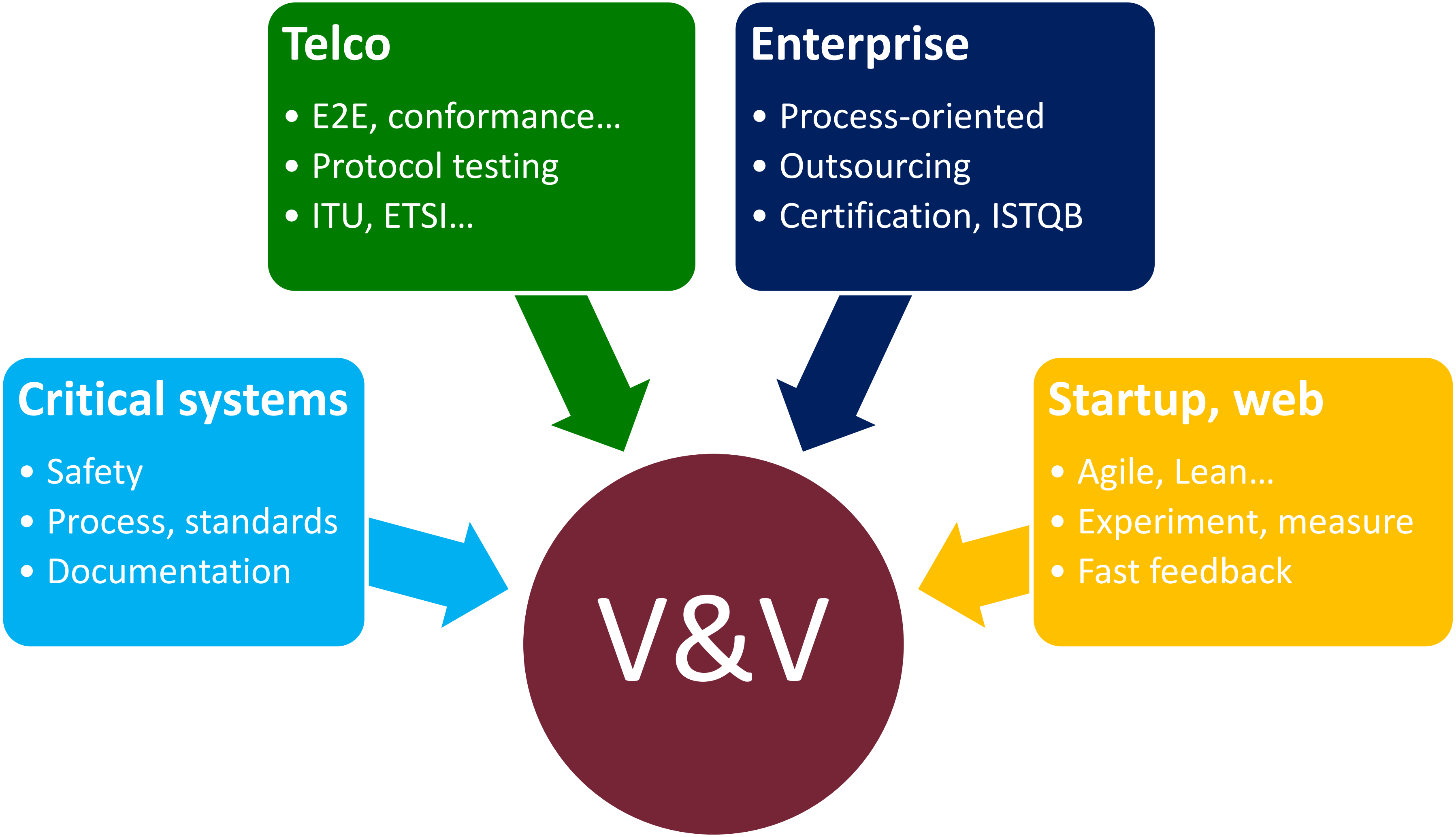
Attention to details

…

Systems level thinking

Specific knowledge

V&V is context dependent!



Telco

- E2E, conformance…
- Protocol testing
- ITU, ETSI…

Enterprise

- Process-oriented
- Outsourcing
- Certification, ISTQB

Critical systems

- Safety
- Process, standards
- Documentation

Startup, web

- Agile, Lean…
- Experiment, measure
- Fast feedback

V&V

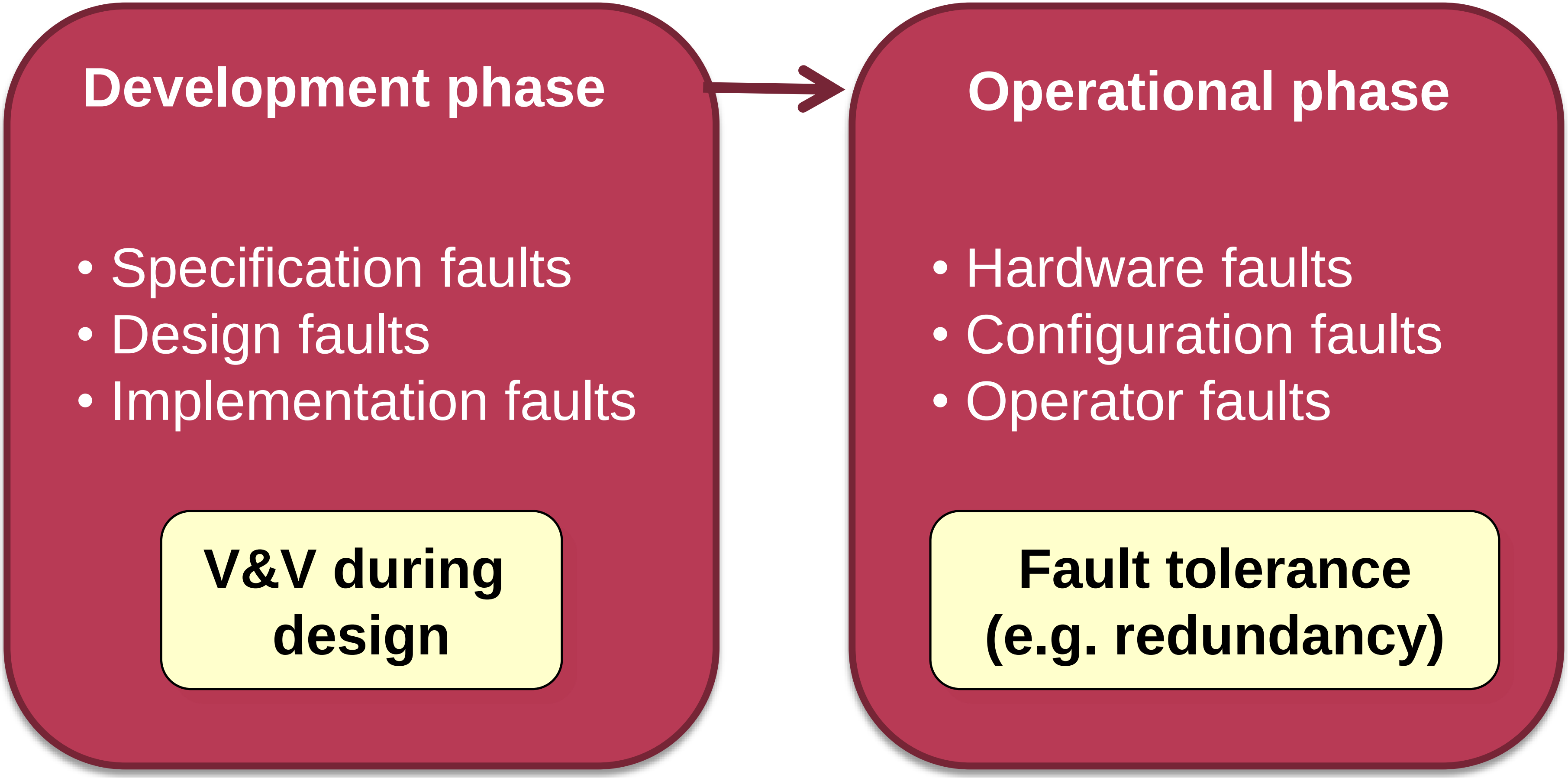
Useful resources (download now!)

- IEEE standards
  - 24765-2010 Systems and SW engineering – Vocabulary
    - SE VOCAB – online searchable form
  - 29148-2011 Requirements engineering
  - 29119 Software testing
    - Part 1 Concepts and definitions, Part 2 Test processes, Part 3 Test documentation, Part 4 Test techniques
- International Software Testing Qualifications Board (ISTQB)
  - Foundation Level Syllabus (2011)
  - Glossary of Testing Terms
- Hungarian Testing Board (HTB)
  - Glossary / Kifejezésgyűjtemény (magyar fordítás)

Useful events

MOTIVATION

Different kinds of faults



Development phase

- Specification faults
- Design faults
- Implementation faults

V&V during design

Operational phase

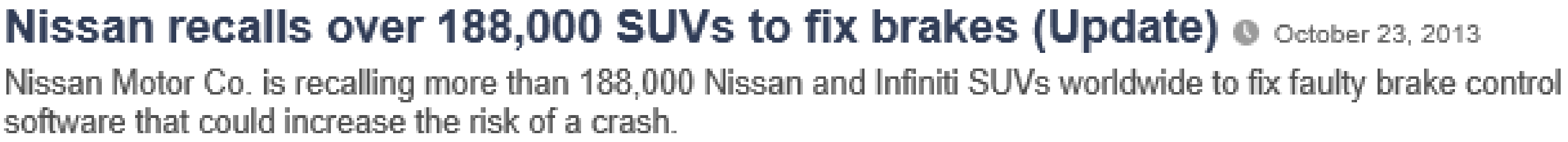
- Hardware faults
- Configuration faults
- Operator faults

Fault tolerance (e.g. redundancy)

Software is the cause of problems

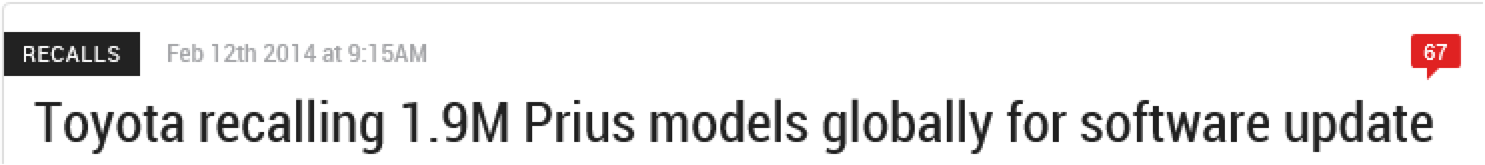
„Defibtech issues a worldwide recall of two of its defibrillator products due to faulty self-test software that may clear a previously detected low battery condition.” (February 2007)

„Cricket Communications recalls about 285,000 of its cell phones due to a software glitch that causes audio problems when a caller connects to an emergency 911 call. (May 2008)”



Nissan recalls over 188,000 SUVs to fix brakes (Update) October 23, 2013

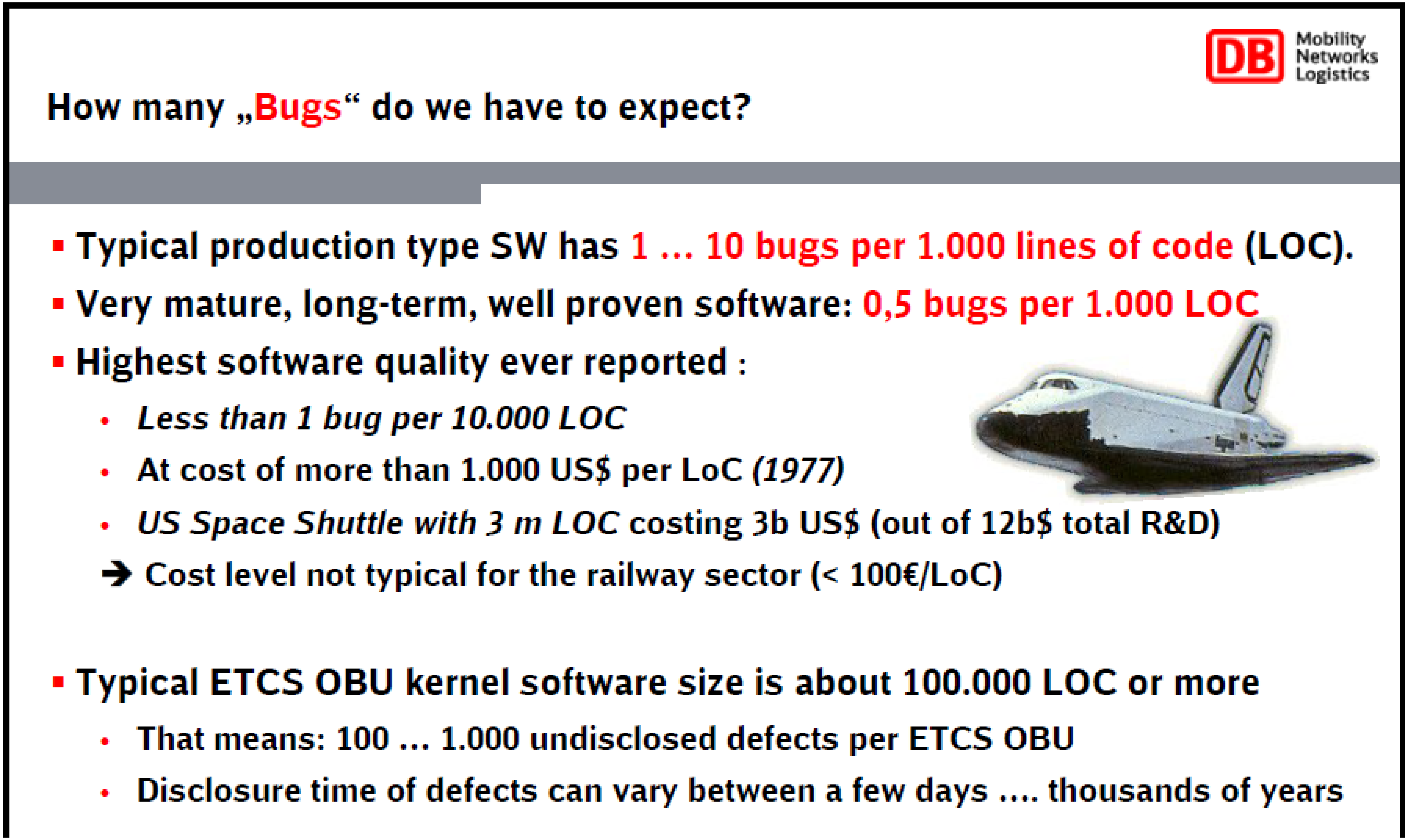
Nissan Motor Co. is recalling more than 188,000 Nissan and Infiniti SUVs worldwide to fix faulty brake control software that could increase the risk of a crash.



RECALLS Feb 12th 2014 at 9:15AM 67

Toyota recalling 1.9M Prius models globally for software update

How many bugs do we have to expect?



DB Mobility Networks Logistics

How many „Bugs“ do we have to expect?

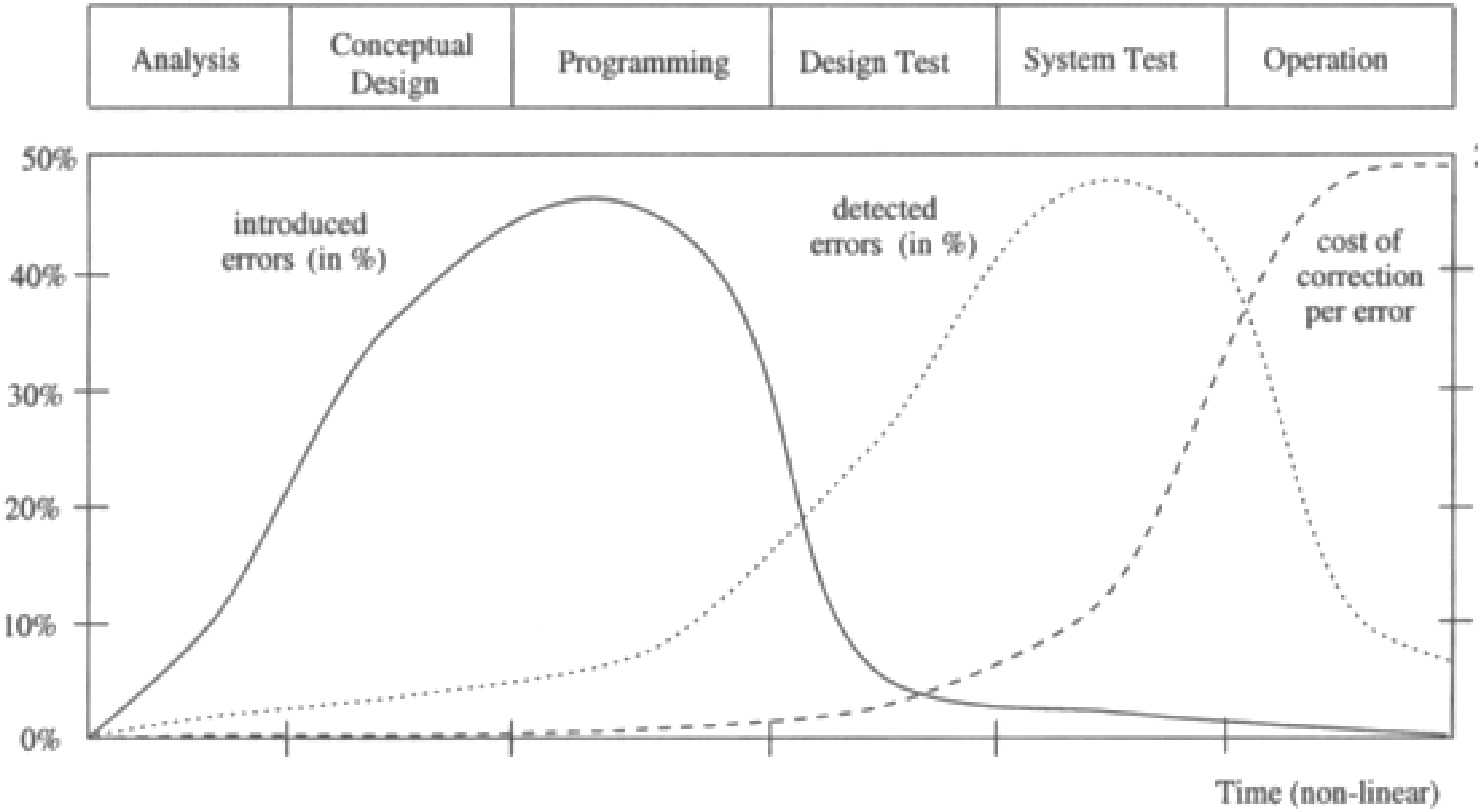
- Typical production type SW has 1 … 10 bugs per 1.000 lines of code (LOC).
- Very mature, long-term, well proven software: 0,5 bugs per 1.000 LOC
- Highest software quality ever reported :
  - *Less than 1 bug per 10.000 LOC*
  - At cost of more than 1.000 US$ per LoC *(1977)*
  - *US Space Shuttle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

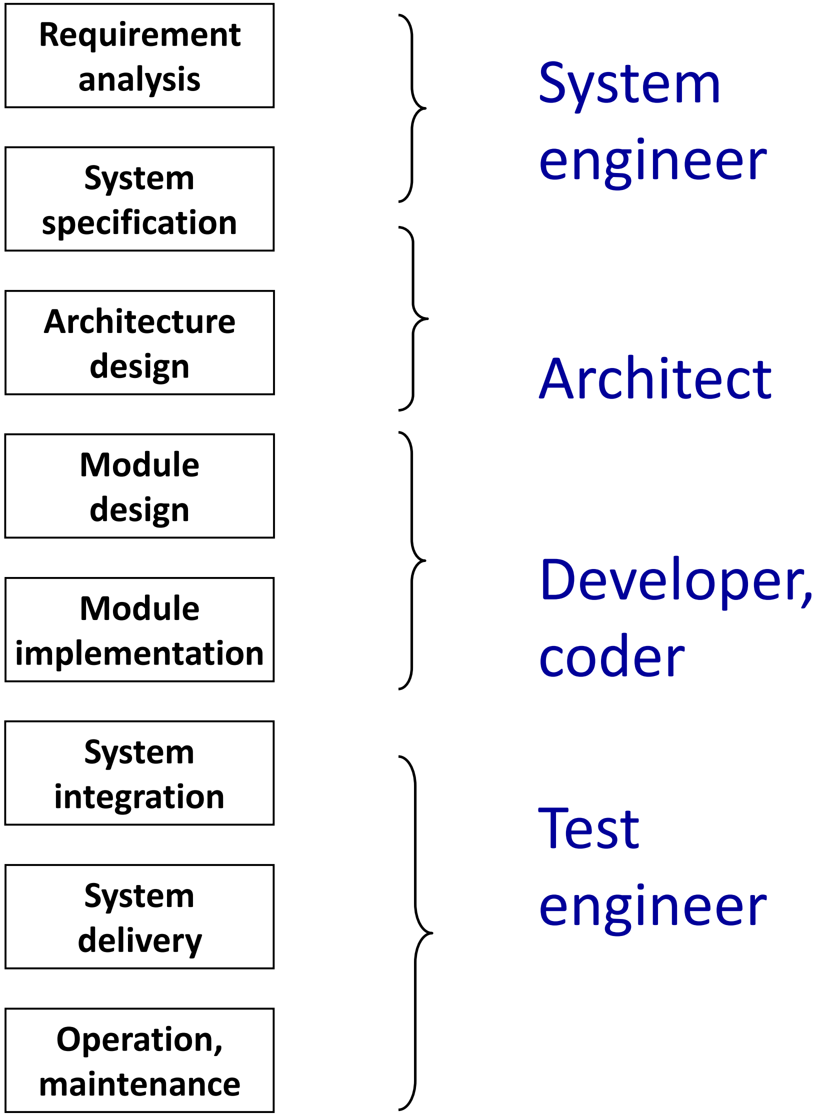
Verification	**Validation**
„Am I building the system right?”	„Am I building the right system?”
Check consistency of development phases	Check the result of the development
Conformance of designs/models and their specification	Conformance of the finished system and the user requirements
Objective; can 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)

OVERVIEW OF V&V TECHNIQUES

Learning outcomes

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

Typical steps in development lifecycle



- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- System integration
- System delivery
- Operation, maintenance

System engineer

Architect

Developer, coder

Test engineer

Schedule, sequencing depends on lifecycle model!

Requirement analysis

Requirement analysis

System specification

Architecture design

Module design

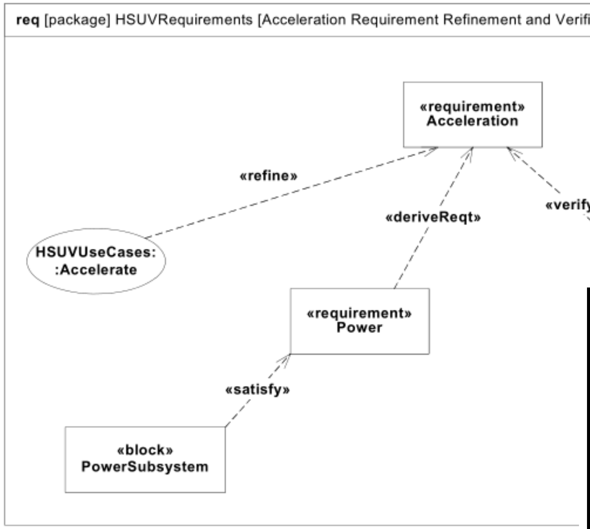
Module implementation

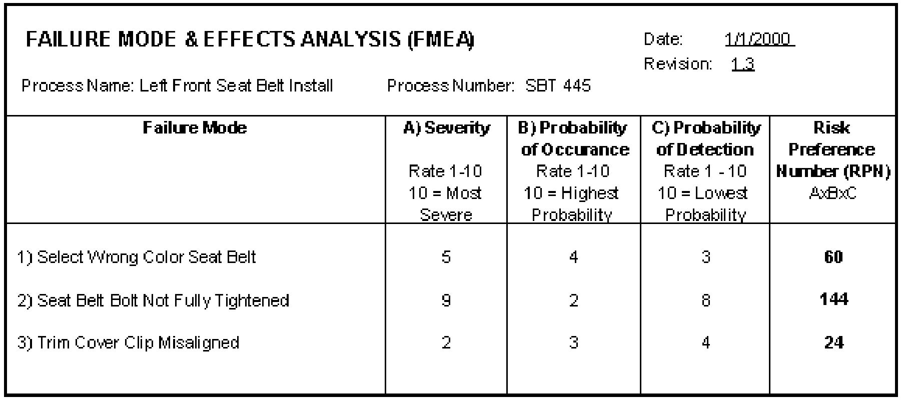
System integration

System delivery

Operation, maintenance

Task	V&V criteria	V&V technique
Defining functions, actors, use cases	- Risks - Criticality	- Checklists - Failure mode and effects analysis


req [package] HSUVRequirements [Acceleration Requirement Refinement and Verification]
«requirement» Acceleration
«refine»
«deriveReqt»
«verify»
HSUVUseCases: :Accelerate
«requirement» Power
«satisfy»
«block» PowerSubsystem


FAILURE MODE & EFFECTS ANALYSIS (FMEA)

Date: 1/1/2000

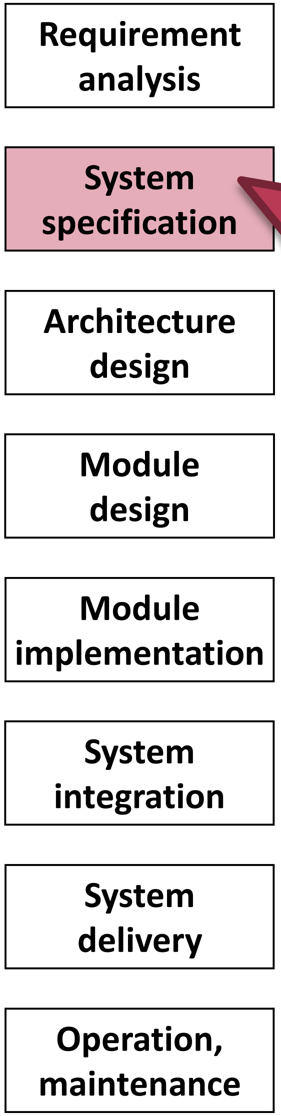
Revision: 1.3

Process Name: Left Front Seat Belt Install

Process Number: SBT 445

Failure Mode	A) Severity Rate 1-10 10 = Most Severe	B) Probability of Occurance Rate 1-10 10 = Highest Probability	C) Probability of Detection Rate 1 - 10 10 = Lowest Probability	Risk Preference Number (RPN) AxBxC
1) Select Wrong Color Seat Belt	5	4	3	60
2) Seat Belt Bolt Not Fully Tightened	9	2	8	144
3) Trim Cover Clip Misaligned	2	3	4	24

System specification



Requirement analysis

System specification

Architecture design

Module design

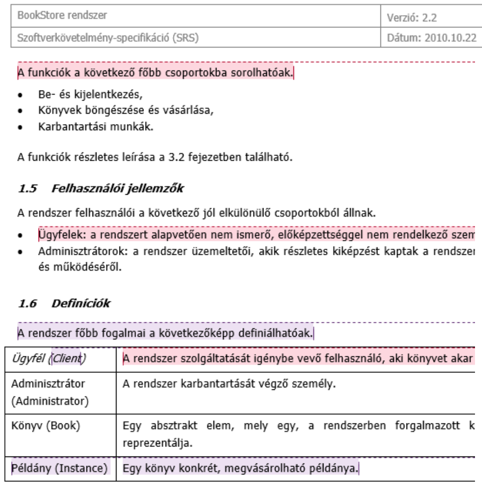
Module implementation

System integration

System delivery

Operation, maintenance

Task	V&V criteria	V&V technique
Defining functional and non-functional requirements	- Completeness - Unambiguity - Verifiability - Feasibility	- Reviews - Static analysis - Simulation



BookStore rendszer	Verzió: 2.2
Szoftverkövetelmény-specifikáció (SRS)	Dátum: 2010.10.22

A funkciók a következő főbb csoportokba sorolhatóak.

- Be- és kijelentkezés,
- Könyvek böngészése és vásárlása,
- Karbantartási munkák.

A funkciók részletes leírása a 3.2 fejezetben található.

1.5 Felhasználói jellemzők

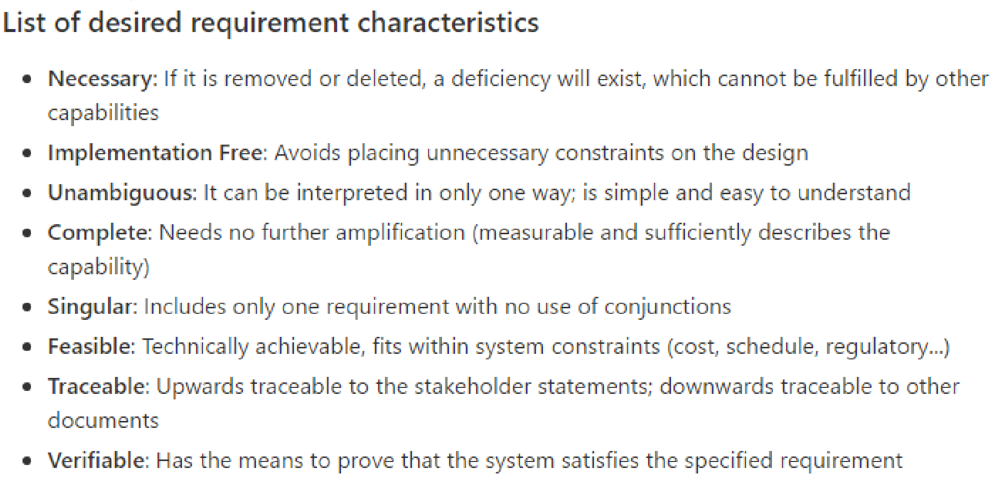
A rendszer felhasználói a következő jól elkülönülő csoportokból állnak.

- Ügyfelek: a rendszert alapvetően nem ismerő, előképzettséggel nem rendelkező szem
- Adminisztrátorok: a rendszer üzemeltetői, akik részletes kiképzést kaptak a rendszer és működéséről.

1.6 Definíciók

A rendszer főbb fogalmai a következőképp definiálhatóak.

Ügyfél (Client)	A rendszer szolgáltatását igénybe vevő felhasználó, aki könyvet akar
Adminisztrátor (Administrator)	A rendszer karbantartását végző személy.
Könyv (Book)	Egy absztrakt elem, mely egy, a rendszerben forgalmazott k reprezentálja.
Példány (Instance)	Egy könyv konkrét, megvásárolható példánya.



List of desired requirement characteristics

- **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
- **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

Architecture design

Requirement analysis

System specification

Architecture design

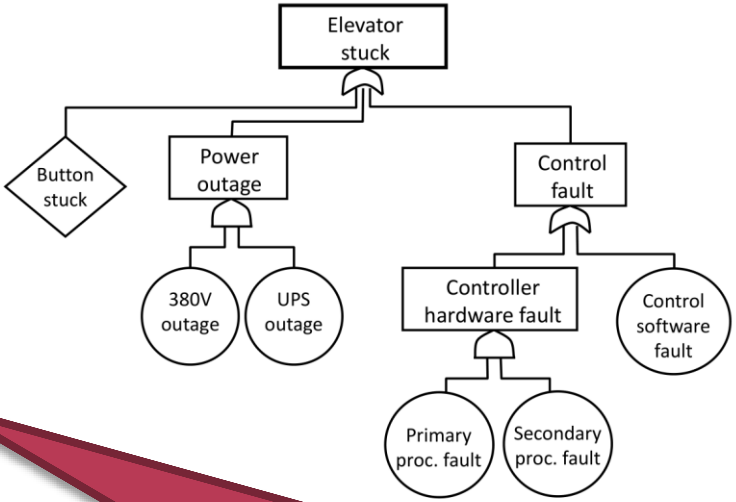
Module design

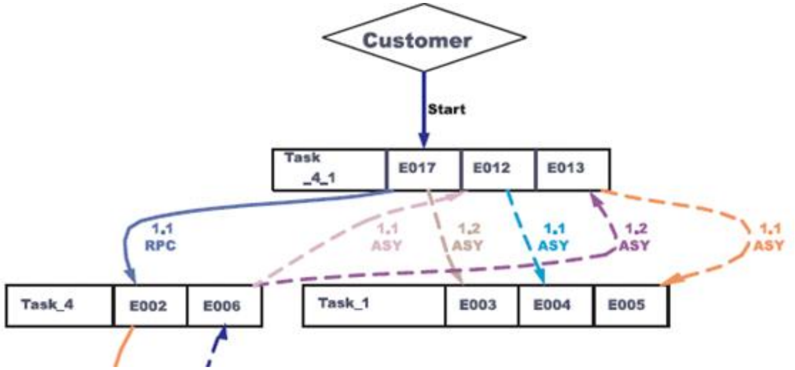
Module implementation

System integration

System delivery

Operation, maintenance

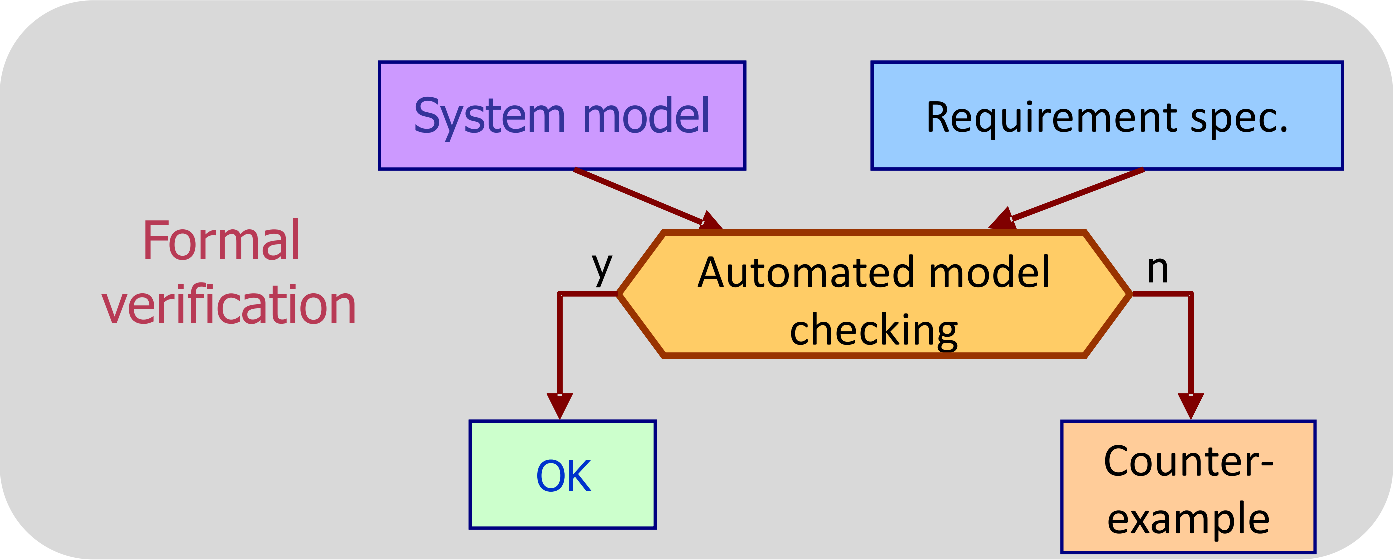




Task	V&V criteria	V&V technique
- Decomposing modules - HW-SW co-design - Designing communication	- Function coverage - Conformance of interfaces - Non-functional properties	- Static analysis - Simulation - Performance, dependability, security analysis

Module design (detailed design)

- Requirement analysis
- System specification
- Architecture design
- **Module design**
- Module implementation
- System integration
- System delivery
- Operation, maintenance



Formal verification

System model

Requirement spec.

Automated model checking

y

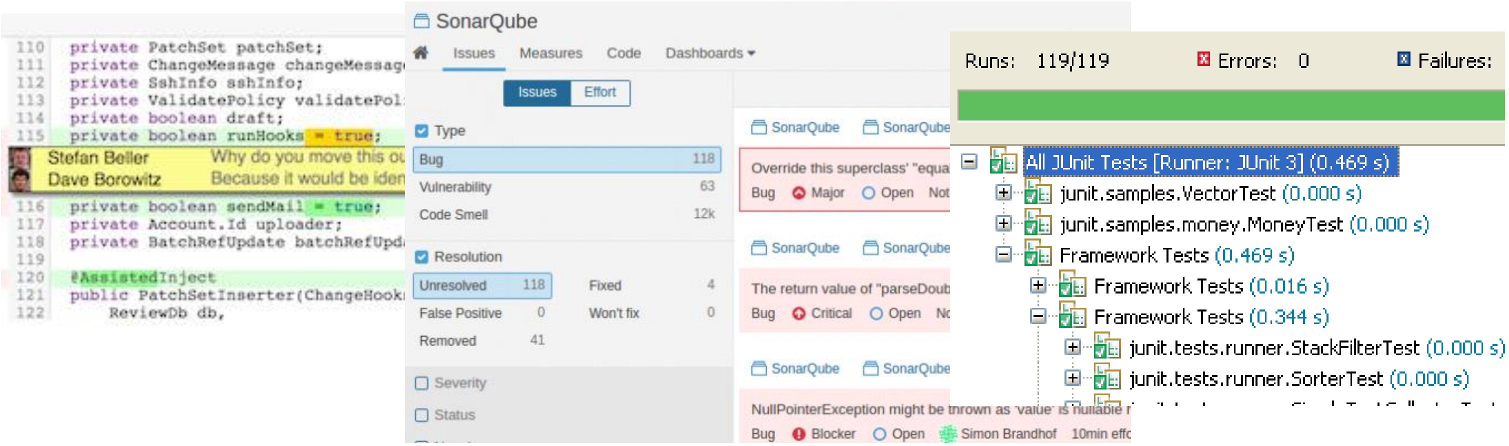
n

OK

Counter-example

Task	V&V criteria	V&V technique
- Designing detailed behavior (data structures, algorithms)	- Correctness of critical internal algorithms and protocols	- Static analysis - Simulation - Formal verification - Rapid prototyping

Module implementation



```
private PatchSet patchSet;
private ChangeMessage changeMessag
private SshInfo sshInfo;
private ValidatePolicy validatePol
private boolean draft;
private boolean runHooks = true;
```

Stefan Beller Why do you move this ou

Dave Borowitz Because it would be iden

```
private boolean sendMail = true;
private Account.Id uploader;
private BatchRefUpdate batchRefUpd

@AssistedInject
public PatchSetInserter(ChangeHook
    ReviewDb db,
```

SonarQube

☑ Type

Bug	118
Vulnerability	63
Code Smell	12k

☑ Resolution

Unresolved	118	Fixed	4
False Positive	0	Won't fix	0
Removed	41		

□ Severity

□ Status

SonarQube SonarQube

Override this superclass' "equa

Bug Major ○ Open Not

SonarQube SonarQube

The return value of "parseDoub

Bug Critical ○ Open No

SonarQube SonarQube

NullPointerException might be thrown as 'value' is nullable r

Bug Blocker ○ Open Simon Brandhof 10min effo

Runs: 119/119 Errors: 0 Failures:

```
All JUnit Tests [Runner: JUnit 3] (0.469 s)
  junit.samples.VectorTest (0.000 s)
  junit.samples.money.MoneyTest (0.000 s)
  Framework Tests (0.469 s)
    Framework Tests (0.016 s)
    Framework Tests (0.344 s)
      junit.tests.runner.StackFilterTest (0.000 s)
      junit.tests.runner.SorterTest (0.000 s)
```

Requirement analysis

System specification

Architecture design

Module design

Module implementation

System integration

System delivery

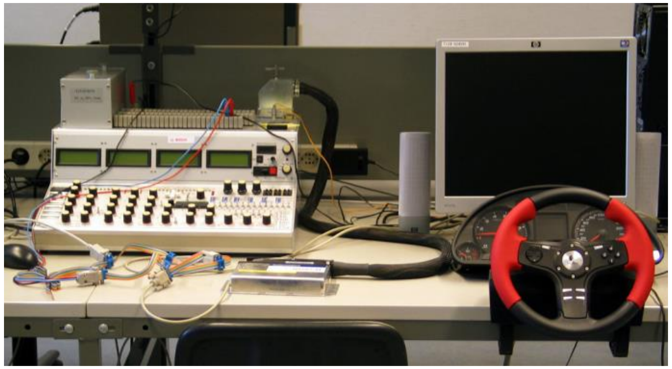
Operation, maintenance

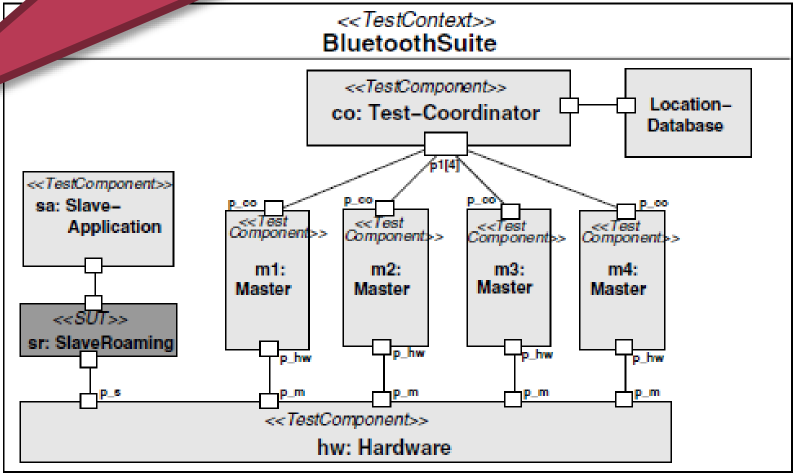
Task	V&V criteria	V&V technique
- Software implementation	Code is - Safe - Verifiable - Maintainable	- Coding conventions - Code reviews - Static code analysis
- Verifying module implementation	- Conformance to module designs	- Unit testing - Regression testing

System integration

- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- **System integration**
- System delivery
- Operation, maintenance

Task	V&V criteria	V&V technique
- Integrating modules - Integrating SW with HW	- Conformance of integrated behavior - Verifying communication	- Integration testing (incremental)

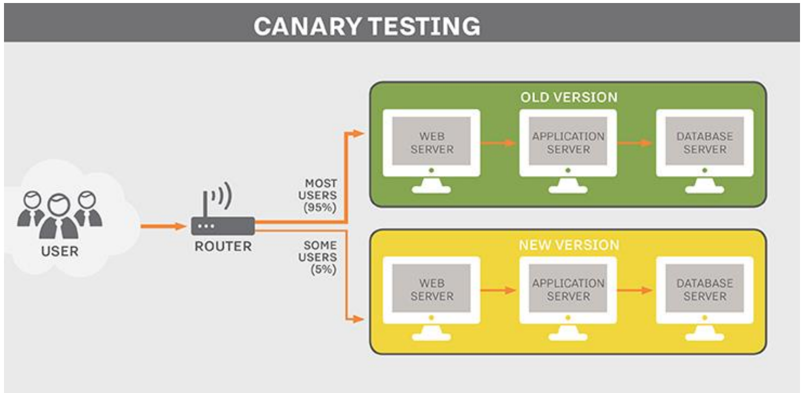




System delivery and deployment

- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- System integration
- **System delivery**
- Operation, maintenance

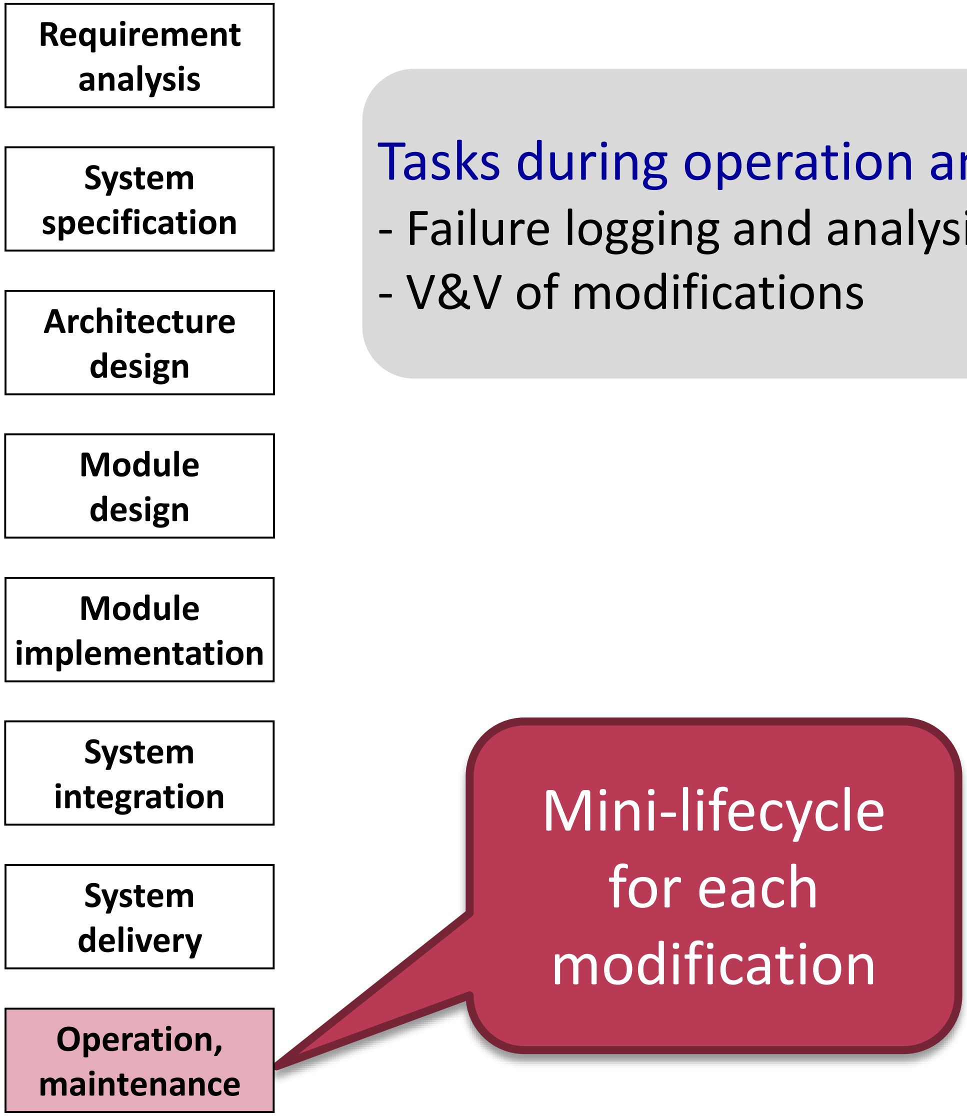
Source: Video and radar test (Bosch)



Source: TechTarget

Task	V&V criteria	V&V technique
- Assembling complete system	- Conformance to system specification	- System testing - Measurements, monitoring
- Fulfilling user expectations	- Conformance to requirements and expectations	- Validation testing - Acceptance testing - Alfa/beta testing

Operation and maintenance



Requirement analysis

System specification

Architecture design

Module design

Module implementation

System integration

System delivery

Operation, maintenance

Mini-lifecycle for each modification

Tasks during operation and maintenance:

- Failure logging and analysis (for failure prediction)
- V&V of modifications

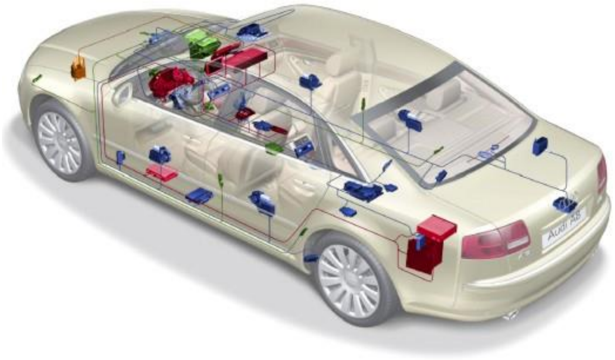
V&V TECHNIQUES IN CRITICAL SYSTEMS

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

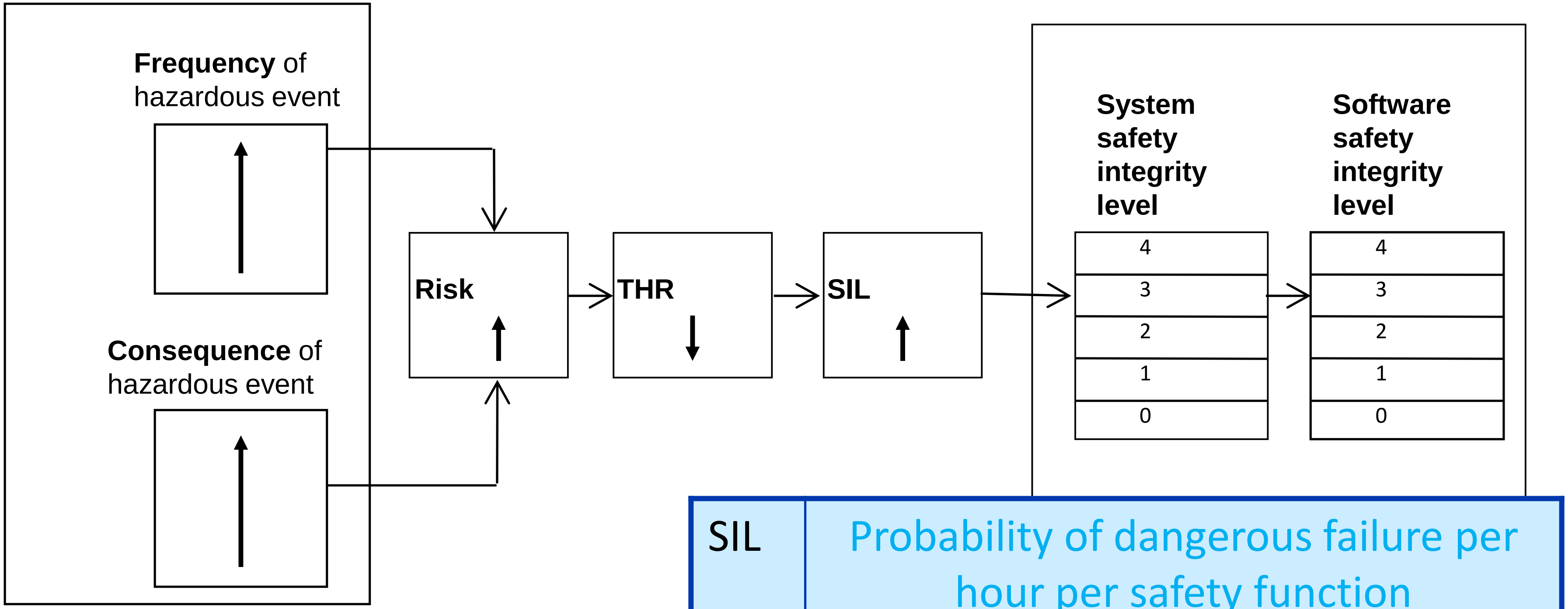
- Basis of certification: Standards
  - IEC 61508: Generic standard (for electrical, electronic or programmable electronic systems)
  - DO178B/C: Software in airborne systems
  - EN50128: Railway (software)
  - ISO26262: Automotive

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


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

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	$10^{-9} \leq THR < 10^{-8}$

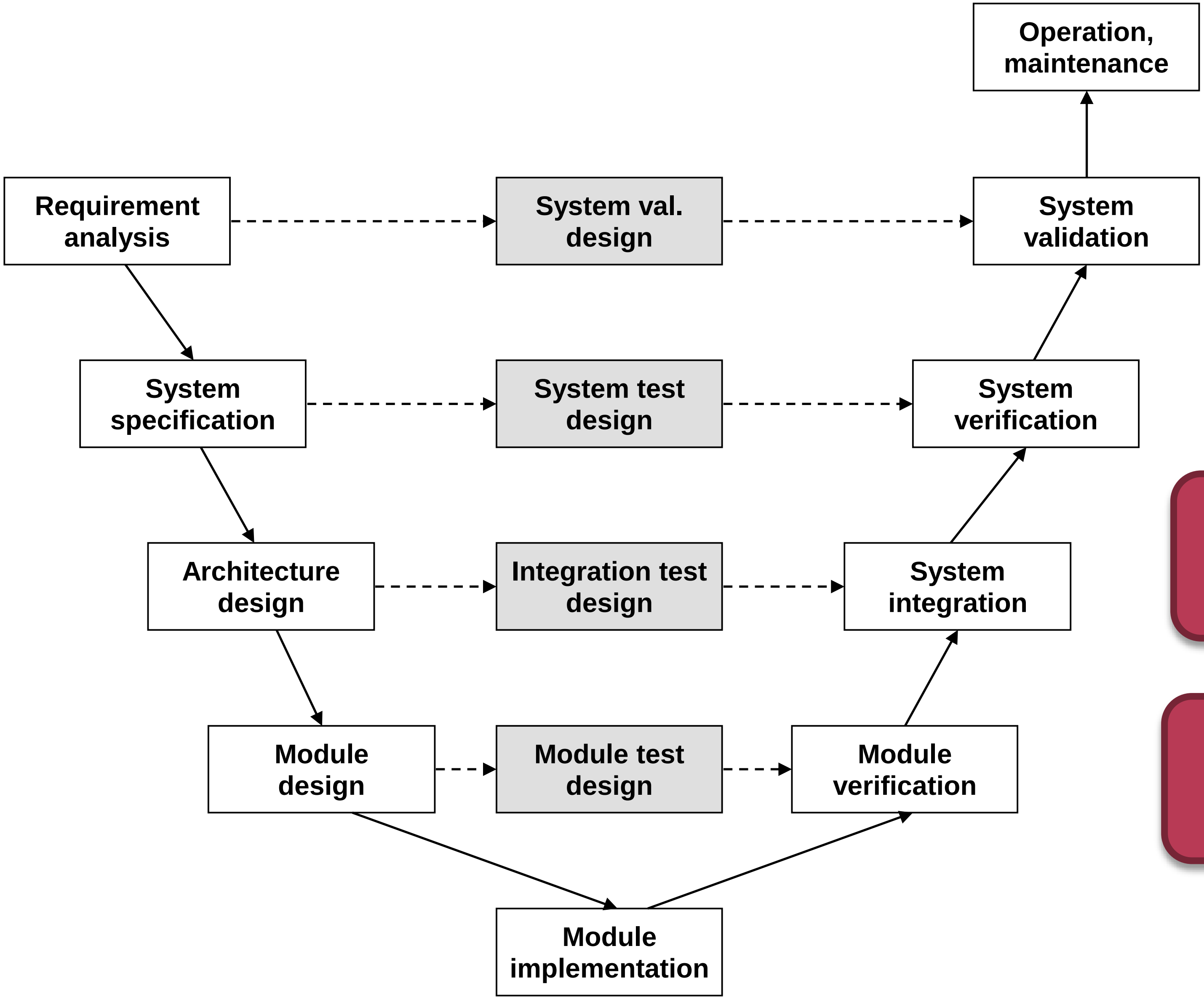
15 years lifetime:
1 failure in case
of 750 equipment

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

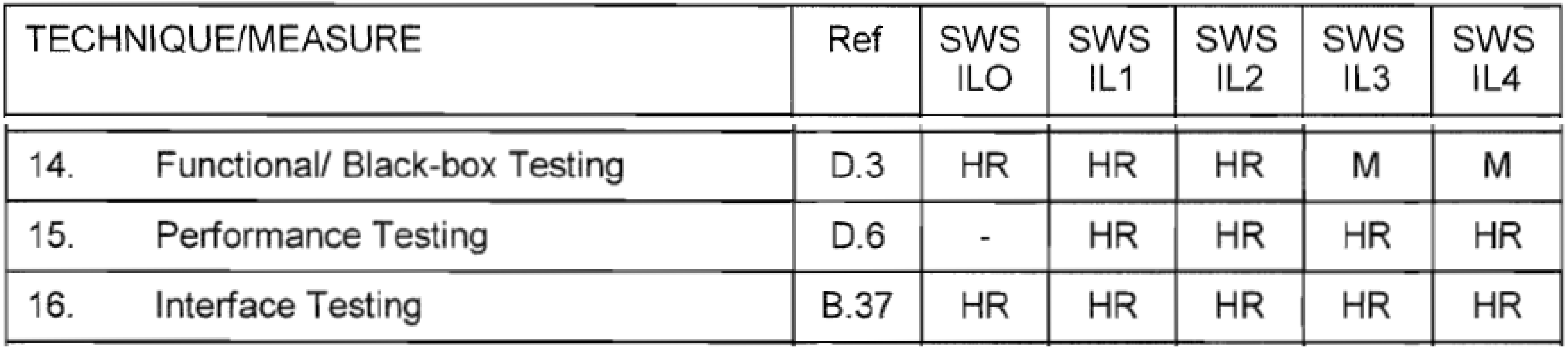
Example: Process (V model)



Well-defined phases

Verification of each step

Example: Techniques (EN 50128)

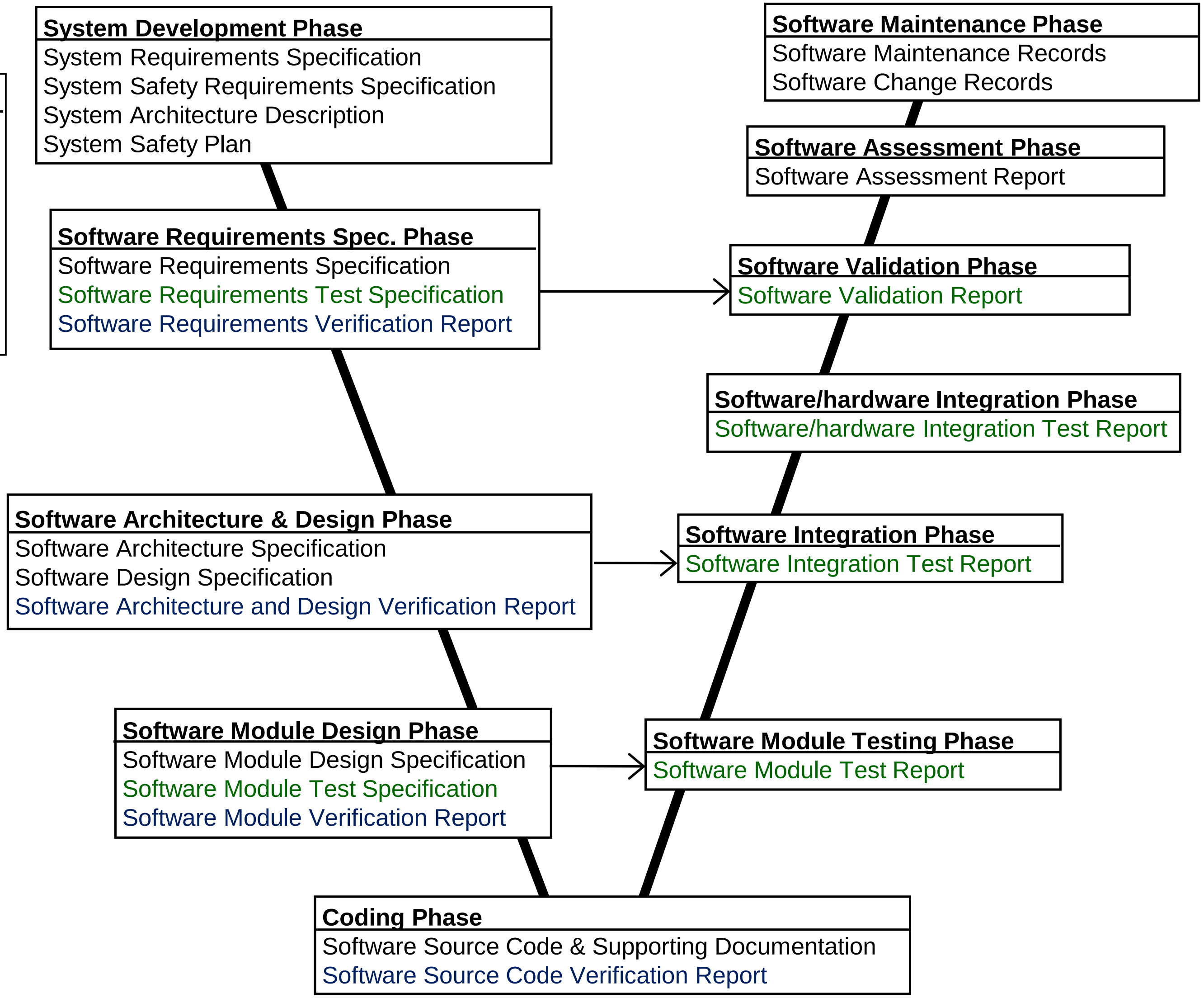


TECHNIQUE/MEASURE		Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
14.	Functional/ Black-box Testing	D.3	HR	HR	HR	M	M
15.	Performance Testing	D.6	-	HR	HR	HR	HR
16.	Interface Testing	B.37	HR	HR	HR	HR	HR

- 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)

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

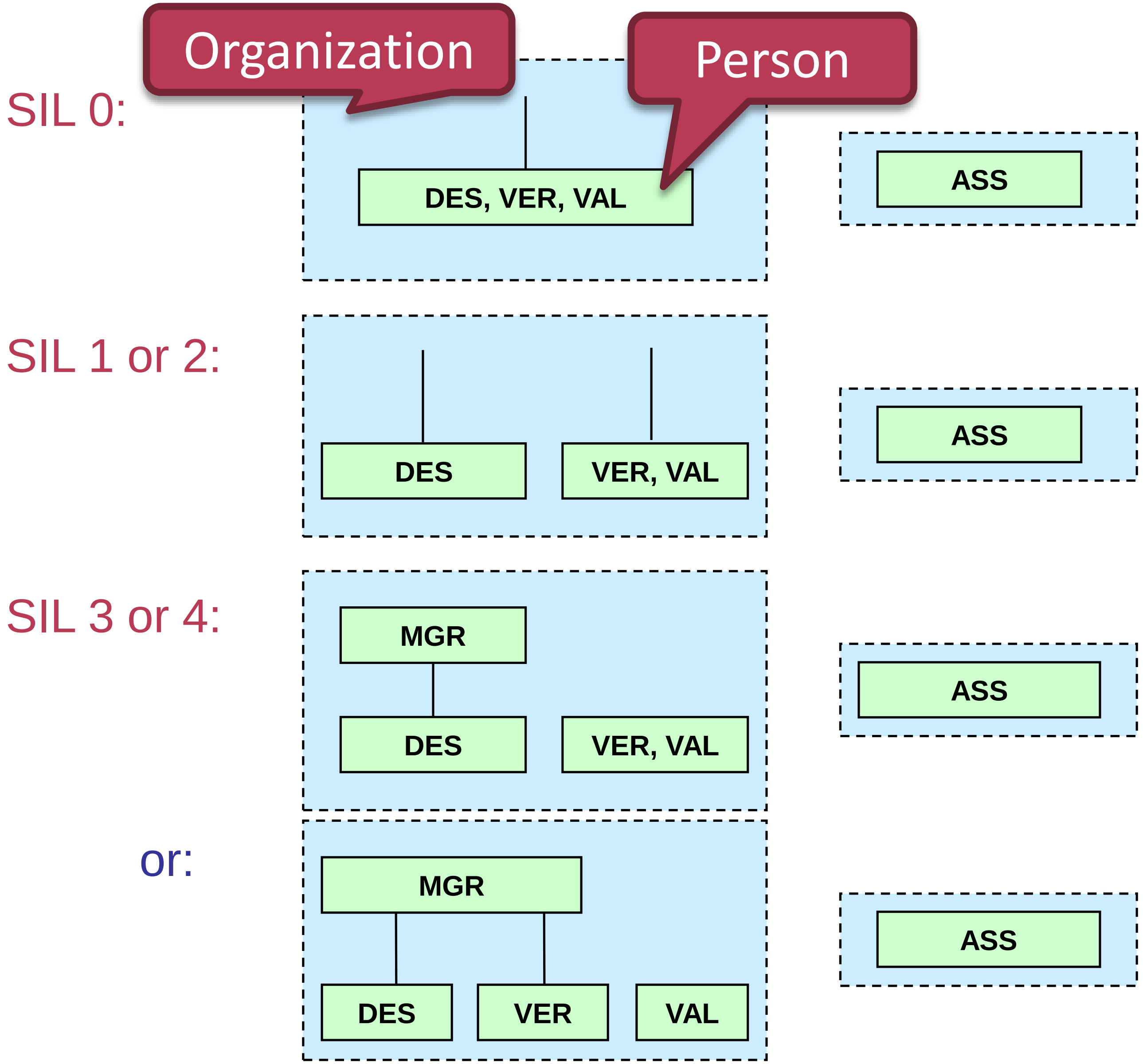
Software Assessment Phase
Software Assessment Report

Software Maintenance Phase
Software Maintenance Records
Software Change Records

30 documents in a systematic structure

- Specification
- Design
- Verification

Example: 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

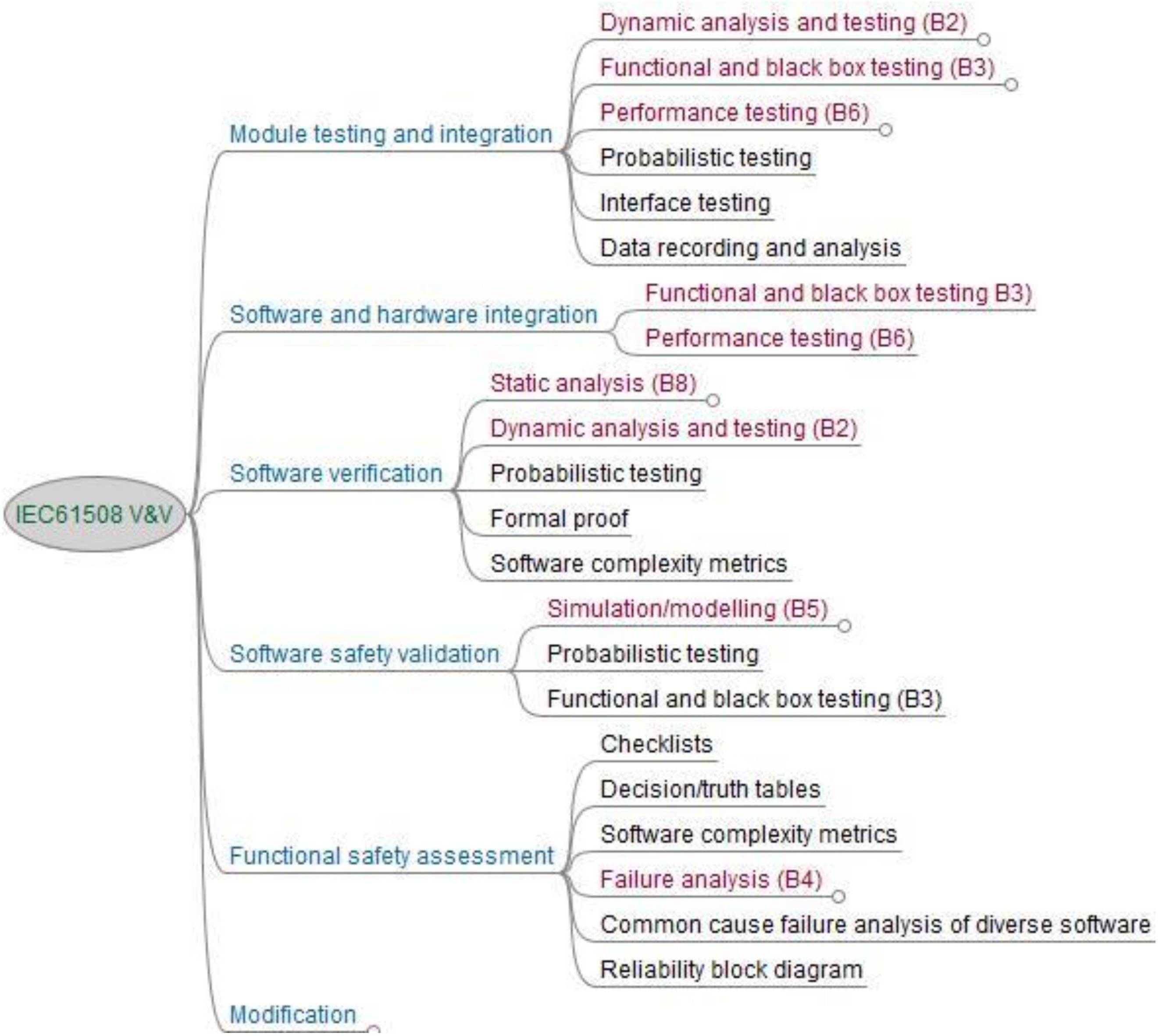
ASS

DES: Designer (analyst, architect, coder, unit tester)
VER: Verifier
VAL: Validator
ASS: Assessor
MAN: Project manager

BACKGROUND MATERIAL

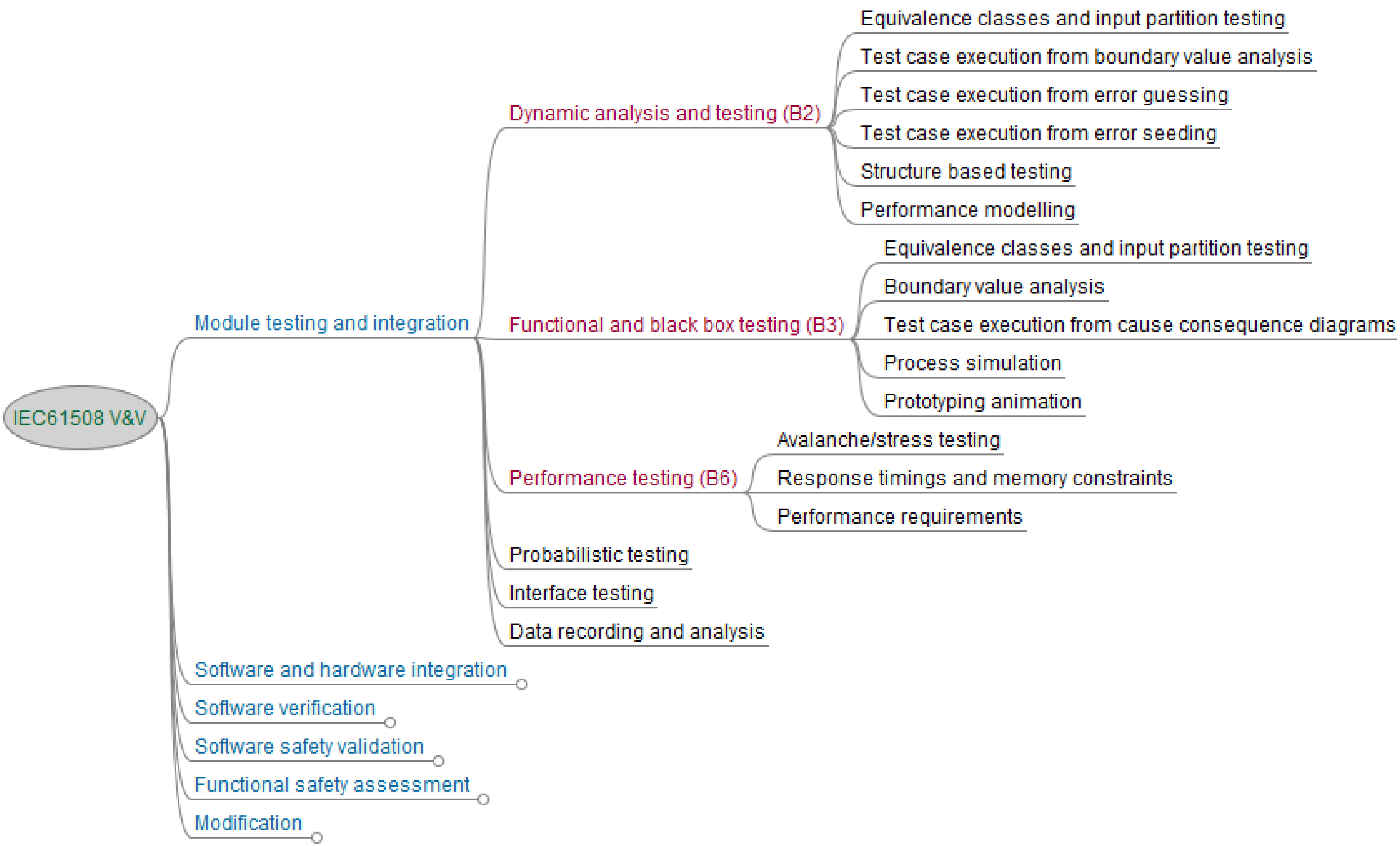
(For reference only, recommended to come back at the end of the course to see how many techniques are familiar)

IEC 61508 V&V methods



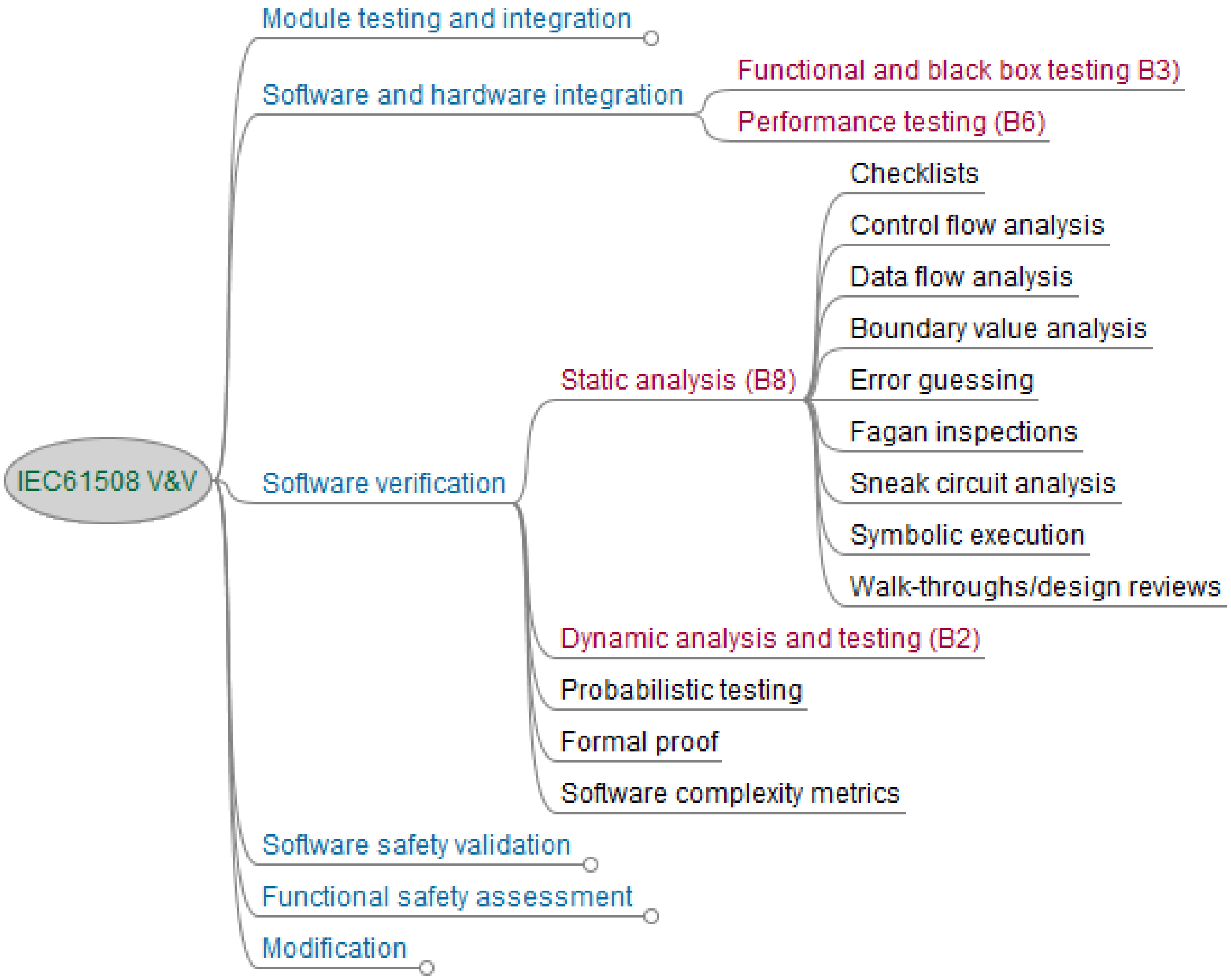
IEC61508 V&V
Module testing and integration
Dynamic analysis and testing (B2)
Functional and black box testing (B3)
Performance testing (B6)
Probabilistic testing
Interface testing
Data recording and analysis
Software and hardware integration
Functional and black box testing B3)
Performance testing (B6)
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

IEC 61508 V&V methods – Testing



- 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

IEC 61508 V&V methods – Static analysis



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