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?

<table>
<thead>
<tr>
<th>Role</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Engineer</td>
<td>- Requirements, verifying specification</td>
</tr>
<tr>
<td>Architect, Designer</td>
<td>- Modeling and verifying designs</td>
</tr>
<tr>
<td>Developer, Coder</td>
<td>- Verifying source code, unit testing</td>
</tr>
<tr>
<td>Test Designer</td>
<td>- Test processes and techniques</td>
</tr>
<tr>
<td>Test Engineer</td>
<td>- Test automation, integration and system tests</td>
</tr>
<tr>
<td>Safety Engineer</td>
<td>- Certification, development standards</td>
</tr>
</tbody>
</table>
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)
Testers help make the system better
Passion for quality

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...

**Critical systems**
- Safety
- Process, standards
- Documentation

**Enterprise, web**
- Agile, Lean
- ISTQB
Useful resources (download now!)

- **IEEE standards**
  - [24765-2010](#) Systems and SW engineering – Vocabulary
  - [29148-2011](#) Requirements engineering
  - [29119](#) Software testing
    - Part 1 Concepts and definitions
    - Part 2 Test processes
    - Part 3 Test documentation

- **International Software Testing Qualifications Board (ISTQB)**
  - **Foundation Level Syllabus** (2011)
  - **Glossary of Testing Terms**

- **Hungarian Testing Board (HTB)**
  - **Glossary** / Kifejezésgyűjtemény (magyar fordítás)
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.

Toyota recalling 1.9M Prius models globally for software update
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

Distribution and cost of bugs

Early V&V reduces cost!
## V&V: Verification and Validation

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

Schedule, sequencing depends on lifecycle model!

- System engineer
  - Architect
    - Developer, coder
    - Test engineer
## Requirement analysis

<table>
<thead>
<tr>
<th>Task</th>
<th>V&amp;V criteria</th>
<th>V&amp;V technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining functions, actors, use cases</td>
<td>- Risks</td>
<td>- Checklists</td>
</tr>
<tr>
<td></td>
<td>- Criticality</td>
<td>- Failure mode and effects analysis</td>
</tr>
</tbody>
</table>

### Failure Mode & Effects Analysis (FMEA)

<table>
<thead>
<tr>
<th>Failure Mode</th>
<th>A) Severity</th>
<th>B) Probability of Occurrence</th>
<th>C) Probability of Detection</th>
<th>Risk Preference Number (RPN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Select Wrong Color Seat Belt</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>68</td>
</tr>
<tr>
<td>2) Seat Belt Belt Not Fully Tightened</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>111</td>
</tr>
<tr>
<td>3) Thin Cover Clip Misaligned</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>24</td>
</tr>
</tbody>
</table>

**Process Name:** Left Front Seat Belt Install  
**Process Number:** SET 445  
**Date:** 10/2000  
**Revision:** 1.3
System specification

<table>
<thead>
<tr>
<th>Task</th>
<th>V&amp;V criteria</th>
<th>V&amp;V technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining functional and non-functional requirements</td>
<td>- Completeness</td>
<td>- Reviews</td>
</tr>
<tr>
<td></td>
<td>- Unambiguity</td>
<td>- Static analysis</td>
</tr>
<tr>
<td></td>
<td>- Verifiability</td>
<td>- Simulation</td>
</tr>
<tr>
<td></td>
<td>- Feasibility</td>
<td></td>
</tr>
</tbody>
</table>

Reality

Analysis

Design space

Modeling
- structuring
- abstraction

Designing
- decomposition

Implementation space

System specification

Requirement analysis

Architecture design

Module design

Module implementation

System integration

System delivery

Operation, maintenance

Defining functional and non-functional requirements

Completeness

Unambiguity

Verifiability

Feasibility

Reviews

Static analysis

Simulation

Completeness

Unambiguity

Verifiability

Feasibility

Reviews

Static analysis

Simulation
Architecture design

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

**Task**
- Decomposing modules
- HW-SW co-design
- Designing communication

**V&V criteria**
- Function coverage
- Conformance of interfaces
- Non-functional properties

**V&V technique**
- Static analysis
- Simulation
- Performance, dependability, security analysis

Design space

- Structure design space and mapping

Analysis

- Mapping (automated)
Module design (detailed design)

<table>
<thead>
<tr>
<th>Task</th>
<th>V&amp;V criteria</th>
<th>V&amp;V technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Designing detailed behavior</td>
<td>- Correctness of critical internal algorithms and</td>
<td>- Static analysis</td>
</tr>
<tr>
<td>(data structures, algorithms)</td>
<td>protocols</td>
<td>- Simulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Formal verification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rapid prototyping</td>
</tr>
</tbody>
</table>
Module implementation

<table>
<thead>
<tr>
<th>Requirement analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>System specification</td>
</tr>
<tr>
<td>Architecture design</td>
</tr>
<tr>
<td>Module design</td>
</tr>
<tr>
<td>Module implementation</td>
</tr>
<tr>
<td>System integration</td>
</tr>
<tr>
<td>System delivery</td>
</tr>
<tr>
<td>Operation, maintenance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>V&amp;V criteria</th>
<th>V&amp;V technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Software implementation</td>
<td>Code is</td>
<td>- Coding conventions</td>
</tr>
<tr>
<td></td>
<td>- Safe</td>
<td>- Code reviews</td>
</tr>
<tr>
<td></td>
<td>- Verifiable</td>
<td>- Static code analysis</td>
</tr>
<tr>
<td></td>
<td>- Maintainable</td>
<td></td>
</tr>
<tr>
<td>- Verifying module implementation</td>
<td>- Conformance to module designs</td>
<td>- Unit testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Regression testing</td>
</tr>
</tbody>
</table>
# System integration

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

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

![System Under Test (SUT)](image)
System delivery and deployment

<table>
<thead>
<tr>
<th>Requirement analysis</th>
<th>System specification</th>
<th>Architecture design</th>
<th>Module design</th>
<th>Module implementation</th>
<th>System integration</th>
<th>System delivery</th>
<th>Operation, maintenance</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>V&amp;V criteria</th>
<th>V&amp;V technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Assembling complete system</td>
<td>- Conformance to system specification</td>
<td>- System testing</td>
</tr>
<tr>
<td>- Fulfilling user expectations</td>
<td>- Conformance to requirements and expectations</td>
<td>- Validation testing, Acceptance testing, Alfa/beta testing</td>
</tr>
</tbody>
</table>
Tasks during operation and maintenance:
- Failure logging and analysis (for failure prediction)
- V&V of modifications

Mini-lifecycle for each modification
Learning outcomes

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

Software safety integrity level

<table>
<thead>
<tr>
<th>SIL</th>
<th>Probability of dangerous failure per hour per safety function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$10^{-6} \leq \text{THR} &lt; 10^{-5}$</td>
</tr>
<tr>
<td>2</td>
<td>$10^{-7} \leq \text{THR} &lt; 10^{-6}$</td>
</tr>
<tr>
<td>3</td>
<td>$10^{-8} \leq \text{THR} &lt; 10^{-7}$</td>
</tr>
<tr>
<td>4</td>
<td>$10^{-9} \leq \text{THR} &lt; 10^{-8}$</td>
</tr>
</tbody>
</table>

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)

- Requirement analysis
- System specification
- Architecture design
- Module design
- Module implementation
- System design
- Integration test design
- Module test design
- Module verification
- System test design
- System verification
- System integration
- System val. design
- System validation
- Operation, maintenance

Well-defined phases
Verification of each step
<table>
<thead>
<tr>
<th>TECHNIQUE/MEASURE</th>
<th>Ref</th>
<th>SWS ILO</th>
<th>SWS IL1</th>
<th>SWS IL2</th>
<th>SWS IL3</th>
<th>SWS IL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>14. Functional/ Black-box Testing</td>
<td>D.3</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>15. Performance Testing</td>
<td>D.6</td>
<td>-</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>16. Interface Testing</td>
<td>B.37</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
</tr>
</tbody>
</table>

- **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

<table>
<thead>
<tr>
<th>Software Planning Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Development Plan</td>
</tr>
<tr>
<td>Software Quality Assurance Plan</td>
</tr>
<tr>
<td>Software Configuration Management Plan</td>
</tr>
<tr>
<td>Software Verification Plan</td>
</tr>
<tr>
<td>Software Integration Test Plan</td>
</tr>
<tr>
<td>Software Validation Plan</td>
</tr>
<tr>
<td>Software Maintenance Plan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Development Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Requirements Specification</td>
</tr>
<tr>
<td>System Safety Requirements Specification</td>
</tr>
<tr>
<td>System Architecture Description</td>
</tr>
<tr>
<td>System Safety Plan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Requirements Spec. Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Requirements Specification</td>
</tr>
<tr>
<td>Software Requirements Test Specification</td>
</tr>
<tr>
<td>Software Requirements Verification Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Architecture &amp; Design Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Architecture Specification</td>
</tr>
<tr>
<td>Software Design Specification</td>
</tr>
<tr>
<td>Software Architecture and Design Verification Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Module Design Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Module Design Specification</td>
</tr>
<tr>
<td>Software Module Test Specification</td>
</tr>
<tr>
<td>Software Module Verification Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Module Testing Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Module Test Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Validation Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Validation Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software/hardware Integration Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software/hardware Integration Test Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Integration Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Integration Test Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Maintenance Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Maintenance Records</td>
</tr>
<tr>
<td>Software Change Records</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Assessment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Assessment Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Maintenance Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Maintenance Records</td>
</tr>
<tr>
<td>Software Change Records</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coding Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Source Code &amp; Supporting Documentation</td>
</tr>
<tr>
<td>Software Source Code Verification Report</td>
</tr>
</tbody>
</table>
Example: Responsibilities (EN 50128)

- **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 – Testing

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
  - Equivalence classes and input partition testing
  - Boundary value analysis
  - Test case execution from cause consequence diagrams
  - Process simulation
  - Prototyping animation

Functional and black box testing (B3)
- Avalanche/stress testing
- Response timings and memory constraints
- Performance requirements
- Performance testing (B6)
  - 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

Module testing and integration
  - Functional and black box testing (B3)
  - Performance testing (B6)

Software and hardware integration
  - Checklists
    - Control flow analysis
    - Data flow analysis
    - Boundary value analysis
    - Error guessing
    - Fagan inspections
    - Sneak circuit analysis
    - Symbolic execution
    - Walk-throughs/design reviews

Static analysis (B8)
  - Dynamic analysis and testing (B2)
    - Probabilistic testing
    - Formal proof
    - Symbolic execution metrics

Software verification
  - Software safety validation
  - Functional safety assessment
  - Modification