Chapter 7: Verification, Validation, Testing

Anton Setzer
Dept. of Computer Science, Swansea University

http://www.cs.swan.ac.uk/~csetzer/lectures/critsys/14/index.html

November 23, 2014
7 (a) Basic Notions

7 (b) Dynamic testing

7 (c) Static Analysis

7 (d) Modelling
7 (a) Basic Notions

7 (b) Dynamic testing

7 (c) Static Analysis

7 (d) Modelling
Verification is the process of determining whether the output of a life cycle phase fulfils the requirements specified in the previous phase.

So task is

- **not** to demonstrate that the **output** of a development phase is **actually correct**,
- but that the **output of a phase conforms to its input**.

Therefore mistakes in early phases of a project may **propagate** through later stages without detection.
Validation

- **Validation** is the process of confirming that the specification of a phase or of the complete system is appropriate and is consistent with the customer requirements.

- Validation
  - might be performed on individual phases,
  - but is usually performed on the complete system.
Testing is the process used to verify or validate a system or its components.

- Sometimes testing is used for testing, in which one executes the software in order to check whether it is performing as required.
- We use testing in the wider sense and dynamic testing for this more restricted version of testing.
Testing

- Testing is performed at various stages during the life cycle of a system.
- There are three main activities.
  - Module testing.
  - System integration testing.
  - System validation testing.
Main Testing Activities

- **Module testing** is the evaluation of small, simple functions of hardware or software.
  - Faults detected during module testing are usually easy to locate and to rectify.

- **System integration testing** investigates the characteristics of a collection of modules.
  - Usually investigates the correct interaction between modules.
  - Faults are more difficult to find and more expensive to rectify.
Main Testing Activities

- **System validation testing** tests whether the complete system satisfies the requirements.
  - Problems detected at this stage are usually due to weaknesses of customer requirements or the specification.
  - Problems detected are usually extremely costly to correct, since modifications have to propagate through the entire development process.
There are three main testing methods:

- **Dynamic testing.**
- **Static analysis.**
- **Modelling.**
Dynamic Testing

- **Dynamic testing** is the execution of a system or component in order to investigate its characteristics.
- The tests may be carried out
  - in the **system’s natural working environment**,  
  - or within **simulation of that environment**.
  - Often more cost effective.
Dynamic Testing and Simulation

- Dynamic testing might as well be carried out on one or a few system components by using **simulation**.
  - Especially of advantage if one simulates **hardware** which has **not** been **developed yet**.
  - Then simulation is **cost effective**, since it allows to compare various designs of the hardware involved.
  - However, simulation **never** provides **complete information** on the system behaviour, e.g.
    - **real-time operation**,  
    - **problems with timing**.
Static Analysis

- **Static analysis** is the investigation of the characteristics of a system or component without operating it.

- **Examples:**
  - Walkthroughs,
  - formal proofs,
  - data flow analysis.

- Automated software testing packages which carry out static analysis are called **static code analysis tools**.

- Many engineers mean by testing only dynamic testing, not static analysis.
Modelling means the mathematical representation of the behaviour of a system or component.

- Usually carried out at an early stage, in order to investigate the basic nature of the proposed system or its environment.
- **Animation** of a formal specification is an example of modelling.
Use of Testing Methods

- Typically, a software life cycle involves
  - dynamic testing,
  - static analysis,
  - some form of modelling.
Testing methods can be categorised by the information available when performing the work.

- **Black box testing** means the test engineer has no knowledge about the implementation of the system.
- **White box testing** means that the test engineer has access to the implementation of the system.
Black Box Testing

- In black box testing, the test engineer relies completely on the specification of the system.
- Therefore it is sometimes called requirements-based testing.
- May be applied to individual modules or (more common) to subsystems or the complete system.
- Is widely used for testing software tools like compilers.
Comparison

- **Advantage of Black Box Testing:**
  - Greatest level of independence between developer and evaluator.

- **Advantage of White Box Testing**
  - The test engineer can use information about the implementation in order to develop better tests.
Black/White-Box vs. Static/Dynamic

- **Dynamic testing** can be white-box and black-box testing.
- **Static analysis** is necessarily white-box testing.
- **Mathematical modelling** doesn’t use the system software and hardware, so categories white/black-box testing don’t apply to it.
Test planning is an essential part of the software life cycle. The next slide shows test planning within the V-model.
V-Model and Test Planning

Requirements analysis
Requirements documents → Test Planning
Specification
Specification → Test Planning
Top-Level Design
Design Specification → Test Planning
Detailed Design
Module Design → Test Planning
Construction/Coding → Module–Test
Modules

Service
Certified System
Certification
Verified System
System Test
Integrated System
Tested Modules
Testing for Safety

- **Overall safety validation** is the test that a system is in accordance with the safety requirements.
  - The results of it are documented in an **overall safety validation report**.

- Some standards require **traceability**, i.e. that the key safety requirements are traceable throughout all stages of the software life cycle.
Testing for Safety

- Testing for safety requires that tests are performed which show that each identified hazard is effectively countered.
  - Dynamic testing might be sufficient.
  - Since exhaustive dynamic testing is impossible, usually static analysis and mathematical modelling is required.
  - Properties like reliability and failure rates can usually not be tested dynamically, therefore static analysis is required.
The Roles of Testing

- Testing has three purposes in a safety-critical project:
  - Development testing.
  - Validation testing.
  - Production testing.
Development/Validation Testing

- **Development testing** is aimed at locating faults within the system, so that they may be removed.
  - Uses dynamic, static and modelling techniques.

- **Validation testing** aims at demonstrating the absence of faults and to demonstrate other positive features.
  - Uses again dynamic, static and modelling techniques.
Production testing aims at testing whether an individual unit has defects as a result of manufacturing or component fault.

▷ Tests the accuracy of the replication of the appropriate design.
▷ Production tests of software are easy and use usually techniques like checksums.
▷ Production tests of hardware are very complicated,
  since the number of possible faults is extremely big.
▷ Production testing is always dynamic.
7 (a) Basic Notions

7 (b) Dynamic testing

7 (c) Static Analysis

7 (d) Modelling
Material Moved to Additional Material

The material for this subsection has been moved to the additional material, which is available from the website.
7 (a) Basic Notions

7 (b) Dynamic testing

7 (c) Static Analysis

7 (d) Modelling
Material Moved to Additional Material

The material for this subsection has been moved to the additional material, which is available from the website.
7 (a) Basic Notions

7 (b) Dynamic testing

7 (c) Static Analysis

7 (d) Modelling
Material Moved to Additional Material

The material for this subsection has been moved to the additional material, which is available from the website.