LTAT.05.003
Software Engineering

Lecture 8:
Verification & Validation (Testing) I

Dietmar Pfahl
email: dietmar.pfahl@ut.ee

Fall 2018
Schedule of Lectures

Week 01: Introduction to SE
Week 02: Requirements Engineering I
Week 03: Requirements Engineering II
Week 04: Analysis
Week 05: Development Infrastructure I
Week 06: Development Infrastructure II
Week 07: Architecture and Design
Week 08: Verification and Validation I
Week 09: Verification and Validation II
Week 10: Continuous Development and Integration
Week 11: Refactoring (and TDD)
Week 12: Agile/Lean Methods
Week 13: no lecture
Week 14: Software Craftsmanship
Week 15: Course wrap-up, review and exam preparation
Structure of Lecture 8

• Testing Basics
• Testing Levels
• Testing Methods
• Testing Types
• Testing Artefacts
• Metrics
Exercise: A Pen

• Quality?
• Testing?
Software Quality – Definition

- **Software quality is the degree of conformance to explicit or implicit requirements and expectations**

Explanation:
- *Explicit*: clearly defined and documented
- *Implicit*: not clearly defined and documented but indirectly suggested
- *Requirements*: business/product/software requirements
- *Expectations*: mainly end-user expectations
Software Quality – Dimensions (1)

- **Accessibility:** The degree to which software can be used comfortably by a wide variety of people, including those who require assistive technologies like screen magnifiers or voice recognition.
- **Compatibility:** The suitability of software for use in different environments like different Operating Systems, Browsers, etc.
- **Concurrency:** The ability of software to service multiple requests to the same resources at the same time.
- **Efficiency:** The ability of software to perform well or achieve a result without wasted energy, resources, effort, time or money.
- **Functionality:** The ability of software to carry out the functions as specified or desired.
- **Installability:** The ability of software to be installed in a specified environment.
- **Localizability:** The ability of software to be used in different languages, time zones etc.
Software Quality – Dimensions (2)

- **Maintainability**: The ease with which software can be modified (adding features, enhancing features, fixing bugs, etc).
- **Performance**: The speed at which software performs under a particular load.
- **Portability**: The ability of software to be transferred easily from one location to another.
- **Reliability**: The ability of software to perform a required function under stated conditions for stated period of time without any errors.
- **Scalability**: The measure of software’s ability to increase or decrease in performance in response to changes in software’s processing demands.
- **Security**: The extent of protection of software against unauthorized access, invasion of privacy, theft, loss of data, etc.
- **Testability**: The ability of software to be easily tested.
- **Usability**: The degree of software’s ease of use.
Software Product Quality Model
– ISO 25010 Standard

Functional Suitability
- Appropriateness
  - Accuracy
  - Compliance
- Suitability
- Availability
- Fault tolerance
- Recoverability
- Compliance

Reliability
- Time-behaviour
- Resource-utilisation
- Compliance

Performance efficiency
- Appropriateness
  - recognisability
  - Learnability
- Ease of use
- Helpfulness
- Attractiveness
- Technical accessibility
- Compliance

Operability
- Appropriateness
- recognisability
- Learnability
- Integrity
- Non-repudiation
- Accountability
- Authenticity
- Compliance

Security
- Replaceability
- Co-existence
- Interoperability
- Compliance

Compatibility
- Modularity
- Reusability
- Analyzability
- Changeability
- Modification
- stability
- Testability
- Compliance

Maintainability
- Portability
- Adaptability
- Installability
- Compliance

Transferability

Software Quality Assurance (SQA)

- SQA is a set of activities for ensuring quality in software engineering processes (that ultimately result in quality in software products).

It includes the following activities:
- Process definition
- Process implementation
- Auditing
- Training

Processes could be:
- Software Development Methodology
- Project Management
- Configuration Management
- Requirements Development/Management
- Estimation
- Software Design
- Testing
- …
Software Quality Control (SQC)

- SQC is a set of activities for ensuring quality in software products.

It includes the following activities:
- Reviews
  - Requirement Review
  - Design Review
  - Code Review
  - Deployment Plan Review
  - Test Plan Review
  - Test Cases Review
- Testing
  - Unit Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing
Verification

Definition
• The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.

Objective
• To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.

Question
• Are we building the product right?

Evaluation Items:
- Plans, Requirement Specs, Design Specs, Code, Test Cases

Activities:
- All kinds of reviews and testing with exception of requirements review and acceptance testing
Validation

Definition
• The process of evaluating software during or at the end of the development process to determine whether it satisfies specified (or implicit) business requirements.

Objective
• To ensure that the product actually meets the user’s needs, and that the requirements were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.

Question
• Are we building the right product?
Software Development Life Cycle (SDLC)

- The SDLC, or Software Development Process, defines the steps/stages/phases in the building of software.
- Various kinds of software development models exist, e.g.:
  - Waterfall model
  - Spiral model
  - Iterative and incremental development (like ‘Unified Process’ and ‘Rational Unified Process’)
  - Agile development (like ‘Extreme Programming’ and ‘Scrum’)

SDLC in summary:
- Project Planning
- Requirements Development
- Estimation
- Scheduling
- Design
- Coding
- Test Build/Deployment
- Unit Testing
- Integration Testing
- User Documentation
- System Testing
- Acceptance Testing
- Production Build/Deployment
- Release
- Maintenance
Software Testing Life Cycle (STLC)

- The STLC defines the steps/stages/phases in testing of software.
# Software Testing Life Cycle (STLC)

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activity</th>
<th>Deliverables</th>
<th>Attitude needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements/ Design Review</td>
<td>You review software requirements/design (if existing)</td>
<td>Review Reports (listing the defects)</td>
<td>Curiosity</td>
</tr>
<tr>
<td>Test Planning</td>
<td>After gathering a general idea of what needs to be tested, you ‘plan’ for the tests</td>
<td>Test Plan, Test Estimation, Test Schedule</td>
<td>Farsightedness</td>
</tr>
<tr>
<td>Test Designing</td>
<td>You design/detail your tests on the basis of detailed requirements/design of the software</td>
<td>Test Cases, Test Data, Test Scripts, Requirements, Traceability Matrix</td>
<td>Creativity</td>
</tr>
<tr>
<td>Test Environment Setup</td>
<td>You setup the test environment (tools)</td>
<td>Test Environment</td>
<td>Interest in test technology</td>
</tr>
<tr>
<td>Test Execution</td>
<td>You execute your Test Cases/Scripts in the Test Environment to see whether they pass</td>
<td>Test Results (intermediate), Defect Reports</td>
<td>Patience</td>
</tr>
<tr>
<td>Test Reporting</td>
<td>You prepare various reports for various stakeholders</td>
<td>Test Results (final), Test/Defect Metrics, Test Closure Report</td>
<td>Accuracy, Diplomacy</td>
</tr>
</tbody>
</table>
STLC integrated with SDLC

Actual Needs and Constraints

User Acceptance (alpha, beta test)

Delivered Package

Review

System Test

Analysis / Review

System Integration

Integration Test

Analysis / Review

Subsystem

Subsystem Design/Specs

Integration Test

Module Test

Unit/Component Specs

User acceptance test

User review of external behavior as it is determined or becomes visible

Test Levels

 LTAT.05.003 / Lecture 10 / © Dietmar Pfahl 2018

INSTITUTE OF COMPUTER SCIENCE
Test Case

- A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.

- Templates and examples of formal test case documentation can be found here:

  http://softwaretestingfundamentals.com/test-case/
Test Case

A **Test Case** consists of:
- A set of inputs + expected outputs
- Execution conditions

Example of ‘execution condition’:
When pressing the ‘save’ button of a word processor, what happens depends on what you did previously (e.g., what you typed in or deleted)

---

**Test Suite** = set of Test Cases
**Test Data** = input to a Test Case
**Test Oracle** = condition that determines whether a test case passed or failed (-> fail happens if actual output is different from expected output)
**Test Verdict** = decision of whether a test passed or failed
Next Lecture

- Date/Time:
  - Friday, 02-Nov, 10:15-12:00
- Topic:
  - Verification and Validation (Testing) II
- Labs:
  - Work on homework assignment 4
  - Go to assessment labs on homework assignment 4