Software Testing ETS200

Projects in Software Testing
Per Runeson
with contributions from Carina Andersson, Elizabeth Bjarnason, Markus Borg, Emelie Engström, Hussan Munir, Mika Mäntylä and Thomas Thelin, Lund University, Dept. Computer Science
January 16th, 2017

Contents

1 Introduction ........................................................................................................................................... 2
2 Learning Objectives ............................................................................................................................. 2
3 Activities ............................................................................................................................................... 2
4 Schedule .............................................................................................................................................. 3
5 Assessment .......................................................................................................................................... 3
6 Report ................................................................................................................................................. 4
7 Presentation ......................................................................................................................................... 4
8 Research Project Areas ....................................................................................................................... 4

8.1 How can different testing techniques (methods) be compared? ....................................................... 4
8.2 How Acceptance Test Harness is used to test Jenkins core and its plug-ins? .................................... 4
8.3 How is testing performed for Agile processes? .................................................................................. 5
8.4 How do you effectively regression test your system? ........................................................................ 5
8.5 Can test cases be generated automatically from the code? .............................................................. 5
8.6 How to handle defect reports in large-scale software testing? .......................................................... 5
8.7 What are the minimal test practices needed in a start-up company? .................................................. 6
8.8 Is it more efficient to generate test cases from a model? .................................................................... 6
8.9 How do you test for security? ........................................................................................................... 6
8.10 Do Static Code Analysis tools really help in improving software code quality? ............................... 6
8.11 Techniques for Testing Android Applications ................................................................................ 6
8.12 How do you know that your test cases are effective? ...................................................................... 7
8.13 How can you measure that the software system is reliable? ............................................................ 7
8.14 What techniques are most effective to verify and validate requirements and design documents? ...... 7
8.15 What techniques can be used to know when to stop testing? .......................................................... 7

9 Example Practical Project Areas ......................................................................................................... 8

9.1 Evaluate tools for software testing ................................................................................................ 8
9.2 Evaluate test methods ...................................................................................................................... 8
9.3 Test a research tool .......................................................................................................................... 8

10 Journals and Conferences ................................................................................................................. 8

11 Keywords in Software Testing .......................................................................................................... 8
1 Introduction

This document gives the practical details regarding the project in the course Software Testing. The project in the course is equivalent to 2 weeks of full time studies (3 ECTS credit points). The main objective of the project is to gain a deeper understanding of a specific area within verification and validation (V＆V) of software products. A project group consists of 4 (±1) persons, who perform the project together. All project members should be involved and the total effort should be evenly distributed among participants.

2 Learning Objectives

The objective of the project is to learn a specific area of software testing. Furthermore, the structure of the project makes it possible to learn important parts that will be practiced later on in the education. Hence, the main learning points of the project are:

- Learn a specific area of software testing
- Collect and summarize research information
- Critical thinking beyond the written information
- Present information in a structured way

There are two types of projects to choose among:

- Research project: Solve a research problem; survey the state-of-the-art and identify the research problems in some area; develop and justify an extension to an existing technique; etc.
- Practical project: Evaluate a testing technique/tool or use a testing technique to verify a system or design. This entails applying a testing technique or a commercial testing or analysis tool on a software system.

3 Activities

The main activities in the project are:

1. **Decide project type**, i.e. choose whether you want to perform a research project or a practical project.

2. **Decide on a subject**. There are some test subjects specified in this document. Only one group may choose a specific subject. Other subjects may be chosen, but have to be agreed with your project supervisor.

3. **Find relevant literature**. In each subject area, there is a suggestion of research reports to start reading. Other research literature can be found in the LUBsearch database http://www.lub.lu.se (within the domain of LU, or by using your STIL password). For research projects, 7-10 references to related research literature should be chosen. For practical projects, somewhat fewer relevant research references are needed.

4. **Outline the report**. Specify heading levels 1 and 2, write in short sentences what will be included in the sections. Submit to ets200@cs.lth.se and ets200.lu@analys.urkund.se with ‘outline’ and your student ID:s in subject.

5. **Read and perform practical work**. Read and understand the new area of software testing and, if a practical project has been chosen, perform practical work.

6. **Write the report**. The report is to be written in English. The report shall be written in the IEEEtran Proceedings Format. Submit to ets200@cs.lth.se and ets200.lu@analys.urkund.se with ‘project’ and your student ID:s in the subject line.

7. **Consult the supervisor**. During reading and writing, you are offered consulting by the supervisor at one occasion. Book meeting time in advance.

8. **Present the report**. Communicate the area to other people who have knowledge of software testing, but not of the specific area. This is done during the course conference. Each group will be given a time slot in which to present their work. The presentation shall be in English.

---

1 For Word: [https://www.conference-publishing.com/templates/MSW_USLtr_format.doc](https://www.conference-publishing.com/templates/MSW_USLtr_format.doc) For LaTeX: use the LaTeX class file IEEEtran v 1.8 ([http://www.ctan.org/tex-archive/macros/latex/contrib/IEEEtran/IEEEtran.cls](http://www.ctan.org/tex-archive/macros/latex/contrib/IEEEtran/IEEEtran.cls)) and the following configuration (without option ‘compsoc’ or ‘compsocconf’):
\documentclass[conference]{IEEEtran}
4 Schedule

<table>
<thead>
<tr>
<th>Course week</th>
<th>Activities</th>
<th>Hand in to supervisor</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create group and decide on a subject</td>
<td>Decided subject for group</td>
<td>Fri Jan 20</td>
</tr>
<tr>
<td></td>
<td>Search for literature</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td>Search literature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read literature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outline the report</td>
<td>Outline</td>
<td>Tue Jan 31</td>
</tr>
<tr>
<td>4-5</td>
<td>Read literature</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Write report</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Consult supervisor</td>
<td>Book time</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Write report</td>
<td>Report to supervisor</td>
<td>Fri Feb 24</td>
</tr>
<tr>
<td>7</td>
<td>Prepare presentation</td>
<td></td>
<td>Thu Mar 2</td>
</tr>
<tr>
<td></td>
<td>Course conference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Update report</td>
<td>If needed, update report to supervisor</td>
<td>Fri Mar 17</td>
</tr>
</tbody>
</table>

5 Assessment

The reports are scored on a scale fail–pass–pass with distinction (U, G, VG) on each of the following four characteristics:

- **Form (20% of total marks):**
  - Correct use of IEEE template (formatting of page, title, names, headings, paragraphs, table and figure captions, references, etc.)
  - Correct referencing style; also: each document listed in the reference section must be referenced from the text at least once
  - Language: no spelling/grammar errors, clarity of expression, appropriateness of expression (no slang!), correct usage of terminology
  - Observe page limit (5-7 pages, all inclusive)

- **Work (35% of total marks):**
  - Appropriate scoping (focus) of work
  - Sufficient depth of work (e.g., number, relevance, appropriateness of literature found)
  - Appropriateness/correctness and level of detail of analysis
  - Discussion of limitations of the work done
  - No over-interpretation of findings
  - No unsubstantiated claims (i.e., claims must either be based on the results of the analysis or be based on results found in the (relevant) literature)

- **Content (35% of total marks):**
  - Structure in line with project description (see Section 6)
  - Correct flow of argument / consistency of argument
  - Analysis must relate to research problem
  - Discussion must be in the light of analysis results and research problem

- **Presentation (10% of total marks):**
  - Clarity of message
  - Contact with audience
  - Timeliness

The reports should be original work by the student groups, and adhere to the department’s guidelines on plagiarism\(^2\). The project reports will be checked via Urkund\(^3\)

---

\(^2\) [http://cs.lth.se/english/education/cooperation_or_plagiarism](http://cs.lth.se/english/education/cooperation_or_plagiarism)

\(^3\) [http://www.urkund.se](http://www.urkund.se)
6 Report
The report shall be written using the IEEE template (see footnote on page 2) and comprise 5-7 pages. The report shall contain:

- **Abstract**: summary of all parts in the report. The purpose of the abstract is to attract people to read your report.
- **Introduction**: introduce the chosen area. The purpose is to give an introduction to the reader who is not familiar to the specific area, but knows software engineering and testing generally.
- **Description** of the chosen area: summary of the chosen area. The purpose is to describe why, how and for which purpose the area is used.
- **Analysis**: analysis, critical thinking and future plans within the chosen area. For research projects, the purpose is to describe the benefits, drawbacks, what research the area needs in the future. For evaluation and practical projects, the work should be described here together with a shorter analysis.
- **Conclusion**: main conclusion in the report. The purpose is to discuss the main points in the report again.

Another structure can be chosen after discussion with your project supervisor.

7 Presentation
The project will be presented at the project conference at the end of the course. Each project will be given 15 minutes to present their chosen area. The presentation should cover the main points of summary, analysis and conclusion. The project decides how to share presentation roles between them.

8 Research Project Areas
Below are some proposed project areas and an overview article related to each area. They are primarily aimed for research projects, but may also be starting points for practical projects. If you would like to choose another area, describe the area to your project supervisor before starting the work. The keywords in Section 9 may be used when you choose a project topic and when you search for literature.

8.1 How can different testing techniques (methods) be compared?
One research methodology is to use empirical methods in order to evaluate which testing methods are best to use. The methodologies are often divided into experiments, surveys and cases studies. All these methods are important in order to help software organizations to choose the right testing technique for their purpose.


8.2 How Acceptance Test Harness is used to test Jenkins core and its plug-ins?
This project consists of a reusable harness that can be used by plugin developers and users to write functional test cases. These tests can be run against Jenkins instances that are deployed in all sorts of different ways, and can interact with complex real fixtures. These tests can be also run with specific version of Jenkins core and a combination of plugins. How this project has been evolved since its introduction, and how does it help to improve the testing in Jenkins?

8.3 How is testing performed for Agile processes?

Test-driven development (TDD) is often applied in agile software development. These principles are based on that test cases are first derived and then used to specify how the system should work, as well as, used as test cases when the system is developed. In such development projects, the system behavior (or requirements) is discussed by testers and developers in close collaboration with their product owner. There are some interesting issues in this area. For example, what is the relationship between requirements and test cases? can test cases be used to document requirements (thereby saving documentation effort)?; do such systems better satisfy what the customer wants in terms of functionality and quality?


8.4 How do you effectively regression test your system?

Regression testing is the process of validating modified software to detect whether new errors have been introduced into previously tested code. Because of time and resource constraints for testing, regression test selection techniques have been proposed, to reduce the expenses.


8.5 Can test cases be generated automatically from the code?

There is a class of testing methods, called search-based testing, that generate test criteria from the code. The approach is promising, but there are of course limitations to it. A practical project in this area may involve use of some open source tools for search-based testing.


8.6 How to handle defect reports in large-scale software testing?

Defect reports are important means for communication within an organization, for example between testers and developers. However, in large-scale software engineering, the number of defect reports quickly grows large, and hence become costly. One problem is duplicate reporting of known problems. Another is the quality of the defect reports as such. Yet another is who is going to fix the problem.


8.7 What are the minimal test practices needed in a start-up company?

When starting a new software company, the quality of the software is important to build trust among customers. However, startups generally have very little time to spend on testing. What is the minimal level of test practices needed to support a small software company?


8.8 Is it more efficient to generate test cases from a model?

Model-based testing is a popular term for generating test cases from various models, for example UML specifications. The advantage is that new test cases can be generated at low costs, once the model is defined. Maintenance of the test model may also be lower than maintaining individual test cases for evolutionary development.


8.9 How do you test for security?

Security is an issue for more and more software systems, as they become more flexible, and also handle more and more sensitive information, e.g. banking or health information. How can you test for security in a systematic and efficient way?


8.10 Do Static Code Analysis tools really help in improving software code quality?

Every static analyzer has a database of vulnerabilities to look for in code. Most of the products have the option of adding custom rules. Static analysis may be performed on modules or unfinished code, although the more complete the code, the more thorough and accurate the analysis can be. What static code analysis tools are available, and what are their strengths and weaknesses?


8.11 Techniques for Testing Android Applications

Android applications can be actually considered Event Driven Software (EDS) whose behavior is driven by several types of events. Hence, a major issue in Android application testing is that of assessing which testing approaches usable for traditional EDS systems (such as GUIs, Rich Internet Applications, embedded software, etc.) are also applicable for Android based mobile applications and which tuning and technological adaptations are needed for them. What techniques are available to test Andriod applications? How can we compare and contrast the existing techniques?

8.12 How do you know that your test cases are effective?

Test cases are developed to check whether a software system is implemented correctly according to the requirements specification. However, if the test cases are badly chosen, they will not detect the failures. Mutation testing can be used to assess whether the test cases are feasible and effective.


8.13 How can you measure that the software system is reliable?

Software testing is performed to detect the failures in the software. There are different metrics to use in order to measure whether the software is correct. One such measure is reliability, which is defined as: “The probability for a failure-free operation of a program for a specified time under a specified set of operating conditions.” (IEEE 610.12-1990). There are several models and techniques that can be used, for example, reliability growth models, Markov models, statistical usage testing, usage-based testing and operational profile testing.


8.14 What techniques are most effective to verify and validate requirements and design documents?

Static verification is often used in the beginning of the development of software. There are several different techniques and methods used to check the static (non-executable) representation of a system in terms of requirements and designs. The common feature of these is that they have to be manually checked by reviewers. Software inspection, reviews, walkthroughs are common techniques, which are used together with reading techniques.


8.15 What techniques can be used to know when to stop testing?

Stopping criteria are used in the testing phase to determine when a certain quality level has been achieved. The quality can for example be defined as the reliability or just the number of faults left in the system. There are several techniques to use as a stopping criterion. One could, for example, estimate the number of faults left after an inspection or to estimate the number failures left in testing.


9 Example Practical Project Areas

9.1 Evaluate tools for software testing

Tools are important in order to implement software testing effectively in an software organization. However, although tools are needed, they do not solve the problems in the testing phase. Several tools exist and they have to be evaluated before a software organization purchases one. Choose an area of interest, select a set of tools and compare in a systematic way.


9.2 Evaluate test methods

Test methods may be more or less feasible for specific software systems. Therefore, it may be relevant, in addition to general evaluations, also evaluate test methods for specific systems or classes of systems. For a specific program, choose 2 or 3 test methods (one white-box and one black-box, or two methods of one type) and generate test cases from those, run the tests, log the outcome and assess which method is best for the current system.

9.3 Test a research tool

In our research, we are currently developing a tool, SERP connect (http://serpconnect.cs.lth.se) which aims at guiding practitioners to find useful knowledge in research. This project aims at identifying needs for testing of the tool, for example, database test or GUI test, define a proposed setup and conduct tests.


10 Journals and Conferences

In Table 1 (below) some journals and conferences are listed. These publish results of software testing research.

11 Keywords in Software Testing

The keywords in Table 2 can be used if you want to decide on a subject not specified above. Another source that is appropriate to use is the Software Engineering Book of Knowledge. The latest version can be downloaded on http://www.swebok.org.

<table>
<thead>
<tr>
<th>Journals</th>
<th>Conferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM SIGSOFT Software Engineering Notes</td>
<td>Int. Conf. on Software Testing, Verification and validation (ICST)</td>
</tr>
<tr>
<td>ACM Transactions on Software Engineering and Methodology</td>
<td>Int. Conf. on Software Engineering (ICSE)</td>
</tr>
<tr>
<td>Empirical Software Engineering</td>
<td>Int. Conf. &amp; Workshop on the Engineering of Computer-Based Systems (ECBS)</td>
</tr>
<tr>
<td>IEEE Software</td>
<td>Int. Conf. on Empirical Software Engineering and Measurement (ESEM)</td>
</tr>
<tr>
<td>Information and Software Technology Journal of Systems and Software</td>
<td>Int. Conf. Conference on Automated Software Engineering (ASE)</td>
</tr>
<tr>
<td>Software Engineering Journal</td>
<td>International Symposium on Software Reliability Engineering (ISSRE)</td>
</tr>
<tr>
<td>IEEE Transactions on Software Engineering</td>
<td>Int. Conf. on Software Testing and Analysis (ISSTA)</td>
</tr>
<tr>
<td>IEEE Proceedings Software Engineering</td>
<td>Software Quality Journal</td>
</tr>
<tr>
<td><strong>Black-box testing</strong></td>
<td><strong>Gray-box testing</strong></td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Equivalence partitioning</td>
<td>Reference models from code-based testing (flow-graph, call-graph)</td>
</tr>
<tr>
<td>Boundary value analysis</td>
<td></td>
</tr>
<tr>
<td>Decision table</td>
<td></td>
</tr>
<tr>
<td>Finite-state machine-based</td>
<td></td>
</tr>
<tr>
<td>Testing from formal specification</td>
<td></td>
</tr>
<tr>
<td>Error guessing</td>
<td></td>
</tr>
<tr>
<td>Random testing</td>
<td></td>
</tr>
<tr>
<td>Operational profile</td>
<td></td>
</tr>
<tr>
<td><strong>Application testing</strong></td>
<td><strong>Testing in the software process</strong></td>
</tr>
<tr>
<td>Object-oriented testing</td>
<td>Connection between requirements and testing</td>
</tr>
<tr>
<td>Component-based</td>
<td>Architecture Testability</td>
</tr>
<tr>
<td>Web-based</td>
<td>Code-based</td>
</tr>
<tr>
<td>GUI testing</td>
<td>Fault-based</td>
</tr>
<tr>
<td>Testing of concurrent programs</td>
<td>Testing from formal specification</td>
</tr>
<tr>
<td>Protocol conformance testing</td>
<td>Random testing</td>
</tr>
<tr>
<td>Testing of distributed systems</td>
<td></td>
</tr>
<tr>
<td>Testing of real-time systems</td>
<td></td>
</tr>
<tr>
<td>Testing of scientific software</td>
<td></td>
</tr>
<tr>
<td><strong>Objectives of testing</strong></td>
<td><strong>Evaluation of the testing</strong></td>
</tr>
<tr>
<td>Acceptance testing</td>
<td>Coverage measures</td>
</tr>
<tr>
<td>Installation</td>
<td>Fault seeding</td>
</tr>
<tr>
<td>Alpha, beta</td>
<td>Mutation score</td>
</tr>
<tr>
<td>Conformance, functional, correctness</td>
<td>Comparison and relative effectiveness of different techniques</td>
</tr>
<tr>
<td>Regression</td>
<td>Stopping criteria</td>
</tr>
<tr>
<td>Performance, stress</td>
<td></td>
</tr>
<tr>
<td>Recovery</td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
</tr>
<tr>
<td><strong>Estimations in software testing</strong></td>
<td><strong>Empirical methods in software testing</strong></td>
</tr>
<tr>
<td>Reliability</td>
<td>Compare testing techniques</td>
</tr>
<tr>
<td>Reliability growth models</td>
<td>Compare inspection and testing</td>
</tr>
<tr>
<td>Types and classification of faults</td>
<td></td>
</tr>
<tr>
<td>Remaining faults and faults density</td>
<td></td>
</tr>
<tr>
<td><strong>Inspections</strong></td>
<td><strong>Test management</strong></td>
</tr>
<tr>
<td>Process</td>
<td>Unit</td>
</tr>
<tr>
<td>Reading techniques</td>
<td>Integration</td>
</tr>
<tr>
<td>System</td>
<td></td>
</tr>
<tr>
<td>Alpha, Beta</td>
<td></td>
</tr>
<tr>
<td><strong>Automated software testing (tools)</strong></td>
<td><strong>Simulation of the test process</strong></td>
</tr>
<tr>
<td>Tools for different techniques</td>
<td>Discrete event models</td>
</tr>
<tr>
<td>System dynamics models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk-based</td>
</tr>
</tbody>
</table>