Software Testing 2017

Justin Pearson

October 30, 2017

1 Introduction

The purpose of this document is to make clear how will be examined, and what is required of you in the course. This course essentially consists of four components:

1. A series of lectures on aspects of software testing;
2. Group work on test design;
3. A lab on test driven development;
4. An exam.

The exam is graded U,3,4,5. The group work and the project is pass or fail.

If you only learn one thing about testing during this course, then you will fail, but if you learn the following idea "You should have a reason for each test. For each test you should be able to explain what you are trying to test for." then you will at least understand why we study testing. When we examine you, and when you produce tests, we expect you to be able to explain the reasons behind your tests.

2 Deliverables and requirements 2017

1. A lab on test driven development (TDD) for BibTex. This is done in groups of 2 or 3. It will be marked at the lab. There is no hand-in.
2. Project work on test case design for some python library. This is done in groups of 4.
3. An exam.

Information on the lab can be found either linked from the student portal or via `http://user.it.uu.se/~justin/Teaching/Testing/index.html`. Information on the exam will come later. It is safe to say that if I mention a concept on the slides, then I might ask an exam question on it. It is also safe to say, that my slides do not contain all the information that you need. You will probably have to read the book to get enough information to pass the exam.

2.1 Project Work

The idea of the project is to pick some python library, and write some test cases for it. You will have to write both black-box and white-box tests and tests that provide coverage of selected parts of the code. The final deliverable will be a written document and a presentation on the 30/11.

- By the start of week 46 you must form a group. This will be handled via the student portal.

- In week 46 you will meet your assigned lab assistant; before the meeting you should pick a python library (for example look at the Python Standard Library or `https://pypi.python.org/`). You must agree with your lab assistant that you can work on that library. We have to decide if the library is too trivial or too complicated.
- You are to construct a number of test cases:
  - Black box testing of the API. You are to produce test cases that cover the API.
  - White box testing. In agreement with your lab assistant you are to pick some areas of code in your library to cover. Together with the python library `https://coverage.readthedocs.io/en/coverage-4.4.1/` you are to produce test cases that not only provide statement coverage but some sort of path coverage. You need to document and motivate your design of the test cases. You should meet your lab assistant in week 47.
  - In week 48 or 49 you again meet with your lab assistant to discuss your draft of your report.
  - On 30/11 there is a compulsory presentation, where you are to present your library and present some of your tests cases and the reasoning behind them. Your presentation is of work in progress.
  - Jan 8th 2018 is the deadline for the final report.

Summary of meetings

- Week 46 — Discuss your choice of library.
- Week 47 — Discuss your test cases and choice of code that you are going to provide test coverage.
- Week 48 or 49 — Discuss the draft of your report.

Summary of classes of test cases

- Black tests of the API. You should provide test cases that cover a large part (or all) of the API of the library.
- White box tests that provide coverage of part of the library.

When documenting your test cases you have document what your tests are designed to do.

What should your report contain

- Description of the python library with examples of use. This should be written for somebody who has not read the documentation.
- Documentation and all tests cases.

3 Grading criteria for written deliverables

- Clarity of presentation. It is important that your document can be read without reference to the documentation of the library. This means that you will have to reexplain things already in the documentation.
- Quality of the test cases. What sort of coverage do their provide? Do they test enough functionality.
- Documentation of the test cases. These can be in comments in your code, for each test case of group of test cases you must provide reasons for the tests and document what you are trying to test.