Testing Techniques 2016 – 2017
Assignment 1

1 Test Approach

You have chosen a SUT (System Under Test) that you will use in these assignments.

Suppose that your SUT is a product that has been developed by some software company. Suppose you are working at a test company. The software company asks you, the test company, to perform independent testing of the product.

The first step for testing is to write a *test approach* document for the SUT. In the *test approach* document you identify the SUT and describe what kinds of testing you would plan to perform for the SUT.

You can use the hints and template in "Hints on Writing a Test Approach" below. Be concise and to-the-point; the total length of the core document shall certainly not exceed 8 pages; probably it will be shorter. It may be that you have to provide additional documents, e.g., describing the specification of your SUT. Provide these as separate documents, and refer to them in the test approach document.

Note: It is not the intention that your group will actually perform all the testing that you describe in the test approach. Of course, try to show what you learned about software testing up to now.

2 Manual Testing

Manually developing and executing black-box functionality test cases for your SUT:

1. Describe and explain how black-box functionality test cases for your SUT look like: the structure of tests, the points of control and observation, i.e., the interfaces at which you test, and the domains of possible inputs and outputs.
2. Develop (at least) 10 black-box functionality test cases to test your SUT manually. Justify the choice of these test cases, e.g., with reference to classical, black-box testing techniques.
3. Test your SUT manually with the developed test cases, and analyse the results.

   (It may be that some interfaces of your SUT cannot be accessed with manual execution, in which case you will have to develop some way to access the interfaces).
4. Discuss the quality, completeness, and coverage of your manual test suite, and, if possible, use some measure, or other arguments to assess its completeness.

Hints on Writing a Test Approach

In the course we use the following description of *software testing*:

> a technical process, performed by executing/experimenting with a product, in a controlled environment, following a specified procedure, with the intent of measuring one or more characteristics/quality of the software product, by demonstrating the deviation of the actual status of the product from the required status/specification.

From this description we can deduce that a test approach must at least describe:

- the product that will be tested,
- the controlled environment in which it will be tested,
- the specified procedure following which it will be tested,
- the quality characteristics that will be tested, and
- the specification.

Moreover, according to ISTQB, a *test strategy* is:

> a high-level description of the test levels to be performed and the testing within those levels for an organization or programme (one or more projects).

And a *test approach* is:

> the implementation of the test strategy for a specific project; it typically includes the decisions made that follow based on the (test) projects goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria, and test types to be performed.

Altogether, a number of items must be addressed in a test-approach document. The proposed items are listed below; they can be used as a template for a test-approach document.

When elaborating this template, be concise and to-the-point, mentioning those aspects that are specific for your project. Explaining standard testing concepts or techniques is not necessary: everybody reading a test-approach document can be assumed to have basic testing knowledge. If necessary, refer to – but do not repeat – standard terminology, e.g., the ISTQB Glossary. It may be that not all items are applicable in your case, but then mention that explicitly.

Proposed Template for a Test Approach

1. **Introduction**
   State the objectives and overview of the document at a high-level.
2. **Test Goal**
   What is the overall goal of the testing effort, what are the final deliverables, who are the stakeholders, i.e., for whom are you doing it, applicable laws and (international) standards.
3. **The Product**
   Identification of the SUT: What is the product (SUT – System Under Test) being tested, its version, its operation context, required platform, its interfaces, and how is it executed.

4. **The Specification**
   What is the test basis, i.e., its specification, and all documentation describing what the SUT shall do. (Do not include specification documents, but refer to them.)

5. **Risks**
   What are the risks of the product (at a high level), of the development process, and of the test process. How are risks handled and mitigated.

6. **Test Environment**
   What is the (controlled) environment in which experiments are performed, what is the test architecture, i.e., how are SUT and test system positioned and connected, which environment and infrastructure (hardware, software, middleware, databases, libraries, ...) are required for testing, how to access the SUT and its interfaces, which stubs and drivers are needed, are tests performed in a laboratory, production, or user environment.

7. **Quality Characteristics**
   Which quality characteristics are tested (IS 9126 or other quality model: functionality, reliability, usability, ...),

8. **Levels and Types of Testing**
   Which levels and types of tests are performed: (V-model: unit, integration, module, system, acceptance, ...), which units, components, subsystems, ... are tested and for what, accessibility (white/black box), verification vs. validation tests, ...

9. **Who will do the Testing**
   Who tests what, and what are the roles: developer, (independent) tester, user, alpha, certification, ...),

10. **Test Generation Techniques**
    As far as already known or required, e.g., by applicable standards: black-box (equivalence partitioning, boundary value analysis, error guessing, cause-effect graphing, decision tables, state transitions, use case testing, exploratory testing, ...), white-box (path, statement, (multiple) condition, decision/branch, function, call, loop, MC/DC coverage, ...), mutation testing, combinatorial testing, ...

11. **Test Automation**
    As far as applicable, which parts of the testing will be automated, which test tools will be used in the various phases of the testing process (planning, preparation, test generation, test execution, completion), which tests are performed manually, what is automated, and which tools have to be obtained or developed.

12. **Exit Criteria**
    What are the criteria for going from one test phase to the next, when is testing finished, when is the product considered sufficiently tested, what are the (final) evaluation criteria.

13. **Testware**
    Which test products are recorded, consolidated, and kept for reuse.

14. **Issue Registration**
    How are issues (defects) registered, analysed, reported, and handled.