Software Test Documentation

Youssef Boudiba
1.0

1 Introduction

The software test documentation is based on the IEEE standard for software test documentation (829-1998). Since we are working on a small project, we will not cover every topic proposed by the standard, but will instead focus on those who are more relevant to our project.

This document consists of 3 parts:

1. Test specification:

Consists of describing (major parts of) the software test plan for our project. By defining a test plan we will be able to cover the management of the tests, how we are going to run those tests, what we are going to tests, etc.

2. Test monitoring:

We will keep tracks of the tests through a test log and report incidents via an incident report in the second part of the document.

3. Test report:

Last but not least is the test report. This will be an overview (summary) of some important tests as well as a conclusion of the software test documentation.

Record of revision

Version	Date	Description
0.1	8/11/2013	Initial draft
0.2	10/11/2013	Added additional content
0.3	8/12/2013	Removing unnecessary content and updating topics
1.0	12/12/2013	Ready for delivery: iteration 1

References

1. ***Software project management plan (SPMP):***

   https://github.com/se1-1314/xiast-docs/blob/master/management/project/spmp.pdf

2. ***Software requirements specifications (SRS):***

   https://github.com/se1-1314/xiast-docs/blob/master/management/requirements/srs.pdf

2 Test Specification

This section describes the software test plan.

2.1 Approach (Testing strategy)

Since we are following an iterative and incremental development model for our project (check SPMP for more info), our test strategy will consist of a series of different tests. These tests are specified below and are done by the whole group since the absence of a testing team.

1. **Unit testing**

   Every small amount of source code (or low level software module) that cannot be divided in smaller part is called a unit. Whenever a developer has finished working on such a unit, it will be tested to check whether or not it is conform to its design and requirements.

2. **Integration testing**

   Integration testing combine and test low level software modules that have been unit tested. This test assures us that the combined higher level module still works and that it is conform to its design and requirements.

3. **System testing**

   System testing tests and combines all integrated (higher level) software component. The test will be performed on the whole system in order to find deviations from the specified requirments.

2.2 Features (not) to be tested

Since we'll most likely follow the test driven development cycle (TDD), every unit will have its test case. If changing a unit's content doesn't cause any test to fail, then the unit will be consider useless and must be deleted (unless a test is missing to support its existence).

2.3 Pass/fail criteria

A test pass if no error occurs and if it's conform to its design and requirements. Whenever an error occurs whilst running a test, the responsible developer will open an issue on gitHub and try to find a solution (with whole team if necessary). When a solution is found, a test incident report must be generated to keep track of the occurred error (see 3. test execution for more info).

2.4 Testing tools

clojure.test The clojure library itself provide testing tools for testing the code and can be found at http://richhickey.github.io/clojure/clojure.test-api.html. Since no testing framework has been decided yet we'll use this library in the meantime.

3 Test Monitoring

Both the treatment of errors as keeping track of tests results is described below in this section. We will do this by generating two kinds of documents:

3.1 Test log

A test log keeps track of what happened during a test and must contain:

1. ***Execution description***
2. ***Results***

We will be using gitHub to store the test log under a specific directory.

3.2 Test incident report

We will keep track of incidents (bugs, unexpected crashes, unexpected results of a test, etc.) by using the issue tool on gitHub and generating a report whenever and issue has been resolved. Such a report consists of:

1. ***Problem identifying***

1. ***Incident description***
2. ***Impact on system components***

2. ***Problem solving***

3. ***Priority***
4. ***Actions taken***
5. ***Solution***

We will be using gitHub to store the test incident reports under a specific directory.

4 Test Report

This section contains the result and evaluation of important tests as well as a conclusion to the software test document.