Software Testing

Assignment 2

Test Strategy

Robert Willem Hallink
rh222ff@student.lnu.se

17 December, 2014

Abstract

This report contains the test plan, test cases and the test report which are all part of assignment 2.

Report in the course Software Testing at the School of Computer Science, Physics and Mathematics, Linnæus University.

Table of Contents

1 Test Plan

There are three different types of tests: unit testing, integration testing and system testing. The developers themselves do unit testing while testers do the other types of testing.

1.1 Roles

This assignment team exists out of one member: Myself, Robert Hallink. Therefore I am taking on the role of tester, developer and manager. In short, I am taking on me all available roles. Therefore in this assignment, roles and responsibilities definition is not that important.

1.2 Tools and Environment

All parts of the assignment are done on Windows 8.1 (x64). No other systems are used or will be used. Not guaranteed, but the assignment will mostly likely without any problems run on Windows 7 and Windows Vista. Older operating systems could also work without any problems, however it is more unlikely to work than newer operating systems. No support is given for Mac and Linux systems, however it is unlikely it would not work. No further support will be given.

Software that has been used are:

- LaTeX , for writing the report
- Eclipse 4.4.1, java development tool
- JUnit 4.12, tool for unit testing within Eclipse
- Mockito 1.9.5, tool for mock objects within Eclipse
- General software, not worth mentioning such Google Chrome

1.3 Risks

This assignment contains a few risks. But these risks mostly cover the same problem. As this assignment is done by only one person, no checks will be preformed by other team members. As said, one a part of the assignment is done, I will move on and not check it over. Only when requirements are outrightly not implemented will checks be preformed. The definition is check is to see if no problems are found within tasks (which could be documentation or java code) and improve it when it is not flawless. These risks could be found within:

- Documentation
- Java Code in general
- This report
- Unit test cases
- The only available youtube video

In order to get this assignment moving and finished before the deadline, it is not possible to check it many times. Checks can exists out of simple and fast checks or expensive checks. Fast checks are often done by gazing on the surface of a part of the assignment and see if it executed as expected, if it is correct no further actions are taken. Expense checks are done by going over the entire part of the assignment, parts could be improved regardless of being wrong or correct.

2 Test Cases

An custom use case on how to run the My Web Server with the provided source code from LNU.

2.1 Requirements

- Should not crash.
- Should print output of what happens.
- Should print the output from previous item into a log file.
- Needs to work on Windows 8.1 at least, but also older Windows systems, Mac and Linux.
- Should show required and explain requirement software if needed.

2.2 Actors

Administator: Runs, installs, stops and inspects.

2.3 Use Case 1

Running the program from java source code.

2.3.1 Primary Actor

Administrator.

2.3.2 Precondition

- Java development tool be installed (ex: NetBeans or Eclipse).
- JRE (Java Runtime Environment) and JDK (Java Development Kit) be installed, preferable version 8.

2.3.3 Postcondition

- Log file should be written after or during execution.
- Visual effect or text of what happend.

2.3.4 *Main scenario*

1. Start the java development kit.
2. Find the class containing the main method.
3. Run the class containing the main method.
4. Follow instructions that are given by the program and let it complete.
5. Be able to use CodeCover on the ran code, which can only be done if the program executed without any unexpected issues.

2.3.5 *Alternative scenarios*

1. In case the program crashes:
   (a) Print out error on screen.
   (b) Save the error in a log file.

3 Test Report

The test report from the provided source [1] for assignment 2. The source code was not written, changed or adjusted by myself.

3.1 HTTPServerConsole

The java class containing the main method is named: HTTPServerConsole. Upon running this class the program simply crashes saying that the program should only have one or two arguments. This error links to row 24: ConsoleView view = new ConsoleView(args, System.out); which in turn is used by the class ConsoleView, which leads into row 36: throw new WrongNumberOfArgumentsException();.

It remains unclear if the program did anything at all. The only visual results of running the program is the exeception above being thrown. It makes me suggest that the program is simply broken.

3.2 CodeCover

CodeCover is not avaible for use within JUnit tests, it simply crashes. However, the JUnit itself preforms correctly without any issues. In order to use CodeCover it has to be done by running the main method of a program. However, as stated above, the main method crashes on startup. CodeCover requires that the main method of the program can execute without unexpected issues. However, this is not the case and CodeCover can therefore not be used to test if all parts of the provided source code are used.

3.3 JUnit

The provided source code contains 19 different JUnit tests. Each JUnit test contains either 1, 3, 4 or 5 tests, with one JUnit test containing 10 tests. Out of all 19 JUnit tests, four of them are not correctly done. These four JUnit tests are:

- SharedFolderTest.java
- StressTest.java
- HTMLFileResponseTest.java
- ConsoleViewTest.java

SharedFolderTest.java contains three tests:

- testGetNonExistantFile - suceeds
- testGetIllegalFile - suceeds
- testGetRootURL - fails

StressTest.java contains one test:

- stressTest - fails

HTMLFileResponseTest.java contains one test:

- testWriteResponse - fails

ConsoleViewTest.java contains ten tests:

- testNoArguments - suceeds
- testOkDirectory - suceeds
- testNotADirectory1 - suceeds
- testDoNotStop - suceeds
- testShowHelp - suceeds
- testDoStop - suceeds
- testOkPort - suceeds
- testCrapArgument1 - suceeds
- testCrapArgument2 - suceeds
- testPortTaken - fails

All other non-mentioned JUnit tests work properly without any fails. The all failing JUnit tests only contain one test that fails. All failed tests, except for stressTest contain a similar problem such as ports or html. This leads to believe me that the written tests are not able to connect with an outside internet connection.

References

[1] Daniel Toll, GitHub, *MyWebServer*, Sweden, Oktober 2014 Available: `https://github.com/dntoll/MyWebServer`