Salesmen (Group2 SE)

Software Test Document

author(s): Patrick Provinciael

Abstract: This document details the various tests and testing procedures required for the Salesmen project. This project is created by group 2 of the course Software Engineering in the 2009-2010 study year at the Vrije Universiteit Brussel.

14-04-2010

History of versions

Version	Date	Author(s)	Changes
v0.0.1	29-11-2009	P. Provinciael	First Draft
v0.0.8	06-12-2009	P. Provinciael	Draft before QA
v0.1.0	07-12-2009	P. Provinciael	QAed version
v0.1.2	11-12-2009	P. Provinciael	Addition of JUnit
v0.2.0	21-12-2009	P. Provinciael	Made the document consistent with the other documents
v0.2.9	14-04-2010	P. Provinciael	Updated Bibliography
v0.3.0	14-04-2010	P. Provinciael	Updated: Use of TestNG instead of JUnit, Required Unit Tests

Contents

Chapter 1

Introduction

1.1 Scope

The purpose of this document is to describe the Software Test Plan(STD) for the Salesmen Project[8]. It will outline the various testing mechanisms and the responsibilities of these testing mechanisms.

Chapter 2 outlines the testing plan, while chapter 3 details the the integration tests. The results of the various integration tests will be included in this document.

1.2 References

See Appendix A.

1.3 Definitions and acronyms

- QAM: Quality Assurance Manager
- SQAP: Software Quality Assurance Plan
- SCMP: Software Configuration Management Plan
- SPMP: Software Project Management Plan
- SDD: Software Design Document
- SRS: Software Requirements Specification
- STD: Software Test Document
- JUnit: Java Unit testing Framework

Chapter 2

Test plan

2.1 Introduction

The purpose of this chapter is to outline the various testing mechanisms and the responsibilities of these mechanisms.

2.2 Test items

The following items must be tested.

- All methods and classes as outlined in the SDD[3]
- All requirements of the system as detailed in the SRS[4]
- The quality requirements as detailed in the the SQAP[5]
- Every piece of code, part of the completed system.

2.3 Features to be tested

Every feature implemented in the final completed system must be tested. These must be tested as soon as possible and whenever changes are made.

2.4 Features not to be tested

Every feature which is not (yet) implemented.

2.5 Approach

Several methods will be used to test the various features.

2.5.1 Kind of tests

The following tests will be run.

- Unit testing of all business logic with TestNG[7].

- internal testing of the various tools used. This will be done with the testing tools provided by these tools. When no testing facilities are provided, testing of these tools will be done within the integration tests.
- Integration tests.

2.5.2 Unit tests

The QAM will foresee the possibility of writing Unit tests. Each programmer is responsible for writing these tests. These unit tests will be created to run with the TestNG[7] testing framework.

Whenever changes are made, the QAM will run all Unit Tests. The programmers are encouraged to run the full Unit Testing suite as well.

Unit tests are required on all code, excluding:

- Variables
- Getters for variables.
- Setters for variables, you can still use these in order to change the contents of your class to suit the given test.

Every Unit Test of a given class must contain a line that checks if the class can actually be initialized. Then for every non-trivial method, the following tests must be written:

- Two tests that have input which gives a normal expected result.
- Any number of inputs that could give a troubling output, but shouldn't. These can be non-existent, but try to think about it.
- As many possible inputs that give an error in the code of that specific method. Don't write tests to check for crashes in procedures used by that method.
- Optional: Any other test you can come up with to check the functionality of your method.

2.5.3 Integration tests

Integration tests will be run whenever a new stable version of the system is released. The purpose of the integration tests is to ensure that the system works as a whole. Integration tests are focused on testing whether the requirements, as outlined in the SRS[4], are incorporated. More details regarding integration tests can be found in chapter 3.

2.5.4 Regression tests

Whenever a new iteration is about to start, there will first be a round of regression tests. This means that all functionality added in the previous iteration is tested to ensure that nothing was broken in the previous iteration.

2.5.5 Interface tests

It is important to assure that the User Interface is user-friendly. To ensure this fact, the following methods will be used.

- The interfaces will be developed with user-friendliness in mind.

- The interfaces will be tested by the other projectmembers. Each will evaluate if the interface was user-friendly enough. If necessary, the interface will then be adjusted and a new round of Interface testing will start.
- When all projectmembers deem the interface user-friendly, the User Interface will be tested by one or more external parties. These persons must all have different backgrounds of the target audience. This to ensure that every person will find his needs in the system.

2.6 Item pass/fail criteria

For every test, the pass criteria is the lack of errors or other problems.

2.7 Suspension criteria and resumption requirements

Testphases can be suspended in the following cases.

- When a unit test fails. If the error appeared in code that was recently written by the tester, he is encouraged to track down the error and fix this. When the error appears in code not written by the tester, an issue must be made on the Google Code Repository website[9]. In the case the test requires a minor fix, the tester may change this locally and inform the responsible programmer of the change he made. The original programmer is responsible of commiting the fix to the repository.
- When an integration test fails, an issue has to be made on the Google Code Repository Website[9]. The integration test must start over again when the issue is resolved.

2.8 Test deliverables

For every code file, there must be a test file. This test file must detail every test applicable on this code. Whenever the code is changed, the test file must be reviewed to ensure all functionality is still tested.

Every other test result will be included in an appendix of this document.

2.9 Test tasks

The unit tests are the responsibilities of the developers. It is the responsibility of the QAM to assure that these tests are complete. Integration tests will be mainly completed by the QAM, but individual projectmembers are encouraged to run their own integration tests.

2.10 Environmental needs

No additional requirements are made for the programming environment, besides those listed in the SCMP[2].

2.11 Responsibilities

The unit tests are the responsibilities of the developers. It is the responsibility of the QAM to assure that these tests are complete. It is also the responsibility of the QAM to assure that everything else is tested.

2.12 4.2.13 Staff and training needs

When needed, the QAM will explain the creation and use of the TestNG testing framework to the individual members. There will also be wiki pages on the Google Code Wiki[9] to explain matters that programmers are having difficulties with.

2.13 Schedule

Whenever anything new is added to the system, there must be tests.

Chapter 3

Overview of the integration tests

3.1 Scope

The purpose of this chapter is to document the various integration tests required. These tests will be conform with the SRS[4] and thus, they will each contain a reference to the correspondent SRS[4] Requirement. Every Requirement will be tested for a possible negative and positive result.

Appendix A

References

[1] Software Project Management Plan, Nick De Cooman, *http://wilma.vub.ac.be/~se2_0910/docs/spmp/index.html*, October 26, 2009.

[2] Software Configuration Management Plan, Jorne Laton & Sina Khakbaz Heshmati, *http://wilma.vub.ac.be/~se2_0910/docs/scmp/index.html*, October 8, 2009.

[3] Software Design Document, Bart Maes, *http://wilma.vub.ac.be/~se2_0910/docs/sdd/index.html*, November 30, 2009.

[4] Software Requirements Specification, Wouter Van Rossem, *http://wilma.vub.ac.be/~se2_0910/docs/srs/index.html*, November 12, 2009.

[5] Software Quality Assurance Plan , Patrick Provinciael, *http://wilma.vub.ac.be/~se2_0910/docs/sqap/index.html* November 22, 2009.

[6] Software Testing Document , Patrick Provinciael, *http://wilma.vub.ac.be/~se2_0910/docs/std/index.html* December 06, 2009.

[7] TestNG Unit Testing Framework, `http://testng.org/`

[8] Salesmen Project Website, Salesmen, *http://wilma.vub.ac.be/~se2_0910/*, Nov, 2009.

[9] Code Repository, Salesmen, *http://code.google.com/p/salesmen*, Nov, 2009.

[10] Slides Software Engineering, Dirk Vermeir, *http://tinf2.vub.ac.be/~dvermeir/courses/software_engineering/slides.pdf*, October 1, 2009.