Test Document

Potkuri-group

Helsinki December 12, 2008

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Veera Hoppula
Mikko Kuusinen
Jesse Paakkari
Tobias Rask
Timo Tonteri
Eero Vehmanen

Client

Valentin Polishchuk

Project Masters

Sampo Lehtinen

Homepage

`http://www.cs.helsinki.fi/group/potkuri`

Change Log

Version	Date	Modifications
1.0	11.12.2008	Coverage information added
0.1	13.11.2008	1st draft

Contents

1 Introduction

1.1 Document Purpose

The purpose of this document is to describe the different tests the group Potkuri will have to perform to ensure that our software will work with the least bugs possible. Those tests can be subdivided into the categories found in section 3.

1.2 System Overview

The software is coded in Java-language and it will be tested with JUnit.

2 Vocabulary

Airport Airport is where arrival tree begins, in the middle of the map.

Arc Arcs are circles at a determined radius distance of the airport. The merge points are located into these arcs.

Arrival tree A binary tree consisting of paths. Has a root at the airport.

Checkstyle Java code review for Eclipse.

dbZ dBZ stands for decibels of Z. It is a meteorological measure of equivalent reflectivity (Z) of a radar signal reflected off a remote object.

EclEmma Java Code Coverage for Eclipse.

Flight plan Every plane has a flight plan which describes its path.

FMI Finnish Meteorological Institute.

Integration Testing Integration testing purpose is to assure that integrated classes do all those services they are planned to do in requirement document.

Java2D Display and print 2D graphics in Java programs.

JAR Runnable Java archive, which based on the ZIP file format.

JUnit JUnit testing framework.

Map A map from somewhere in the world used in this product.

Merge point A point on the map where two paths merge into one path.

nmi nautical mile (=1,8520km)

Path A route to the airport that should avoid storms.

PGM Portable Gray Map, a graphics file format.

Plane An airplane that tries to land at an airport along a path avoiding storms.

PMD Java code review for Eclipse.

Storm A set of pixels with a dBZ-value over a certain threshold (that is a parameter) close each other on the map. Indicated with red color on the map.

System testing System testing purpose is to assure that software corresponds it?s requirements.

User A person using the product to watch animations on aircrafts landing at an airport in presence of hazardous weather systems.

Unit testing Unit testing purpose is to assure that certain class or unit do all those services it is planned to do in requirement document.

3 Testing

The goal is to prove that software is corresponding it's definitions (verification) and it contains all those services the requirement- document specify (validation).

Testing is made within development.

3.1 Testing Tools

Unit testing and integration testing is made by JUnit -program. The scope is observed with EclEmma, which measures JUnit- tests clause scope.

4 Unit Testing

Unit testing purpose is to assure that certain class or unit do all those services it is planned to do in requirement document.

4.1 Approach

Every Unit has own Test Report. Test Report -model is attached to this document. There is described

1. Test Plan
- How the testing will be done
- Who will do it
- What will be tested
- How long it will take
- What the test coverage will be
2. Test Inputs and expected results.
4. Test Incident Report.
6. Test Summary Report.

A Unit Test is to be created for each class by the person who wrote that class. Tests can then later be used to check whether all units still works after changes in code.

Black-box Testing assures that Units services corresponds to its definition. There will be several JUnit -tests, that constitute the Test package.

Stubs can be used to assist testing if a module is conditional on some other class.

White-box testing uses an internal perspective of the system to design test cases based on internal structure.

4.2 Coverage

Usually Unit test coverage is measured by parts of code are executed during running tests. Testing aim to 100% scope in prime classes. In the aggregate level the Unit testing coverage must be over 80%.

4.3 Selecting Test Inputs

Values for Test Inputs are selected from different value range.

4.4 Item Pass/Fail Criteria

Unit tests are written so that the result either pass or fail. Test module is approved, when all tests passed.

5 Integration Testing

Integration testing purpose is to assure that integrated classes do all those services they are planned to do in requirement document.

Integration testing concentrates to ensure that classes cooperation works correctly. Modules are combined and tested as a group. There shouldn't be bugs on interfaces after Unit Testing, so units are exercised through their interfaces using black box testing.

5.1 Approach

The approach is selected case-specific. There are few types in structural Integration Testing:

- Top Down testing.
- Bottom Up testing

The top-down approach to integration testing requires the highest-level modules be tested and integrated first. This requires that all modules are almost ready.

The bottom-up approach requires the lowest-level units be tested and integrated first. By using this approach, the need for stubs is minimum.

5.2 Interface testing

Every module group has own test -package which test groups interfaces and the analogy between interface and interface's definition. There will be also test-report where is specify which interfaces belongs to ensemble.

5.3 Coverage

Tests must coverage all the services defined in test-report.

5.4 Selecting Test Inputs

Test Inputs are selected on the grounds of design, and values are from different range.

5.5 Item Pass/Fail Criteria

Integration tests are written so that the result either pass or fail. Tested group is approved, when all tests passed.

6 System Testing

System testing purpose is to assure that software corresponds it's requirements.

Every member of group must attend to System test- event. Then group members test all use cases and go through non functional requirements, quality requirements and confines.

Group will then write test-report, which reveal tested requirements and results. Requirement document's non-functional requirements, quality requirements and confines are validated or justify, if requirement can't be validate.

6.1 Approach

Functional requirements are tested through user interface. Test use cases are defined in test-plan.

6.2 Coverage

Tests must coverage all the usecases defined in requirement document.

6.3 Selecting Test Inputs

Test Inputs are selected on the grounds of Unit testing and Integration testing.

6.4 Item Pass/Fail Criteria

System testing is approved, when all use cases are tested and passed.

7 Managing the Plan

7.1 Timing

- Unit testing and Integration testing will be done at the same time with implementation. The schedule is set up in test- plan.

- System testing event is arranged 28th November 2008.

- System testing is ready 1th December 2008.

7.2 Responsibilities

Unit Test is to be created for each class by the person who wrote that class. Integration testing and System testing are group's common project.

7.3 In Summary

JUnit testing coverage was about 92% in the end of project. This value is measured with EclEmma. These tests are included in source code under test -package.

Unit tests and integration tests are commented in Java code. System testing is documented in Test Plan -document.