Middle East Technical University

Department of Computer Engineering

Computer Engineering Design 2

Spring 2012

Test Specification Report for PATİKA Project

Striders of the Modern World

Levent Oral

Pınar Yılmazer

Bahar Şevket

Gözde Özcel

Table of Contents

1. Introduction

The Patika project is an Android application which contains map rendering with OpenGL ES 2.0 and path finding algorithms to find shortest path between multiple points. The Patika project also contains a user-friendly GUI to make usage simpler for end users who are military personals. In the Patika project GUI, map rendering with OpenGL ES 2.0 and path finding algorithms are integrated in deeply. So developing and testing a consistent application is tidies work which has to be done carefully and systematically. To accomplish testing correctly, a testing procedure must be followed. So, TSR document is prepared to determine testing procedure. After introduction part, information about testing strategy is given and project management issues are described. Next, detailed test procedure is explained with test tactics and test cases. Afterwards, testing resources are given, test work products are identified, result evaluation and logging are explained and organization mechanism is determined. Finally, test schedule is described in details.

1.1. Goals and Objective

The Patika project's three main modules, GUI, map rendering and path finding, are complicated and are need lots of time and effort to develop. Moreover, maintaining each part and integration between them is very important and hard work while developing independently.

After testing procedure, result will be almost bug-free and consistent application. Integration and correctness of parts will be almost done. Some points will gain extra concern during testing, which are:

- Each module behaves correctly, especially main components (unit testing)
- Modules' integration is done correctly, especially within main components (integration testing)
- Requirement testing, main requirement by main components (validation test)
- Over all test (higher-order testing)

1.2. Statement of Scope

In this document, a brief description is given about the testing procedure of the Patika Project. The testing procedure can be found in this TSR document with some major points which are:

- Tested software in details,
- Strategy of testing in terms of unit testing, integration testing, validation and higher-order testing,
- Testing tool and environment,
- Testing constraints,
- Logging and evaluating of testing,
- Handling of errors,
- Responsibilities of each members,
- Testing schedule.

1.3. Major Constraints

The major constraints are listed here:

- Time: In Patika project, developing and testing are tried to accomplish in limited time. Thus, testing time of each process affects result.
- DTED Data: In Patika project, DTED data is used for map rendering. Availability of sufficient and variant type of DTED data affect testing result.
- Hardware: In Patika project, Android platform is used for development which restricts developers in testing with optionality.
- Number of People: In SMW team, there are 4 members who serve the project not only for development purposes but also testing purposes in addition to their extra and unrelated workload. Addition to member count and increasing interest time by members to the Patika project affects result.

1.4. Definitions, Acronyms and Abbreviations

- DTED: Digital Terrain Elevation Data
- SMW: Striders of Modern World
- TSR : Test Specification Report

1.5. References

- Pressman, Roger S. Software Engineering: A Practitioner's Approach, Sixth Edition. New York, NY: McGraw-Hill
- IEEE Standard for Software Test Documentation

2. Test Plan

2.1. Software to be tested

PATIKA project has 6 main components, namely: DataModel, Engine, Factories, GUI, Connection and Pathfinder. DataModel package is used for constructing main objects such as vehicles, paths and points. Therefore, it must be tested to verify if all objects are generated correctly or not. Connection package will be tested to verify connection between command device and mission device. Besides, GUI package and Pathfinder package must be tested regularly in order to achieve project needs and get correct flows on device screen.

2.2. Testing strategy

In this document, whole testing process of Patika is described. We will test all of the components mentioned above thoroughly during the development process. After successfully completing unit testing phase, we will combine individual software modules to each other and test as a group. Then, we will conduct validation testing in order to check that system or product meets the requirements and specifications. Lastly, we will apply system testing to whole integrated system in order to ensure its functionality.

2.2.1. Unit testing

All main modules will be tested using unit testing in our project. Firstly, unit is chosen to be an individual function or procedure in one module. After correctly testing all functions, unit is chosen to test entire class or module. Because, some functions and modules need other components to work properly, we will ignore these components temporarily. Unit test will be constructed according to this restriction. Unit testing is a very important part of the testing phase because it enables us to see incomplete and incorrect sides of the each individual part of the project.

2.2.2. Integration testing

Even if each component works properly individually, they may lead to problems during integrating phase. For this reason, we need to test them one more time after integration operation. In our project, after finishing integration of DataModel component with Engine and Factory components, entire component need to be tested in order to verify correct intermediate data creation and data flow. Besides, pathfinder component needs to be integrated with GUI and other components and must be tested to see if correct path is shown on the device or not. Lastly, connection module must be integrated with previously combined components. Integration testing is another important phase of the testing such that we can check if modules can work together and if they are synchronized properly.

3. Test Procedure

3.1. Unit Test Cases

3.1.1. Data Model

This unit testing includes reading of XML data and constructing data classes for vehicles, points, paths… When testing this unit, we will supply various XML files to our system and then check whether the classes generated from them have any discrepancies with the original data.

3.1.2. Factories

This unit testing includes creation of relevant data structures from data model package. This gives us the required structures to display vehicles, map and paths on GUI. When testing this unit, at first, we will check the displayed vehicles with the given XML's. Secondly, we will check the data structure formed for map representation with the data file of that map and check for errors. The results are not error tolerant since it directly affects GUI.

3.1.3. Pathfinder

This unit testing includes finding of point to point paths, point to multiple paths and multiple to multiple paths. When testing this unit, at first, we will try to determine that whether the found point to point path is actually the shortest one. After verifying the

correctness of this part, we will need to integrate this part to the point to multiple path finding. After performing integration test to that bigger module, we will take this bigger module as our new unit and perform the same tests we performed on point to point algorithm. This process will repeat itself for the multiple to multiple point paths finding also.

3.1.4. GUI

This unit testing includes the response testing of the GUI. Since this part is a little tricky to wholly test, we will determine some extreme cases such as:

- Clicking on the screen many times
- Leaving the application open for a long time
- Trying to choose and place vehicles repeatedly many times
- Putting the device to sleep and then waking it.

During these tests, we will try to determine the functional and visual correctness of the screens.

3.1.5. Connection

This unit testing includes the testing of establishing a connection between the commanding device and the mission device. When testing, we will try to establish connection between several devices in various distances. We will see that whether we have connection timeouts, data loss or corruption, or not.

3.2. Integration Testing

All of the modules are strongly connected to each other in this project. So, as soon as a new interaction or a new object is defined between modules, an integration test is performed to check correctness of the communication. Therefore, this procedure lasts as long as the implementation continues.

3.3. Validation Testing

Validation testing is the testing procedure that checks compatibility between implementation and the other previous reports.

- Is the data design consistent with the reports?
- Is the architecture design consistent with the reports?
- Does the interaction between modules work correctly?

- Are the objects recognized in the way that it's stated in the reports?
- Are the objects labeled in the way that it's stated in the reports?

These are some of the test cases to be used to check compatibility.

3.4. High Order Testing

3.4.1 Stress Tests

The test cases to be used are:

- How big a DTED2 data can we represent on the screen as a 3D map
- How many mission devices can be connected to the commanding device
- Number of points to find a shortest path to and from

3.4.2. Performance Tests

The test cases to be used are:

- Does the project work in real time?
- How long will it take to visualize the map?
- How long will it take to find shortest paths?
- How long will it take to prepare the simulation screen?
- What will be the performance differences between various Android devices?

3.4.3. Compatibility Tests

The test cases are:

- Will our program give us a reasonable performance on different Android devices?
- Will our GUI design be compatible for various Android devices?

3.4.4. GUI Tests

In addition to tests performed in unit testing phase, we will test the ability to apply different scenarios mentioned in the Software Requirements Report on our GUI.

4. Testing Resources and Staffing

Testing resource is the members of SMW team. Unit tests will be made by the developers of corresponding modules. Integration test will be practiced by the developers of modules to be integrated. Validation testing will be conducted by random group members, because we think that for validation testing it is important that tester should be a person who did not implement that module. As a result, it can be said that each of the group members will take responsibilities in testing phase.

- Unit Testing

  Path Finding Unit: Pınar Yılmazer, Gözde Özcel

  Visualization Unit: Bahar Şevket

  Mobile Application Unit: Levent Oral, Gözde Özcel, Bahar Şevket, Pınar Yılmazer

- Integration Test:

  Integrate path finding with visualization: Bahar Şevket, Pınar Yılmazer

  Integrate mobile Application with visualization: Gözde Özcel, Bahar Şevket, Levent Oral

- Validation Test: Levent Oral, Gözde Özcel
- High-Order Testing: Levent Oral, Pınar Yılmazer

5. Test Work Products

We are going to use TRAC as test work product to test our system. We aim to identify the bugs in our system in testing phase and to fix them. Therefore, when a team member finds a bug in the system, he or she will assign a correction to the developer of the corresponding module using TRAC. It is necessary that each of our group members to check TRAC on daily basis, in order to maintain the testing and bug fixing phase.

6. Test Record Keeping and Test Log

Eclipse Android debugger is a plug-in for Eclipse which is the best choice for us to use for debugging our project. This debugger includes all necessary tools that keep records and logs.

7. Organization and Responsibilities

- Unit Testing

  Path Finding Unit: Pınar Yılmazer, Gözde Özcel

  Visualization Unit: Bahar Şevket

  Mobile Application Unit: Levent Oral, Gözde Özcel, Bahar Şevket, Pınar Yılmazer

- Integration Test: Bahar Şevket, Pınar Yılmazer
- Validation Test: Levent Oral, Gözde Özcel
- High-Order Testing: Levent Oral, Pınar Yılmazer

8. Test Schedule

Unit Testing May 7 – May 13
Integration Testing May 14 – May 20
Validation Testing May 21 – May 27
System Testing May 28 – June 12