KAROSHI

24.04.2011

Test Specifications Report

İlker Argın – 1559897
Meryem Sağcan – 1560499
İsmail Can Coşkuner – 1560036
Fatma Akıncı - 1630540

Table of Contents

1. Introduction

MAP-MET is a map application which has a context aware user interface. This

context awareness makes application more easy to use in where movement and light conditions are not optimal. This application is designed to work on mobile devices which use Android operation system. It is obvious that this system, given its goals, needs carefully chosen and adequately run testing methodologies.

1.1. Goals and objectives

Since MAP-MET employs an agile approach, MAP-MET gets developed with a very fast pace. New features are added and code gets refactored all the time with continuous integration. The speed of development combined with the small group size, which leads to each member having a wide area of responsibility, makes efficient testing an important necessity.

Because of our constraints, we can't aim to solve all the bugs. We use data driven development and exception driven development to fix the problems in order of scale and severity. The objectives of our testing process is to keep MAP-MET functional at all times, and to effectively utilize our resources to keep balance between fixing bugs and doing new development.

We will have several different concerns while testing, such as:

- the independent modules behaving correctly (unit testing)
- the independent modules (and external entities) communicating correctly with each other (integration testing)
- all requirements are met for a satisfactory end product (validation testing)
- testing all pieces together (higher-order testing)

1.2. Statement of scope

This document has been prepared by MAP-MET developers who are the group members of Karoshi team. The document covers the descriptions of what is used in testing process. It represents different testing strategies for the project. Procedures to be followed while testing are also included in the document. Even though each module of the product is tested after the implementation of it is completed, there is an obligation that the modules will be further tested when the whole system implementation is finished.

1.3. Major constraints

1.3.1. Time

We had only a few months on this semester, which is mostly occupied by

implementation of the project. Our remaining time will be also mostly occupied with implementation. So we will do implementation and testing in parallel. Whenever a module is completed, or some modules are integrated, required unit tests and integration tests will be done immediately.

1.3.2. Data

Since MAP-MET is composed of many modules running concurrently and passing data to each other minimizing the amount of data being transferred between modules is a goal in the sense of data constraint. For instance, the network module should be sending minimal data to the other participant for better performance of networking.

1.3.3. Hardware

Since our project is mobile application and need to run on Android device, our lack of access to hardware rigorously restricts the testing options for our project.

1.3.4. Staff

Our team consists of only 4 people each of which has a lot to do in terms of project implementation and academic work which is not related the project. In addition, each module should be reviewed by someone else than the one who developed it. We will use our human resources carefully for testing, beginning from testing first the most critical ones.

1.4. References

Standards and guidelines used in the preparation of this document are from the following 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

We have 3 major components, namely: Vision part, Messaging part and Mobile part. All of these components will be tested thoroughly. Their interactions will also be tested to ensure robustness of the system as a whole.

2.2. Testing strategy

2.2.1. Unit testing

Movement extraction, light information extraction, networking, and the map application are software to be tested in testing phase. Whereas some modules of the project can be test as individual parts, some parts cannot be tested individually. Therefore, other modules are going to be ignored during unit testing a module.

2.2.2. Integration testing

Although each module might be working individually, integrating modules together might cause problems. Moreover, there are some modules that cannot be tested as an individual module, such as networking. For example, if we did not integrate networking part and the mobile part, we cannot test whether the message common from command center is received or not. As a result, integration testing is going to check if modules can work together and if they are synchronized properly.

2.2.3. Validation testing

According to IEEE Standard Glossary of Software Engineering Terminology, Software Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. The functional and non-functional requirements of the Map-Met project are specified in our Software Requirements Specification Document. In this testing phase, our project will be tested as a whole according to whether it provides all requirements in the SRS Document.

2.2.4. High-order testing

High-order testing phase includes some testing that is run just after Integration tests. Tests which are used in this phase are stress testing, security testing, performance testing and alpha/beta testing. After integration testing, the software modules are validated to communicate with each other correctly at least as described within those tests. High-order testing, in fact, is conducted on a complete, integrated system to see if the integrated modules comply with the specified requirements. Its goal is to detect any errors within the software modules, connections between modules and also the system as a whole.

2.3. Test metrics

We will use the following metrics:

- Number Of Test Cases Executed
- Number Of Bugs Detected
- Number Of Bugs Fixed
- Number Of Priority Bugs Fixed

2.4. Testing tools and environment

Firstly, we conduct our test on our personal computer with the help of Eclipse and Android Emulator. After all tests are finished and bugs are fixed, we will deploy the application on HTC mobile device which has Android OS. In case of bugs are found in this step, we will turn back to the previous step.

3. Test Procedure

3.1. Unit test cases

3.1.1. Image Processing Unit

This unit testing includes motion and light information extraction. When testing this unit, at first, sample images, which were previously prepared, will be given as input to this module. If this test ends with success, then the unit will be tested with continuously coming frames. The results of the first test and the second one will be compared to be sure that light and motion information can be extracted from the continuously coming frames successfully.

3.1.2. Network and Messaging Unit

First, we try to send simple string messages between server and the client. This module is to be tested if it sends and receives the packets to/from the server/client machine correctly and of course again quickly. It should be fast enough that the client can send frame data in a reasonable time interval. The result returning from server should not be corrupted since wrong light and movement data may end up with inconsistent changes on user interface.

3.1.3. Mobile Application Unit

In this project static and dynamic graphical user interfaces are used. For static user interface, we test whether the buttons, information bar, map, zoom in/out, brightness, revealing enemy alias and place names work properly. After these tests are conducted, we will test dynamic interface step by step. For dynamic interface we test color management of the screen, change of user interface and font sizes, hiding

details.

3.2. Integration testing

First we will test integration of user interface module and context manager module. We will apply different scenarios to see whether the choices of user and adjustments of context manager conflicts. Then, we will integrate server side and client side applications using networking. We will define a set of messages to test the messaging part of the project and measure the speed of total operations that consists of network and image processing.

3.3. Validation testing

1. What if user stands still and shakes the phone?
2. What if user is moving and camera is recording another moving object?
3. What if user is sitting in a car which is moving and user is recording outside of the car?

Expectations:

1. Since phone is moving, we expect a change in user interface.
2. Since we extract movement from all objects on the frame, movement of one object does not have any effect.
3. There can be changes that we do not desire, but there is nothing we can do.

3.4. High-order testing

We will deploy client application to a mobile device. Then we start our server, and start application which presents on the mobile device. Then we will test it under different environment conditions with different velocity values.

4. Testing Resources and Staffing

Responsibility Staff

Testing resource is the members of Karoshi team. All group members are somehow responsible for testing and therefore everyone in the group contributes to the testing process of the product modules.

- Unit Testing: İlker Argın, İsmail Can Coşkuner
- Integration Test: İsmail Can Coşkuner, Meryem Sağcan

- Validation Test: Fatma Akıncı, İlker Argın
- High-Order Testing: Meryem Sağcan, Fatma Akıncı

5. Test Work Products

The Result of the testing process is the identification of the bugs. When a team member finds a bug, he will assign the correction to the developer of the corresponding module using TRAC. So we automatically keep records of the tests and corrections.

6. Test Record Keeping and Test Log

For debugging purposes, we plan to use Eclipse Android debugger which is a plug-in for Eclipse. The debugger includes necessary user interfaces therefore we do not need any record keeping and logging.

7. Organization and Responsibilities

- Unit Testing
    - Image Processing Unit: İsmail Can Coşkuner, Meryem Sağcan
    - Networking and Messaging Unit: İlker Argın, İsmail Can Coşkuner
    - Mobile Application Unit: Fatma Akıncı, Meryem Sağcan
- Integration Test: İsmail Can Coşkuner, Meryem Sağcan
- Validation Test: Fatma Akıncı, İlker Argın
- High-Order Testing: Meryem Sağcan, Fatma Akıncı

8. Test Schedule

Test Name	Deadline
Image Processing Unit	05.05.2011
Networking- Messaging Unit	08.05.2011
Mobile Application Unit	12.05.2011
Integration Test	15.05.2011
Validation Test	25.05.2011
High – Order Testing	29.05.2011