TEST SPECIFICATION REPORT

Group Name: Babylon

Group Members:

M. Tuğberk İşyapar
Arif S. Özçelik
Okan T. Tursun
Ö. Anıl Türkkan

26, April 2010
TABLE OF CONTENTS

1 INTRODUCTION ................................................................................................................. 3
  1.1 PURPOSE OF DOCUMENT .............................................................................................. 3
  1.2 SCOPE OF DOCUMENT .................................................................................................... 3
  1.3 STATEMENT OF TESTING PLAN REPORT ......................................................................... 3
  1.4 MAJOR CONSTRAINTS ......................................................................................................... 4
    1.4.1 Time .......................................................................................................................... 4
    1.4.2 Hardware .................................................................................................................. 4
    1.4.3 Software ................................................................................................................... 4
    1.4.4 Staff ......................................................................................................................... 4

2 TESTING SPECIFICATION .................................................................................................... 5
  2.1 UNIT TESTING .................................................................................................................. 5
  2.2 HARDWARE TESTING ....................................................................................................... 5
  2.3 SOFTWARE TESTING ....................................................................................................... 5
  2.4 INTEGRATION TESTING .................................................................................................... 5

3 TEST PLANS .......................................................................................................................... 5
  3.1 TEST PLAN FOR NETWORK COMMUNICATION AND MAP PACKAGES ......................... 5
    3.1.1 Description of module .............................................................................................. 5
    3.1.2 Testing Strategy and Test Tools ................................................................................. 6
      3.1.2.1 Static code analysis tools ................................................................................... 6
      3.1.2.2 Dynamic analysis ............................................................................................ 7
      3.1.2.2.1 Unit tests .................................................................................................... 7
    3.1.3 Test Cases .................................................................................................................. 7
    3.2 TEST PLAN FOR GUI ..................................................................................................... 8
    3.3 TEST PLAN FOR CONFIGURATION PACKAGE ............................................................. 17
    3.4 SYSTEM TESTING PLAN ............................................................................................... 17

4 TEST STAFFING AND SCHEDULE ..................................................................................... 18

TABLE OF FIGURES

FIGURE 1 BNET MAIN WINDOW GUI ...................................................................................... 10
FIGURE 2 CONNECTION MENU ............................................................................................ 11
FIGURE 3 CONNECTION DIALOG ........................................................................................ 11
FIGURE 4 CONFIGURE MENU ............................................................................................ 11
FIGURE 5 CONFIGURE DIALOG ........................................................................................ 11
FIGURE 6 MAP MENU ............................................................................................................ 12
FIGURE 7 FILE DIALOG ......................................................................................................... 12
FIGURE 8 LAYERS MENU ...................................................................................................... 12
FIGURE 9 FIND MENU ........................................................................................................... 13
FIGURE 10 HELP MENU ....................................................................................................... 13
FIGURE 11 PANELS ................................................................................................................ 14
FIGURE 12 LAYERS DIALOG ................................................................................................ 15
1 Introduction

The goal of writing this document is to describe a systematic approach to testing our senior year software project, Hammur’Abi Vehicular Adhoc Network.

1.1 Purpose of Document

As a senior year team, we propose a vehicular ad hoc network software that improves the quality of daily traffic flow. The testing process has a lot of goals. We will especially test our software for coding bugs and logic errors. Also, as we will be using wireless ZigBee communication modules, we will be testing the software’s communication performance and coordination with these modules in addition to a network simulator called GrooveNet.

Testing is very important for a software project, as it improves the quality of it and ensures that the product designed meets the requirements expected of it.

1.2 Scope of Document

The document describes,

• Testing strategies like static code analysis and dynamic code analysis.
• Test cases for certain packages.
• Test plan and testing scenarios for the GUI.
• Testing the restoration of previous settings and configuration once the system is booted.
• Test success and failure criteria.
• The test staffing and schedule.

1.3 Statement of Testing Plan Report

We have been and will be applying the following strategies:

• Testing each function separately, Unit Test
• After unit testing, combining different functional components to test them together, *Integration Testing*
• Comparing outputs received from software components with expected outputs, *Blackbox Testing*.
• *Whitebox Testing* for testing various GUI functionalities.

### 1.4 Major Constraints

#### 1.4.1 Time

Time is probably the hardest constraint to satisfy. The team is at the stage of implementing the project and as we approach the deadline of our project, it will be hard to spare time for testing and debugging.

#### 1.4.2 Hardware

The only hardware testing we will be done while establishing communication between the wireless devices and the software. Time needed to adapt these devices to our system will consume some time that can be used for testing and debugging our software components.

#### 1.4.3 Software

We will be using the testing functionalities of Integrated Development Environment’s like Eclipse and NetBeans.

#### 1.4.4 Staff

Team members are working on different areas of development. Thus it will be hard to reorganize the team to do both testing and development at the same time.
2 Testing Specification

2.1 Unit Testing

Every class and their methods are implemented and reviewed by more than one project member. Thus, unit testing is a part of the development process, therefore individual unit testing of the project is not required.

2.2 Hardware Testing

The project consists of three major hardware components; the laptop the system works on, the GPS device and the ZigBee device. All these three devices are customer end products and thus they are not required to be tested at the scope of the project.

2.3 Software Testing

There are two different software present is this project. First one is the user interface (the main program) and the simulator.

Software testing will be detailed at the following chapters.

2.4 Integration testing

During integration testing, there are four scenarios. They are “the main program with the simulator”, “the main program with GPS device”, “the main program with the ZigBee device” and “the main program with the GPS and ZigBee” device.

At all the scenarios, the system must be functional. When a device is not available, the simulator must be able to simulate the missing device or devices.

3 Test Plans

3.1 Test Plan for Network Communication and Map Packages

3.1.1 Description of module

The \texttt{comm} package contains classes that are used for network communication. The \texttt{map} package consists of classes that store and manage entities such as other vehicles
in the map. Map package is responsible for data structural management of entities. For instance providing necessary java containers, providing interface for adding/removing entities etc. Graphical representation of entities is not responsibility of map package. gui package is in charge of graphical representations; hence, no GUI testing procedure is involved in testing of map package.

3.1.2 Testing Strategy and Test Tools

The methods used in testing comm and map packages are classified in two categories in literature:

- **Static code analysis**: Test methods which are performed without executing the program, often performed on source code and sometimes on object code.
- **Dynamic analysis**: Methods which are performed by executing programs giving sample outputs and comparing outputs against a set of expected values.

3.1.2.1 Static code analysis tools

The following tools are used in static code analysis. All of them provide plug-ins for Eclipse IDE. The reason of using multiple code checkers is to attain a finer degree of bug proofing since each tool uses a different method to examine the code.

**Findbugs 1.3.9**

Findbugs is a tool which looks for predefined bug patterns in Java bytecode to find bugs in source code. It comes with a predefined set of bug patterns. The complete set of bug patterns is available at web page of the project\(^1\).

**PMD 4.2.5**

PMD operates directly on Java source code. It comes with predefined rule sets and checks for suboptimal and overcomplicated code in addition to bugs. Descriptions of available rule sets are available at web page of PMD\(^2\).

---

\(^1\) [http://findbugs.sourceforge.net/bugDescriptions.html](http://findbugs.sourceforge.net/bugDescriptions.html)

Checkstyle 5.1
Checkstyle focuses on coding standards rather than finding bugs. It is used to ensure that written code is clear and well-documented with Javadoc comments. The checks performed by Checkstyle are available at Checkstyle web page.

3.1.2.2 Dynamic analysis
Although static code analysis is very effective for catching bugs, it is not sufficient to verify whether a program will run correctly. For this reason, dynamic analysis methods are included in testing procedure.

3.1.2.2.1 Unit tests
Unit testing is verification and validation method which takes units of software gives inputs from a predefined set and compares outputs against an expected output set. For unit testing of classes in communications and map packages, JUnit v4.5 testing framework is used. The tested methods are picked from the methods which constitute the software interface of the individual classes and packages.

3.1.3 Test Cases
The following test cases are used in unit testing of comm and map packages:

Comm package – Message class
1. Constructor must not return null nor throw exception.
2. hashCode() for two different messages must be different.
3. hashCode() for same message in two different objects must be equal.
4. isValid() must return true for strings returned by getFormattedMessage().
5. isValid() must return true for messages with correct number of fields.
6. equals() must return true for two Message objects with same content.

Map package – EntityObject class
1. hashCode() must be equal for two EntityObjects constructed from same message.
2. Constructor must set all fields correctly.

http://checkstyle.sourceforge.net/checks.html
3. equals() must return true for two EntityObjects constructed from same message.

**Map package – EntityManager class**

1. Constructor must not return null nor throw exception.
2. After addEntity() method, associated layer of MapGUI object must contain the added object.
3. After a random amount of time(t) is set with setTimeout(t) method, added entity must be automatically removed after t + checkPeriod msecs both from container of EntityManager object and from associated MapGUI layer (depending on type of object).

Testing of NetworkClient and NetworkSender classes in comm package requires integration with other components of system. Therefore their testing is done using Groovenet simulator rather than unit testing which is described in the next chapters.

### 3.2 Test Plan for GUI

The code for the GUI is included in the `edu.metu.ceng.bnet.gui` package and that package is imported by the main program of our project in the `edu.metu.ceng.bnet` package.

It performs the functionality of the user interface with extensive use of Openmap-4.6.5 libraries. Bugs or quality of performance issues related to Openmap-4.6.5 libraries are little or not to mention in the report. Tests for the package are going to be based on a scenario depicting the user actions taken on the GUI. Each item in the scenario is included in the order of the program flow and for each of them success and failure conditions are specified. Each of the failure conditions occurring exclusively or coexistingly under testing process may be considered as a system degradation.

#### 2.2.1 Scenario for Testing

1. The code is designed to work on a PC with a Linux OS. The GUI window shall appear with the settings formed by the user at the time of the system boot assuming no
system failure. The operating system GUI will not be visible to the user. For the first use of the BNET system, GUI will be initialized according to factory settings.

**Success condition:** The GUI is initialized with complete components which are described in the following item of the scenario. While the components are loading, a window depicting the status of initialization process shall appear. Following that, main window shall be visible.

**Failure conditions:**

The GUI is not initialized due to bugs in the code at the time of the system boot.

The status window appearing before the main window does not work properly. Transition between the two windows never occurs.

2. The main window shall include menus and sub-menus, panels, the map and graphic images of the agents of the BNET system which are varying over the locations on the map and other information added as layers to the map, and an information bar. The correct behaviors of these components are explained in the following sub-items.

**Success condition:** The main window shall appear with all the components correctly and completely included.

The BNET GUI main window with all components included appears as the following figure.
Failure condition: If any component fails to appear correctly, this will be considered a system degrade.

2.a. The menus shall appear on the top of the main window of the GUI following each other. The menus are Connection, Configure, Map, Layers, Find, and Help.

Success conditions:

Connection menu, which is depicted in Figure 2, includes Connect to network... sub-menu. By left clicking this, the user shall see a connection dialog, which is depicted in Figure 3, asking him/her to provide hostname or ip, and port number to connect to a network. By clicking the Connect button on the dialog, the user shall connect to a network or see why he/she is not able to connect. By clicking the Cancel button, user shall be able to cancel his/her connection request.
Configure menu, which is depicted in Figure 4, includes Validate yourself... sub-menu. By left clicking this, the user shall see a configuration dialog, which is depicted in Figure 5, asking him/her to provide information about the type of the agent he/she represents in the system, and destination and extra information optionally. After the user clicked on the Configure button, other users of the system shall see the graphical image depicting the agent the user validated on their maps. By clicking the Cancel button, user shall be able to cancel this process.

Map menu, which is depicted in Figure 6, includes Center Myself and Lock checkbox and Add shp file as new layer... sub-menu. Provided that Center Myself and Lock checkbox is checked, the user shall the graphic image of the agent he/she
represents at the center of the map at all times. Centering of the map must be repeated every time the projection changes due to changing locations of the user or any other reason. Whenever Center Myself and Lock checkbox is not checked, the user shall be free to see any location on the map. By left clicking the Add shp file as new layer... sub-menu, a file chooser, which is depicted in Figure 7, asking the user to select a shape file residing on the disk of the PC appears. The user shall be able to choose a shape file and by clicking the Open button, the chosen file must be added as a new layer to the map. By clicking the Cancel button, user shall be able to cancel this process.

![Figure 6 Map menu](image1.png)

![Figure 7 File dialog](image2.png)

Layers menu includes all layers included in the system with checkboxes. Visible layers are checked. By unchecking any of the layers, the user shall be able to not see them on the map. At the bottom of the Layers menu, the user shall be able to see Edit Layers... sub-menu. By left clicking this, the Layers Panel shall appear.
Figure 8 Layers menu

Find menu includes Find Nearest Hospital, Find Nearest Pharmacy, Find Nearest Police Station, Find Nearest Car Park sub-menus. By left clicking these, the user shall see the nearest specified agent centered on the map. If the Center Myself and Lock checkbox is checked, it is automatically unchecked before that.

Figure 9 Find menu

Help menu includes About sub-menu. By left clicking this, the user shall be see information about the version of the program, credits and any other information provided by the developers.

Figure 10 Help menu

Failure conditions:

- Any of the menus and sub-items does not appear.
- Any of the menus and sub-items does not function properly.
2.b. Below the line of the menus, shall appear the panels.

![Figure 11 Panels](image)

**Success conditions:**

*Navigation Panel* shall appear. By clicking on the directional arrows included in the panel user shall navigate to other locations residing on corresponding directions on the map. By clicking on the center mini-button, the map shall be centered on a point specified by the program as default center point.

*Zoom Panel* shall appear. The user shall be able to zoom in/out to see restricted areas on the map.

*Scale Panel* shall appear. The user shall be able to scale the map by providing a ratio on the panel.

*Layers Panel* shall appear. By left clicking on this, the user shall see a dialog depicting layers in the order they are painted on the map. The dialog, which is depicted in Figure 12, includes buttons to move a selected layer to the top, to the bottom, to up one or down one and a button to delete a layer. The system shall function accordingly with the user left clicking on these buttons. Each layer included in the dialog has a button to turn the layer on and off and a button to initialize the tool which sets properties and redraws the layer if available. The system shall function accordingly with the user left clicking on these buttons.
Figure 12 Layers dialog

*Mouse Modes Panel* shall appear. The user can choose select, navigation or pan mouse modes by clicking on the specified button on the panel and perform the functionality imposed by the chosen mouse mode on the map.

*Overview Map Panel* shall appear. By left clicking on the panel, the user shall see an overview map, which appears in Figure 1, depicting a larger area surrounding the actual area depicted on the map.

**Failure conditions:**

Any of the panels and sub-items does not appear.

Any of the panels and sub-items does not function properly.

**2.c.** Below the panels, shall appear the map. Basic map functionality is depicted in Figure 1, at the center of the window.

**Success conditions:**

The map shall appear with all the active layers properly added and depicted.

The map shall be redrawn at the change of projections or at intervals specified by the program to animate the agents. All layers appearing correctly before the redraw operation must appear correctly after it.
The map shall be centered to the user if he/she chooses it to behave so.

The agent of the system representing the user shall be depicted on the map as a red directional arrow on the map.

Agents who are disconnected are deleted from the map after some interval of time specified by the program.

The user shall be able to see information about other agents by moving the mouse over them in the selection mode. This information shall be displayed in a tooltip under the specified agent and in the information bar. Some information received from the surrounding agents shall be displayed in tooltips or pop-up menus around the graphic image representing the user on the map, i.e. emergency situation. The user shall be able to take necessary actions through these menus.

**Failure conditions:**

Any of the layers do not appear on the map.

Some functionality of the map do not work.

2.d. Below the map, an information bar shall be visible. Information bar appears at the bottom of Figure 1.

**Success conditions:**

User shall be able to see the location of the mouse in latitude/longitude and x/y terms in the information bar.

User shall be able to see active layers as green rectangles and see names of the layers by moving the mouse over these rectangles as tooltips.

User shall be able to see any other information on the bar.

**Failure conditions:**

The information bar does not depict correct information.

The information bar has missing components.

3. The user shall be able to exit the system by clicking the on the upper-right button.
Success conditions:

The user shall be able to disconnect from the network.

The main window and other windows shall be closed.

The active layers and other settings imposed by the user shall be logged to be used to configure the system at the next boot.

The operating system shall turn off the computer.

Failure conditions:

Any of success conditions fails.

3.3 Test Plan for Configuration Package

Configuration package is used to restore last used configuration of application on startup. This test will be the last one to be applied during our testing phase. After we are done with testing the communication and interface packages, the last main functionality of our software to be debugged will be restoring the settings used previously by the user. In order to test this module, a black box unit test will be applied. At a given state of the system we will produce an expected file where the settings will be stored and we will use this file to compare another file that will be produced directly by our System Configuration module. We hope to discover and fix any bugs by applying this method.

3.4 System testing plan

Since working with GPS and ZigBee devices is not efficient and not possible at large scale scenarios, a simulator is developed. As explained before at previous reports, GrooveNet is modified and used as a simulator for our project. GrooveNet is a tool which simulates a car network with GPSes. We modified the GrooveNet to simulate the GPS data and network (ZigBee) part of our project. First we define the number of cars and then ZigBee waits until that number of users are connected. The users are connected via the main program with different port numbers. After all users are connected, ZigBee sends own data and neighboring car data to all cars. Also network data is send from all individual cars to the ZigBee and then routed to neighbor cars. The simulator is tested
and verified to be work as intended during development phase. No more testing shall be
done to the simulator.

If a GPS or ZigBee device is made available to the project, then the system shall
be tested by integrating that product and making it take the responsibility of the
simulator. The system must work as intended when real not simulated data is available.

### 4 Test staffing and Schedule

<table>
<thead>
<tr>
<th>Module tested</th>
<th>Person in charge</th>
<th>Starting Date</th>
<th>End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network communications integration and performance tests</td>
<td>Okan, Anıl</td>
<td>18/3/2010</td>
<td>10/6/2010</td>
</tr>
<tr>
<td>Map package unit and integration tests</td>
<td>Okan</td>
<td>8/4/2010</td>
<td>10/6/2010</td>
</tr>
<tr>
<td>System Test</td>
<td>All group</td>
<td>29/4/2010</td>
<td>10/6/2010</td>
</tr>
</tbody>
</table>