CENG 492
Computer Engineering Design

TEST SPECIFICATIONS
REPORT

KÜP ŞEKER

Deniz OLGUN
Deniz TUNA
Osman Tuncer KAPLANKIRAN
Mehmet Safa ERTEKİN
# Table of Contents

1. Introduction .................................................................................................................. 3  
   1.1. Goal and Objectives ................................................................................................. 3  
   1.2. Statement of Scope .................................................................................................. 3  
   1.3. Major Constraints .................................................................................................... 4  
      1.3.1. Time .................................................................................................................. 4  
      1.3.2. Cost ................................................................................................................... 5  
      1.3.3. Staff .................................................................................................................. 5  
   1.4. Definitions, Acronyms and Abbreviations ............................................................... 5  
   1.5. References .............................................................................................................. 5  

2. Test Plan ........................................................................................................................... 6  
   2.1. Software to be tested ............................................................................................... 6  
   2.2. Testing Strategy ....................................................................................................... 6  
      2.2.1. Unit testing ......................................................................................................... 6  
      2.2.2. Integration Testing ............................................................................................. 6  
      2.2.3. Validation Testing .............................................................................................. 6  
      2.2.4. High-order testing ............................................................................................. 7  
   2.3. Test Metrics ............................................................................................................. 7  
   2.4. Testing tools and environment .................................................................................. 8  

3. Test Procedure ................................................................................................................ 8  
   3.1. Unit Test Cases ....................................................................................................... 8  
      3.1.1. Ownship Perspective ......................................................................................... 8  
      3.1.2. OpenMap Perspective ....................................................................................... 9  
      3.1.3. Repast Perspective ............................................................................................. 9  
   3.2. Integration Testing .................................................................................................. 9  
   3.3. Validation Testing .................................................................................................. 9  
   3.4. High-Order Testing ................................................................................................ 10  
      3.4.1. Recovery Testing ............................................................................................... 10  
      3.4.2. Security Testing ............................................................................................... 10  
      3.4.3. Performance Testing ........................................................................................ 11  

4. Test Resources and Staffing ......................................................................................... 11  

5. Test Work Products ........................................................................................................ 11  

6. Test Record Keeping and Test Log .............................................................................. 12  

7. Organization and Responsibilities .............................................................................. 12  

8. Test Schedule .............................................................................................................. 13  

1. Introduction

1.1 Goals and Objectives

The purpose of this document is to define and outline the Test Specifications for our Project named as ABVCSF. It specifies the methods that we use in the tests which are created for ABVCSF. As far as we know from our previous experiences we all ready for that hard part of our Project. By testing we will try to find and clear all of the bugs that can be in our system.

Our main goal in testing is to make our project tested in different parts since it will make our job a lot easier. By using that method we can much more easily find bugs and errors that can be in our system.

The objective of this process is to come up with a bugfree system that will work in every condition.

1.2. Statement of Scope

This document is created by ABVCSF’s developer team. It consists of a general test plan, testing strategies that we will implement to our system, test procedure and our testing schedule.

Generally we tested all of our projects parts as soon as we created them. However some parts of our project like integrating OpenMap and Repast S needs much more care and testing. Those will be talked in the next chapters of this document. To do this we will focus on:

- What will be our constraints
- What we will test
- How will we test each part of our system
- How to manage our timing schedule
- How team members are responsible of testing methods
We will use different methods to make this testing efficient:

- **Integration Testing**: This testing will be our major concern since our simulation environment depends on integration of 2 different platforms. If we find bugs or errors in integrated parts, they will be fixed as soon as possible.

- **System Testing**: By using this method, we will test our agents namely USV’s in our system and their behaviors. Since our simulation environment depends on intelligent agents we have to implement them correctly. If a bug or error find during testing they will be fixed by our AI developers.

- **Security Testing**: This method will not be a concern since our system does not depend on security.

- **Validation Testing**: Since our project is for simulation users we have to keep with our schedule and implementation. Validation testing will check if our project goes in a wrong or right way.

- **Stress Testing**: We will test the stability of our program by rapidly increasing the number of agents in the environment. Since users can want to play with more agents than usual, it must be tested and fixed before it is out.

- **Regression Testing**: After finding and fixing the bugs and possible errors in our project we will rerun all of those testing methods in order to be sure about fixing events.

1.3. **Major Constraints**

As Küp Şeker Project Group we have a few constraints which make this process a bit hard for us.

1.3.1. **Time**

Since we are near to end of semester, we will generally busy with implementing our project’s integrated parts, agents and other parts.
Because of this reason we have to do both implementation and testing in the same time. So doing them in parallel will be the solution of this problem.

1.3.2. Cost

Since testing is the most costly part of our project, using our time and efforts in a efficient manner will be our main constraint. Wasting too much time on worthless tests will be problem so we have to decide on more important tests before wasting our energy into others.

1.3.3. Staff

Our project team consists of 4 members and each of our members has lots of things to do about our project implementation. Since we are trying a new method to create a simulation game, we all have lots of things to be concern. In the mean time making those tests efficient and worthy needs our staff. So sharing each of those testing methods is important and man power is currently a constraint for us.

1.4. Definitions, Acronyms and Abbreviations

- ABVCSF: Agent-Based Virtual-Constructive Simulation Framework

1.5. References

2. Test Plan

2.1. Software to be tested

The software to be tested here is the OpenMap and Bridge Command simulator. OpenMap is used to simulate our agents with world coordinates. Agents are created in the java-based programme Repast. Then Repast and OpenMap are integrated to each other to simulate agents with coordinates. Therefore the main softwares to be tested are OpenMap-Repast and Bridge Command simulator like testing a game.

2.2. Testing strategy

2.2.1. Unit Testing

Agent Creation (via Repast), artificial Intelligence, user interface of Bridge Command, simulation graphics and socket modules are needed to be tested. Since other modules do not need to be tested, they will be ignored in here.

2.2.2. Integration Testing

Although each module might be working individually, integrating modules together might cause problems. Integration testing is going to check if modules can work together and if they are synchronized properly. In our project, integration test is used to test OpenMap-Repast integration and to test socket module related with Bridge Command simulator.
2.2.3. Validation Testing

Result of this test is going to show whether expectations in design is met or not. That is, modules that is tested working or processing properly. In our project firstly agent creation process is tested, then artificial intelligence of these agents is tested, after that socket module is tested in the scope of validation test.

2.2.4. High-order testing

Various system tests are going to be held such as:

**Stress Testing**: This test is going to be held to check to see behavior of the simulation to high level creation of rule based agent and to see behavior of the simulation to large amount of OpenMap data.

**Performance Testing**: Performance of the simulation is tested for different options like number of agents, intelligence level of agents or environment options in the simulation.

**Security Testing**: This is not exactly a security testing like virüs protection, in fact this security testing is to test whether one Repast collapse, second Repast is to be active in reasonable time, or not.

2.3. Test metrics

In our project main test metrics are based on the agents, taking OpenMap coordinates, simulation programme and socket module. In the unit testing part we
stated that which modules are to be tested. Since there is a time constraint on the project, each group member has to apply test procedures in efficient way for developing and debugging the project.

Staff responsibilities for testing different modules are seen below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mehmet Safa Ertekin</td>
<td>Agent Creation, Artificial intelligence Modules</td>
</tr>
<tr>
<td>Deniz Tuna</td>
<td>Agent Creation, Artificial intelligence Modules</td>
</tr>
<tr>
<td>Deniz Olgun</td>
<td>Openmap, Bridge Command, Socket Modules</td>
</tr>
<tr>
<td>Osman Tuncer Kaplankiran</td>
<td>Openmap, Bridge Command, Socket Modules</td>
</tr>
</tbody>
</table>

2.4. Testing tools and environment

There are no additional testing tools other than the simulation programme itself. However, different testing environments are required.

First environment type is the operating system. It does not matter that the operating system is Linux or Windows, the simulation programme works in both, however programme needs 64 bit operating system to work.

Second environment type is hardware. As we state in design report, simulation programme needs high performance graphical card and minimal core 2 duo inside,
that is high-tech CPU.

3. Test Procedure

3.1. Unit Test Cases

There are 3 main components of our project. They all come together for different purposes. We use their own properties and put them into our project. General aim is to make them work together and interactive with user. We will explain them in each part below.

3.1.1. Ownship Perspective

In this component of our project, we will test that if any user-selected ship from bridge command can be transferred correctly to OpenMap. It is important to see specific ownship model in user interface since it shows how we are able to implement Bridge Command ship models into our software. We implement black box testing method for these cases in order to see the result will be as expected or there is any misinterpreted concept.

3.1.2. OpenMap Perspective

In this component, we will test that if our agents in Repast and their movements can be seen on OpenMap screen. Moreover, it is crucial that the software can set correct world coordinates for our ships. User must be able to put the ship in any world location he/she wants. After that we must ensure that when simulation runs, we must be able to see location changes of all ships in the map. Black box testing method will be our first choice for this again in order to see if there is any incorrect work of our software.

3.1.3. Repast Perspective
In this component of our project, we will test that if our agents are created and they work correctly or not. In more detail, since the main feature of the software is being an agent based simulation, agents play a crucial role. They have to do some tasks according to their behaviour. In our scenario, there are 2 agents that trying to detect the ship of user by using their radars. Therefore, the software should accomplish that task accordingly. Black box method will be used for this test phase. Our agents should be able to move, evaluate what they see in radar and find user ship.

3.2. Integration Testing

In our software, there are 3 component which are mentioned above. In order to run our software properly, it is important that each component is able to send and recieve data from each other. We will test our integration for that purpose. More precisely, we will test if socket connection from Bridge Command to Repast is working to send and recieve ship coordinates in run time. We will also test if agents in Repast will show in OpenMap in their correct locations. Therefore, there will be again integration testing between Repast and OpenMap.

3.3 Validation Testing

Validation test of our software will be about testing user interface. We will test some potential situations that user can face. Remembering that user can select any location in world map, he/she should face no errors on that. Moreover, at any point in simulation, user can pause it or save it to replay later or restart all over again. These are main functionalities of our user interface, so there should not be any error on these in any condition. Moreover, there are still considered features that can be add
to user interface such as determining agent ships number and therefore difficulty of simulation. We will test for this too if it is considered to be add to project.

3.4 High-Order Testing

3.4.1. Recovery Testing

We will test the software in cases that it may collapse. In detail, we will exhaust each component of simulation and then reopen simulation. There is supposed to be no crush or corruption in simulation when reopened.

3.4.2. Security Testing

Our project dont have network connection for now. However, when it is designed to be a server and a client side there will be security considerations. In order to test that we overload the network bandwidth and lock its connection to server. Then we will see if there is any data loss of user session.

3.4.3. Performance Testing

This is one of our main constraints since we run a simulation having heavy visual components. It requires our simulation to run at a satisfied speed so that it reaches its aim. Therefore, response time of simulation should not bother user. We will test for this purpose in many computers and see if its performance is good.

4. Test Resources and Staffing

Our test resources are generally our own computers. Firstly we will run our project’s parts in our computers and test each of them. Since our system is running on windows we will use the debugger of Eclipse in order to find the bugs
and errors. We will use this resource as a bug catcher and also a tester for our agents. After we are done with testing packages, we will start testing whole project in a different computer and try to find out if we did wrong things.

All of the staff in our project group is responsible from each of those methods since we all have different assignments on different testing methods.

5. Test Work Products

After each of testing methods, there will be some outputs about our project. We will put those outputs into a log file in order to keep them in track and also to avoid from losing test datas which are really important for both time and work power.

We will get different kinds of outputs from each of testing methods that we will implement. So in order to not lose in those huge data we will put them into a log file which we will call Test Log. By this method we will keep them in track and also we will avoid those test datas from losing.

6. Test Record Keeping and Test Log

We will use SVN in order to keep our codes fresh. And thanks to SVN we can see the bugs that we find during testing events. After finding those errors we will make corresponding changes to our code and update our projects version. Also we will make a Test Log in one of our members own computer in order to keep the track of testing. By using such a technique we will get rid of losing time on same processes and same tests.

Furthermore our test log will have a chronological habit to keep track of each test. This will help us a lot during testing process.
7. Organization and Responsibilities

Our team members have got responsibilities on different testing resources.

The responsibility of Unit Testing is shared by all of our group members since we have to have a general idea what is happening in our project.

Integration testing is taken by Deniz Tuna and Mehmet Safa Ertekin. Making those tests efficient and time helper is in their responsibility.

Validating testing is under the responsibility of Tuncer Kaplankiran and Deniz Olgun. What is more, they are responsible for other tests such as stress testing.

<table>
<thead>
<tr>
<th>Responsibility</th>
<th>Staff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Testing</td>
<td>All of staff</td>
</tr>
<tr>
<td>Integration Testing</td>
<td>Deniz Tuna, Mehmet Safa Ertekin</td>
</tr>
<tr>
<td>Validation Testing</td>
<td>Tuncer Kaplankiran, Deniz Olgun</td>
</tr>
<tr>
<td>High-Order Testing</td>
<td>Tuncer Kaplankiran, Deniz Olgun</td>
</tr>
</tbody>
</table>

8. Test Schedule

<table>
<thead>
<tr>
<th>Task</th>
<th>Start Time</th>
<th>Due Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Testing</td>
<td>10.03.2011</td>
<td>10.05.2011</td>
</tr>
<tr>
<td>Integration Testing</td>
<td>10.03.2011</td>
<td>10.05.2011</td>
</tr>
<tr>
<td>Validation Testing</td>
<td>24.04.2011</td>
<td>13.06.2011</td>
</tr>
<tr>
<td>High-Order Testing</td>
<td>10.05.2011</td>
<td>13.06.2011</td>
</tr>
</tbody>
</table>