CENG 491
Computer Engineering Design
SOFTWARE REQUIREMENTS
SPECIFICATION

KÜP ŞEKER

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

1. **Introduction**.................................................................................................4
   1.1. **Problem Definition**..............................................................................4
   1.2. **Purpose**...............................................................................................4
   1.3. **Scope**..................................................................................................5
   1.4. **User and Literature Survey**.................................................................5
   1.5. **Definitions and Abbreviations**..............................................................6
   1.6. **References**...........................................................................................7
       1.6.1. **Web Sources**..................................................................................7
       1.6.2. **Book Sources**..............................................................................8
   1.7. **Overview**..............................................................................................8
2. **Overall description**.......................................................................................9
   i. **REPAST**...................................................................................................9
   ii. **OpenMap**..............................................................................................10
   iii. **Bridge Command**..................................................................................12
2.1. **Product Perspective**................................................................................13
   2.1.1. **System Interface**..............................................................................14
   2.1.2. **User Interface**..................................................................................14
   2.1.3. **Hardware Interface**..........................................................................15
   2.1.4. **Software Interface**..........................................................................15
       i. **REPAST-OpenMap Interface**...............................................................16
       ii. **REPAST-Bridge Command Interface**...............................................16
       iii. **OpenMap-Bridge Command Interface**............................................17
   2.1.5. **Memory Constraints**.......................................................................19
2.2. **Product Functions**....................................................................................19
2.3. **Constraints, Assumptions and Dependencies**.........................................20
   i. **Constraints**.............................................................................................20
   ii. **Assumptions and Dependencies**............................................................20
3. **Specific Requirements**..................................................................................20
   3.1. **Interface Requirements**.......................................................................20
3.2. Functional Requirements .............................................................. 23
  3.2.1. Creating Harbor ................................................................. 23
  3.2.2. Creating Agents ................................................................. 23
  3.2.3. Creating Player ................................................................. 23
  3.2.4. Changing Location Information ......................................... 23
  3.2.5. Transferring Positions to Bridge Command .................... 23
  3.2.6. Sensing User Directions ................................................... 23
3.3. Non-Functional Requirements ..................................................... 24
  3.3.1. Performance Requirements ............................................... 24
  3.3.2. Design Constraints ............................................................ 25
    3.3.2.1. Hardware Constraints ................................................. 25
    3.3.2.2. Software Constraints ................................................ 25
    3.3.2.3. Software System Attributes ........................................ 25
4. Data Model and Description ............................................................ 26
  4.1. Data Description ...................................................................... 27
    4.1.1. Data Objects ................................................................... 27
    4.1.2. Relationships .................................................................. 30
    4.1.3. Complete Data Model .................................................... 31
    4.1.4. Data Dictionary ................................................................. 31
5. Behaviour Model and Description .................................................... 32
  5.1. Description for Software Behaviour ........................................ 32
  5.2. State Transition Diagrams ....................................................... 35
6. Planning ......................................................................................... 36
  6.1. Team Structure ....................................................................... 36
  6.2. Estimation(Basic Schedule) ..................................................... 36
7. Conclusion ....................................................................................... 39
1. Introduction

This document is the software requirements specification of Agent Based Virtual & Constructive Simulation Framework (ABVCSF). We will talk about the main components of the software and give brief information about these components. These components are agent based simulation toolkit namely The Recursive Porous Agent Simulation Toolkit (REPAST), toolkit for geographic information namely OpenMap and ship simulation programme namely Bridge Command. Furthermore, we will explain the functionality of each component in the project and the data flow between them. In addition, use case diagrams will be introduced in order to visualize the concepts.

1.1. Problem Definition

In this document, we will design an agent based live, virtual and constructive (LVC) simulation framework by integrating two open source toolkits namely REPAST and OpenMap. After producing this framework, we will combine it with a sample game called Bridge Command. In this stage we will bind the virtual components in the game with our simulation platform. Although, in proposal, we introduced that we will combine our framework with a basketball game after meeting with our project manager, we decided to change the scenario into a military ship simulation.

1.2. Purpose

This document includes software requirements for Agent Based Virtual & Constructive Simulation Framework (ABVCSF). ABVCSF aims to integrate REPAST and OpenMap and create a framework in order to make a simulation. By this way, this framework will gain different functionality. This feature is not available in the other products in market. Therefore, our project will be different from them in that way.
1.3. **Scope**

The project that we will work on is Agent Based Virtual & Constructive Simulation Framework. The general usage of this project will be in military services. It aims to train marines in naval attacks. Users of this software product consists of related staff from navy forces.

There will be unmanned surface vehicles (USV) in the simulation which protect harbor. They are armed with radars and guns. Player reaches to this harbor by avoiding other USVs. Then, USVs will detect them according to its AI functions and alerts when they see player in their radar. Detection will happen after player come to USVs’ range.

The user will gain experience by using this simulation before he actually gets into this kind of operation. Map utility namely OpenMap, will provide us to make simulations that are very close to reality. By this way, we can locate our USVs in real harbors and real open sea locations. Therefore, it will be very beneficial for navies to learn how to make an operation.

1.4. **User and Literature Survey**

Potential users of this product:

- Since simulation games such as Microsoft Flight Simulator are played by many players, this product can be used by those players too.
- Since product's aim is to train navy forces, armies of countries will use this.
- In order to be informed about such technologies and create such kind of products, especially military simulation based corporations can use this product.

There are several existing products made by various companies:

- Jane’s Company[^1] made Submarine And Surface Ship Asw Combat Information Systems
- CAE’s[^2] Simfinit 2D Trainers. They provide such simulations as a part of training specific departments.
• Danish company IFAD TS A/S\textsuperscript{[10]} made The RDN Tactical Trainer which provides for tactical team training.

• Maritime and Offshore - Ships' Bridge Simululator. The Bridge Simulation Suite consists of a realistic mock-up of a ship's bridge with all conventional controls and instruments.

1.5. Definitions and Abbreviations

\textbf{ABVCSF:} Agent Based Virtual & Constructive Simulation Framework

\textbf{AI:} Artificial Intelligence

\textbf{GIS:} Geographic Information System

\textbf{JVM:} Java Virtual Machine

\textbf{LVC:} Live-Virtual-Constructive

\textbf{OS:} Operating System

\textbf{REPAST:} The Recursive Porous Agent Simulation Toolkit

\textbf{SAGE:} Savunma Sanayii Araştırma ve Geliştirme Enstitüsü

\textbf{SRS:} Software Requirements Specification

\textbf{TÜBİTAK:} Türkçe Bilimsel ve Teknolojik Araştırma Kurumu

\textbf{UAV:} Unmanned Aerial Vehicle

\textbf{USV:} Unmanned Surface Vehicles
1.6. References

I. Web Sources:


http://www.openmap.org/

http://www.bridgecommand.co.uk/download.php

http://repast.sourceforge.net/

[7] Official webpage of the programming language Blitz3D.
http://www.blitzbasic.com/Products/blitz3d.php

http://www.lowestoft.ac.uk/marbridge.asp

https://creately.com/
II. Book Sources:

• Fundamentals of Multiagent Systems with NetLogo Examples
  - Jose M Vidal

• MULTIAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations
  – Yoav Shoham and Kevin Leyton-Brown

1.7. Overview

The rest of the document is divided into chapters for better understanding.

• In chapter 2 an overall description of “Agent Based LVC Simulation Framework” project is provided. First product perspective is presented with product features and main functions. Then follow user classes and characteristics, operating environments that “Agent Based LVC Simulation Framework” supports as well as design and implementation constraints.

• In chapter 3, major software requirements are detailed.

• In chapter 4, most important features are presented with detailed description, use cases and requirements.

• In chapter 5, we will briefly introduce the behavioral model of our project and illustrate it with specific diagrams.

• In chapter 6, we will introduce our team and planning of our team.
2. Overall Description

We will use mainly three toolkits while creating the project:

i. **REPAST**[^6]:

   The Recursive Porous Agent Simulation Toolkit (REPAST) is one of several agent modeling toolkits that are available. REPAST borrows many concepts from the Swarm agent-based modeling toolkit. REPAST is differentiated from Swarm since REPAST has multiple pure implementations in several languages and built-in adaptive features such as genetic algorithms and regression.

   REPAST includes various numbers of different functionalities:
   - REPAST includes a variety of agent templates and examples. However, the toolkit gives users complete flexibility as to how they specify the properties and behaviors of agents.
   - REPAST is fully object-oriented.
   - REPAST includes a fully concurrent discrete event scheduler. This scheduler supports both sequential and parallel discrete event operations.
   - REPAST offers built-in simulation results logging and graphing tools.
   - REPAST provides a range of two-dimensional agent environments and visualizations.
   - REPAST allows users to dynamically access and modify agent properties, agent behavioral equations, and model properties at run time.
   - REPAST includes libraries for genetic algorithms, neural networks, random number generation, and specialized mathematics.
   - REPAST includes built-in systems dynamics modeling.
   - REPAST has social network modeling support tools.
   - REPAST has integrated geographical information systems (GIS) support.
   - REPAST is fully implemented in a variety of languages including Java and C#.
   - REPAST models can be developed in many languages including Java, C#, Managed C++, Visual Basic.Net, Managed Lisp, Managed Prolog, and Python scripting.
• REPAST is available on virtually all modern computing platforms including Windows, Mac OS, and Linux. The platform support includes both personal computers and large-scale scientific computing clusters.

In our software product, it will be used for creating special agents and also it will provide the framework which will run our simulation. It will run simultaneously with OpenMap in our framework.

REPAST interface can be seen from Fig 2.1.

![Fig 2.1: REPAST Interface](image)

ii. OpenMap:

OpenMap is a Java Beans based toolkit for building applications and applets needing geographic information. Using OpenMap components, you can access data from legacy applications, in-place, in a distributed setting. At its core, OpenMap is a
set of Swing components that understand geographic coordinates. These components help you show map data, and help you handle user input events to manipulate that data.

OpenMap will be used as a GIS provider in our project. It will generate GIS for our agents which will provide geographical computational utility to our simulation. Our agents will also use this generated GIS to get the information about their positions in the world. Furthermore, in order to give the needed realistic environment for our simulation, OpenMap will give us real coordinates of real harbors and etc.

OpenMap interface can be seen from Fig 2.2.
iii. Bridge Command\textsuperscript{[5]}:

Bridge Command is a free interactive ship simulator programme. Its aim is to be a training tool for navigation, ship handling, and other seamanship skills. Bridge Command is very highly flexible, with the environmental model, ships, buoys, individual scenarios and almost all aspects of the simulation being configurable to your own needs. Bridge Command can be run on a single computer, with no additional devices needed. It can also be run with physical controls, and has an interface where a trainer can control scenarios in order to carry out training exercises, from a second computer. It can display on a single screen, or can display a
full panoramic view across multiple screens using multiple computers. Bridge Command is written using the Blitz3D language.

In our project, we will use Bridge Command as a tool for providing the simulation environment of the virtual components.

Bridge Command interface can be seen from Fig 2.3.

![Bridge Command Interface](image)

**Fig 2.3: Bridge Command Interface**

### 2.1. Product Perspective

In our project, “Agent Based LVC Simulation Framework”, the purpose is to provide a simulated warfare environment in the sea. There are already available simulations for the same purpose. However, in our design, it will be composed of two opensource toolkits which are REPAST and OpenMap as integrated. It is one of the main difficulties of this project to integrate them. As an interface feature, we will use bridge command simulation to let the user interact with our simulated USV. Block diagram can be seen from Fig 2.4.
2.1.1. System Interface

The system interfaces in this project consists of USVs (our agents) and a GIS which they will use for receiving the location information. The decision making mechanism of USVs’ is that each agent will act according to rules described in their knowledge base. This will be achieved by turning simple agents into rule-based agents.

2.1.2. User Interface

Our user interface will be integrated into Bridge Command simulation. The users of simulation will be able to do the followings:

- Accelerate and decelerate the USV by clicking left mouse button.
- Get the latitude and longitude of their location.
• Get the radar information in order to see other agents around.
• Get the weather forecast and the vision range.
• Get the speed and depth under keel of USV.

2.1.3. Hardware Interfaces

<table>
<thead>
<tr>
<th>Processor</th>
<th>RAM</th>
<th>HDD</th>
<th>Graphic Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Core</td>
<td>2 GB</td>
<td>40 GB</td>
<td>256 MB</td>
</tr>
</tbody>
</table>

• The system should have minimum 1.6 GHz dual core Intel or AMD Processor.
• The system should have a main board with support of VGA LAN and sound.
• Minimum of 256 MB Graphic Card should be available on the system.
• The system should have a DDR2 Memory with at least 2 GB capacity.
• An external storage of minimum 40 GB should be available on the system.
• Minimum 15” LCD TFT Touch Screen should be connected to the system.
• The system should be connected to a power supply that provides minimum 250 Watt power.
• A cooler bigger than 10x10 cm should be connected to the system.

2.1.4. Software Interface

• Agent Based Simulation Toolkit
  Name of the Software: REPAST Symphony
  Version Number: 1.2.0
  Mnemonic: 8 KB
  Source: http://repast.sourceforge.net/download.html

• Toolkit for Geographic Information
  Name of the Software: OpenMap
i. REPAST-OpenMap Interface:

REPAST interface will be used for creating special agents and also it will provide the framework which will run our simulation. Furthermore, it will have an interface with OpenMap in order to maintain a dataflow of GIS information for the agents. It will use that dataflow to determine the locations of USVs. The frequency of information coming from OpenMap will be determined by a special function in order to maintain real-time interaction.

Dataflow diagram can be seen from Fig 2.5.

![Dataflow Diagram of REPAST-OpenMap Interface](image)

ii. REPAST-Bridge Command Interface:

Bridge Command will be used for visualize the environment which includes USVs and other geographical scenery. Furthermore, we will introduce REPAST to Bridge Command so that USV models will act according to created agent classes in REPAST.

REPAST will take user commands from Bridge Command and keeps related coordinates in the framework.

Dataflow diagram can be seen from Fig 2.6.
iii. Bridge Command-OpenMap Interface:

The GIS information which is generated by OpenMap will be brought to Bridge Command. This information will be used on radar screen so that user can avoid from AI. Moreover, there will be latitude and longitude information of our USV which is also taken from OpenMap.

Dataflow diagram can be seen from Fig 2.7.
Overall diagram can be seen as follows:

![Overall Dataflow Diagram](image)

*Fig 2.8: Overall Dataflow Diagram*
2.1.5. Memory Constraints

The simulation will need a memory of 2 GB and in case of saving the simulation, our system should provide 40 GB hard disk drive.

2.2. Product Functions

User functions:
- special_keys(): This function allows user to change certain factors of simulation environment. It is explained in the table 2.2.a.
- adjust_speed(): This function allows user to adjust the speed of USV.
- adjust_direction(): This function allows user to adjust the direction of USV.

<table>
<thead>
<tr>
<th>Key</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrows</td>
<td>Look around</td>
</tr>
<tr>
<td>Space</td>
<td>Change viewpoint on vessel</td>
</tr>
<tr>
<td>F1</td>
<td>Help</td>
</tr>
<tr>
<td>F4</td>
<td>Magnify view</td>
</tr>
<tr>
<td>F5</td>
<td>Normal view</td>
</tr>
<tr>
<td>ESC</td>
<td>Exit Bridge Command</td>
</tr>
<tr>
<td>F6</td>
<td>Make wind &amp; weather worse</td>
</tr>
<tr>
<td>F7</td>
<td>Make wind &amp; weather better</td>
</tr>
<tr>
<td>F8</td>
<td>Make visibility worse</td>
</tr>
<tr>
<td>F9</td>
<td>Make visibility better</td>
</tr>
<tr>
<td>Home</td>
<td>Decrease wind bearing</td>
</tr>
<tr>
<td>End</td>
<td>Increase wind bearing</td>
</tr>
<tr>
<td>0</td>
<td>Pause simulation</td>
</tr>
</tbody>
</table>

Fig 2.9: Table of user functions.
2.3. Constraints, Assumptions and Dependencies

i. Constraints:

- In order to bind the virtual components to our simulation framework, we have to design a method and make the simulation framework suitable for that.
- Connection ports of virtual simulation will be made.
- Real time working problem will be solved.
- Updates for connecting framework which is developed for opensource virtual simulations will be made.
- If the agents that are created does not work properly, users can not get the benefit of the simulation that is aimed for. Therefore, our system have to be aware of such kind of situations.

ii. Assumptions & Dependencies:

This project will be produced in Windows OS. Therefore, it may cause some problems if we try it on Linux OS. However, Java applications are typically compiled to bytecode (class file) that can run on any JVM regardless of computer architecture.

3. Specific Requirements

3.1. Interface Requirements

Player functions are:

a) Looking around from board of USV

By using arrow keys; UP, DOWN, LEFT, RIGHT user can see the environment from different perspectives.

b) Changing viewpoint on vessel
By using SPACE key, user can change the place of viewpoint to 3 different spots.

c) Help

By using F1 key, user can see user manual of the simulation.

d) Changing view mode

By using F4 key, user can change to magnified view and using F5, user can change to normal view.

e) Quit

By using ESC key, user can exit the simulation.

f) Changing weather conditions

By using F6 and F7 keys, user can make the weather worse or better.

g) Fog function

By using F8 and F9 keys, user can determine the level of fog.

h) Changing wind speed

By using HOME and END keys, user can change the speed of wind.

i) Pause

By using 0 key, user can pause and then continue the simulation.

Player Use Case:

Fig 3.1: Player Use Case 1.
Administrator functions are:

a) \textit{Creating a new scenario}

\textit{By changing the data in framework, administrator can change the default scenario.}

b) \textit{Changing type of agents}

\textit{Administrator can change the type of the agent as he needed such as simple agent.}

c) \textit{Error handling}

\textit{Administrator maintains the usability of software by handling the errors that can happen through game time.}
3.2. Functional Requirements

3.2.1. Creating Harbor

When we first start the simulation, we will get location input, which we will use to create the harbor, via the OpenMap interface. After user select the harbor position, this information will be sent to REPAST so that agent creation function will be called.

3.2.2. Creating Agents

We will use REPAST in order to design our own special agents by using this method. We will use REPAST’s default agent classes and change them into rule-based agents for our purpose. The data flow for this function can be explained that after we create agents, OpenMap must be informed and we have to send the geographical location information to OpenMap.

3.2.3. Creating Player

We will create an instance of a USV class in REPAST in order to let the player control the specific vehicle. We will initialize its current position to a default distance from the harbor. After that this information will be sent to OpenMap.
3.2.4. Changing Location Information

In this function, we will get the current location information of the user from OpenMap. After that according to user direction input we will update its location and then give it back to OpenMap. We will do the same procedure for the agents.

3.2.5. Transferring Positions to Bridge Command

In this function, we will get the positions of the harbor and USVs from OpenMap and transfer it to Bridge Command input function as an argument. By this way, we will see simulation on the Bridge Command screen.

3.2.6. Sensing User Directions

In this function, we will move player according to commands that are given to Bridge Command. Then, it will trigger moves in our map and functions in our framework.

3.3. Non-functional Requirements

3.3.1. Performance Requirements

Performance is one of the key requirements for this project because this project works in a specific environment which requires real-time decisions and reactions. The performance requirements should be evaluated in two ways: decision-based performance and reaction-based performance. First, performance requirements, decision-based performance, must be accomplished in a plausible time so that reaction-based task works in accurate way. Decision-based task should not interfere in the interaction between agents which are USVs other than our USV in this project.

Secondly, reaction-based task should also be achieved in a plausible time to make simulation works correctly. Reaction-based task also requires interaction between agents. Moreover, simulation graphics should be updated in fewer than one second. Since there is one user in this project, we are not thinking about scalability and its requirements.
3.3.2. Design Constraints

1. Hardware Constraints

- The minimum requirements are 1.6 GHz dual core Intel or AMD Processor with 256 MB Graphic Card, DDR2 Memory with at least 2 GB capacity, VGA LAN and sound. The system should be connected to a power supply that provides minimum 250 Watt power. Moreover, an external storage of minimum 40 GB should be available on the system and minimum 15” LCD TFT Touch Screen should be connected to the system.

2. Software Constraints

Any Windows with Java Virtual Machine will be used. As a programming language Java is going to be used. As a standard Java Coding Standards to which Repast and OpenMap adapt are going to be used.

When we will bind our framework with Bridge Command, we are going to use Blitz3D programming language. Blitz3D provides a simple yet powerful environment for game creation - simple, because its based around the popular and easy to use BASIC programming language; and powerful, thanks to a highly optimized underlying 2D/3D engine. Blitz3D includes many commands to help programmers out with game creation. Blitz3D’s command set has been carefully designed to provide maximum flexibility for minimum effort.

3. Software System Attributes

Usability: The system should have easy and minimum number of user interfaces. Any person knowing how to use computer should be able to use our system.

Documentation: The system should include a tutorial-like documentation which includes the informations about how to set up and run the simulation.

Availability: The system should be available to user any time he/she wants to access after installing the software.
Reliability: The system should run two simulations in case of one of them fails. Thus, in this case, the second simulation will be active and the system will continue working normally. This two simulation systems helps users to continue their experience with simulation environment.

Security: Since, for now, the system is not working on the web or web based systems, we are not thinking about the security of the system.

Portability: Since we are using Javadoc, our system can run on any system which provides JVM.

4. Data Model and Description

In this section, data models and data object types are described. Then attributes and methods of object types are discussed. After that, relationship between them are given with diagram representation. Finally, overall entity-relationship model is given according to specifications given in previous sections.

When we explain data object types’ attributes and methods, we emphasize on main attributes and main methods they include. Extendibility feature is discussed generally, but no details about implementation is given.

In our software, we do not use any advanced database method. Alternately, we will use XML or text to keep our records. Then, we will use this records for some features like saving and replaying.

Saving:

For this operation, we keep current positions of agents, player and objects on map.

Replaying:

For this operation, we keep track of player’s moves, agent’s moves and distribution of objects in our map by observing them at some regular time intervals. Then, we will have a chance to gain some features like pausing, continuing from where we stop and restarting.
4.1. Data Description

There will be 5 main data object types. These are:

Agent:

Instances that are created from this data object type represents agents in our software product. It keeps various attributes of agents and has some functionality for moving agents in order to AI logic.

Map:

Instances that are created from this data object type represents map that we are dealing with. It contains some information about shapes in our map and some methods for manipulating our map.

Coordinate:

Instances that are created from this data object type represents coordinates of shapes, agents or other objects that require coordinate logic.

Shape:

Instances that are created from this data object type represents shapes in our map. They can be sea, land, gulf etc. It has some features for extendibility.

Player:

Instances that are created from this data object type represents our player. It would be very useful in replay manner. We can observe behaviour of player by keeping instances of Player class at different time intervals. Then, we can use this for replay feature.

4.1.1. Data objects

Agent:

a) Extendibility:

There is no need for making this class abstract.

b) Attributes:

State: It keeps user command information that is taken from Bridge Command. It behaves according to this state.
Coordinate: It will show coordinate of agent. Coordinate object type will be used.

c) Methods:

Move Left: It is for moving agent to left.
Move Right: It is for moving agent to right.
Move Up: It is for moving agent to up.
Move Down: It is for moving agent to down.

Map:

a) Extendibility:

There is no need for making this class abstract.

b) Attributes:

Shape List: It will keep list of shapes in map

c) Methods:

Add Shape: It is for adding shape to map.
Remove Shape: It is for removing shape from map.

Coordinate:

a) Extendibility:

There is no need for making this class abstract.

b) Attributes:

North: It will keep north coordinate.
South: It will keep south coordinate.
West: It will keep west coordinate.
East: It will keep east coordinate.

c) Methods:

Change Coordinate: It is for changing coordinate from existing to another.
Shape:

a) Extendibility:

This class will be abstract. It will be extendible. According to shapes characteristic attributes we will add some fields to this object type. For example, for the mountain we will add height field to shape class extend it as an Mountain class.

b) Attributes:

Coordinate: It will show coordinate of shape. Coordinate object type will be used.

c) Methods:

Change Coordinate: It is for changing coordinate of map from existing to another.

Player:

a) Extendibility:

We can extend agent class to form Player object type. It is implementation dependent.

b) Attributes:

Coordinate: It will show coordinate of player. Coordinate object type will be used.

c) Methods:

Execute Right Engine(Port): It gives power to right power engine according to arguments that it takes.

Execute Left Engine(Sted): It gives power to left power engine according to arguments that it takes.

Adjust Rudder: It will adjust rudder to indicate the direction where it moves.

Move Back(Astern): It will enables player for moving back. It is similar to a two state machine.
4.1.2. Relationships

Relationships among data objects are shown in Figure 4.1.

Fig 4.1: Relationship among data objects
4.1.3. Complete Data Model

Overall entity-relationship diagram is shown in Figure 4.2.

![Overall entity-relationship diagram](image)

Fig 4.2: Overall entity-relationship diagram

4.1.4. Data Dictionary

Since data will be kept in XML format, data dictionary is provided in simple format. Actually, all fields seem as string but inner types are provided below.

<table>
<thead>
<tr>
<th></th>
<th>DataTypes</th>
<th>DataEditable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AgentID</td>
<td>int</td>
<td>No</td>
</tr>
<tr>
<td>State</td>
<td>int</td>
<td>Yes</td>
</tr>
<tr>
<td>CoordinateID</td>
<td>int</td>
<td>Yes</td>
</tr>
<tr>
<td>Shapes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ShapeID</td>
<td>int</td>
<td>No</td>
</tr>
<tr>
<td>CoordinateID</td>
<td>int</td>
<td>Yes</td>
</tr>
<tr>
<td>Maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapID</td>
<td>int</td>
<td>No</td>
</tr>
<tr>
<td>Shapeld</td>
<td>int</td>
<td>Yes</td>
</tr>
<tr>
<td>Cooridnates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CoordinateID</td>
<td>int</td>
<td>No</td>
</tr>
<tr>
<td>North</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>South</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>West</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>East</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>Players</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PlayerID</td>
<td>int</td>
<td>No</td>
</tr>
<tr>
<td>CoordinateID</td>
<td>int</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Fig 4.3: Data Dictionary
5. Behavioral Model and Description

Firstly, the player starts the simulation game. Then the simulation game provides a user interface for the player. Player can select some of the specific features of the software which will be about choosing the specific harbor location by the help of OpenMap, difficulty level of game and number of USVs that will protect the harbor.

After taking the input from player, our simulation will do the followings:

- Create harbor on a specific location.
- Create specific number of USVs.
- Grant these USVs needed level of AI.

When the required environment is generated, simulation will start and perceive the actions of player with a real time manner. Then it responds according to them.

The simulation ends when USVs detect the location of the player or player reaches to the harbor without getting captured.

5.1. Description for software behavior

State 1: Start State

User interface of ABVCSF provides the functionalities for players which are mentioned in the section 5.
State 2: Input State

Player selects the options which he/she wants the simulation to include. These options can be seen in figure 5.1. They are namely “Create Agent, Locate Harbor and Change Location”.

State 3: Generation State

After getting information from player, in this state, ABVCSF creates the simulation environment. Simulation environment consists of geographical data and also the agents that are defending the harbor. The generated scene will be modified version of the below:
State 4: Game State

This state consists of basic gaming interactions between player and simulation framework. It continues until simulation comes to the end.
State 5: End State:

This state indicates that simulation ends in certain conditions as we mentioned before.

5.2. State Transition Diagrams

Fig 5.4: Overall Transition Diagram
6. Planning

6.1. Team Structure

- In our team, we do not have a team leader.
- Every decision is made by team members.
- Every Friday a meeting is held and every one should be on time.
- If somebody can not come to the meeting, he should inform other members as soon as possible.
- The given responsibilities in previous meeting should be completed until the next meeting.
- Every team member should check his mail everyday.

Roles of team members:
Deniz Olgun: Optimist & Recorder
Deniz Tuna: Devil’s Advocate & Summarizer
Osman Tuncer Kaplankıran: Timekeeper & Recorder
Mehmet Safa Ertekin: Initiator & Gate Keeper

6.2. Estimation (Basic Schedule)

Week 1-4: Introduction to Project
- Grouping
- Selection of Project Topic
- Submission of Proposal

Week 5-9: Analysis
- Market Research
- Requirement Analysis
- Submission of Software Requirement Specifications

Week 10-12: Design
- Framework Design

Week 13-16: Implementation and Demo
- Team Presentation
7. Conclusion

In this document, we explained the main components of our project, ABVCSF.

- In chapter 1, we gave a brief introduction.
- In chapter 2, we stated a general description.
- In chapter 3, major software requirements are detailed.
- In chapter 4, most important features are presented with detailed description, use cases and requirements.
- In chapter 5, we will briefly introduce the behavioral model of our project and illustrate it with specific diagrams.
- In chapter 6, we will introduce our team and planning of our team.