CENG 491
Computer Engineering Design

DETAILED DESIGN REPORT

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. Overview.................................................................................................... 6  
   1.5. Definitions, Acronyms and Abbreviations ............................................. 6  
   1.6. References.................................................................................................. 7  
2. System Overview................................................................................................ 8  
   2.1. Repast......................................................................................................... 8  
   2.2. OpenMap.................................................................................................... 9  
   2.3. Bridge Command...................................................................................... 10  
3. Design Considerations....................................................................................... 18  
   3.1. Design Assumptions, Dependencies and Constraints............................... 19  
      3.1.1. Design Assumptions............................................................................. 19  
      3.1.2. Design Dependencies......................................................................... 19  
      3.1.3. Design Constraints........................................................................... 19  
   3.2. Design Goals and Guidelines..................................................................... 20  
4. Data Design......................................................................................................... 22  
   4.1. Data Description........................................................................................ 22  
   4.2. Data Dictionary.......................................................................................... 24  
5. System Architecture............................................................................................ 26  
   5.1. Architectural Design................................................................................... 28  
      5.1.1. Repast.................................................................................................. 28  
      5.1.2. OpenMap............................................................................................ 29  
      5.1.3. Bridge Command................................................................................ 31  
   5.2. Description of Components........................................................................ 32  
      5.2.1. Information Gathering.......................................................................... 32  
      5.2.2. Event Scheduler.................................................................................. 32  
      5.2.3. Agent Creation and Modeling............................................................. 32  
      5.2.4. Framework Provider........................................................................... 33  
      5.2.5. Error Management............................................................................. 33  
      5.2.6. Database Updater............................................................................... 33  
      5.2.7. Decision Management........................................................................ 33
5.2.8. Data Manipulation ................................................................. 34
5.2.9. GIS Provider ................................................................. 34
5.2.10. Scenario Gathering ............................................................. 34
5.2.11. Data Collector (State Return) .............................................. 34
5.2.12. Navigator ................................................................. 35
5.2.13. Condition Management ..................................................... 35
5.2.14. Graphical Management .................................................... 35
5.3. Design Rationale .................................................................. 36
6. User Interface Design .............................................................. 38
   6.1. Overview of User Interface .................................................... 38
   6.2. Screen Images ..................................................................... 38
   6.3. Screen Objects and Actions .................................................. 41
7. Detailed Design ........................................................................ 42
   7.1. Detailed Component Design .................................................. 42
      7.1.1. Component Diagram ..................................................... 42
      7.1.2. Decision Management Component .................................. 44
      7.1.3. Data Manipulation Component ......................................... 45
      7.1.4. GIS Provider Component ............................................... 46
      7.1.5. Database Updater Component ......................................... 46
      7.1.6. Scenario Gathering Component........................................ 47
      7.1.7. Data Collector (State Return) Component ......................... 48
      7.1.8. Condition Management Component .................................. 49
      7.1.9. Graphical Management Component ................................... 50
      7.1.10. Navigator Component .................................................. 50
      7.1.11. Information Gathering Component ................................... 51
      7.1.12. Event Schedular Component .......................................... 52
      7.1.13. Agent Creation and Modeling Component ....................... 53
      7.1.14. Framework Provider Component ..................................... 54
      7.1.15. Error Management Component ....................................... 54
   7.2. Class Diagram ..................................................................... 56
8. Libraries and Tools .................................................................. 57
   8.1. Agent Based Simulation Toolkit ............................................. 57
   8.2. Toolkit for Geographic Information ........................................ 57
   8.3. Ship Simulation Programme .................................................. 57
9. Time Planning (Gannt Chart) ..................................................... 58
   9.1. Term 1 Gannt Chart ............................................................ 58
   9.2. Term 2 Gannt Chart ............................................................. 60
10. Conclusion ................................................................................. 61
Introduction

This document is the detailed design report document of the Agent Based Virtual & Constructive Simulation Framework. 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, geographic information toolkit 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. Moreover, we will introduce system architecture of our project in a brief manner. Then we will give some screenshots from our newly designed user interface from a point of view of user. Furthermore, we will briefly mention details about our design. We will briefly explain the parts of System Architecture part by using modules, class diagrams and etc.

1.1. Problem Definition

We will design an Agent Based Live, Virtual and Constructive simulation framework in order to provide a military simulation game.

We will make this simulation 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. While doing this stage in our project, we will bind the virtual components of the game with our framework. Moreover, we will provide an user interface which will override the user interface of Bridge Command in order provide our users much more beneficial interface regardless of different functionalities that Bridge Command has.

1.2. Purpose

This document includes detailed design report of our project named as Agent Based Virtual & Constructive Simulation Framework. Main purpose of this project is gathering two open source toolkits namely The Recursive Porous Agent Simulation Toolkit and OpenMap in a manner to create a simulation framework. Since there are lots of simulation frameworks that
have been already created by using other softwares or by using just one of our toolkits easily; our task is to do this job with the opensource toolkits REPAST and OpenMap in a different manner. We are supposed to create a simulation framework with these two toolkits so we have to integrate them. There are not any other simulation products which uses the framework as ours since they are not using our sources by integrating. It will be a hard task to make such a project. We will encounter with so many problems but we will learn how to evade from such problems with ease.

Furthermore this simulation will provide a realistic environment for soldiers so military will not waste money on items that are being used in such kind of operations. By using our simulation game, we will help military to save lots of money in reality.

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 and ambush operations. Users of this software product consists of related staff from navy forces.

There will be unmanned surface vehicles in the simulation which protect harbor. They are armed with radars and guns. Player reaches to this harbor by avoiding other unmanned surface vehicles.

In our design unmanned surface vehicles are equipped with radars. By using those radars, they will detect players according to their artificial intelligence functions and will generate alerts when they see a player in their radar. Detection will happen after player come to any of the unmanned surface vehicles’ range.

Map utility namely OpenMap, will provide us to make simulations that are very close to reality. By this way, we can locate our unmanned surface vehicle in real harbers and real open sea locations. Therefore, it will be very beneficial for military troops to learn how to make an operation without getting into a real war. The user will gain experience by using this simulation before he actually gets into this kind of scenario in real life. Soldiers will be much more experienced after using this simulation game in using radars to avoid from enemies in such operations.
1.4. Overview

The rest of the document is divided into chapters in order to provide a better understanding of our product.

In the next chapter, namely second chapter of this document, we will provide a general description of the software system including its functionality and matters related to overal system and design.

Third chapter of this document will be about design considerations. Moreover we will talk about design assumptions, dependencies and constraints and also our design goal.

Fourth chapter of this document will give brief information about most important features of data structures of our project with detailed description. We will present use cases and requirements about our data structures in this chapter.

Fifth chapter will be about system architecture of our project. We will show how our system works by using needed UML diagrams and etc. Furthermore in this chapter we will talk about our project’s main components in a detailed manner and also we will talk about the algorithms and solving methods for our system.

In the sixth chapter of our this document , we will provide the functionalities that user can reach in our system. Moreover we will grant some screenshots of our user interface for more understanding.

In the seventh chapter, we will provide a brief explanations for our System Architecture by using models, classes, packages and etc.

In the eighth chapter, libraries and tools which are used will be discussed.

In the ninth chapter, we will give the gaant chart of our process in a detailed manner both for this semester and also next semester.

Last chapter namely the tenth chapter of this document will be about the conclusion part.

1.5. Definitions, Acronyms and Abbreviations

- ABVCSF: Agent Based Virtual & Constructive Simulation Framework
• AI: Artificial Intelligence
• GIS: Geographic Information System
• JVM: Java Virtual Machine
• LVC: Live-Virtual-Constructive
• MAS: Multi Agent Systems
• OS: Operating System
• REPASt: The Recursive Porous Agent Simulation Toolkit
• SAGE: Savunma Sanayii Araştırma ve Geliştirme Enstitüsü
• SRS: Software Requirements Specification
• TÜBİTAK: Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
• UAV: Unmanned Aerial Vehicle
• USV: Unmanned Surface Vehicles

1.6. References

• http://www.cae.com/en/military/simfinity.2d.asp
• http://www.sage.tubitak.gov.tr/index.asp
• http://www.openmap.org/
• http://www.bridgecommand.co.uk/download.php
• http://repast.sourceforge.net/
• http://www.blitzbasic.com/Products/blitz3d.php
• http://www.lowestoft.ac.uk/marbridge.asp
• https://creately.com/
• http://openmap.bbn.com/doc/openmap-arch.html
2. System Overview

We will use mainly three toolkits while creating the project:

2.1. REPAST:

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.

In our software product, REPAST 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 will be integrated with OpenMap in our design so it is the most important tool of our project.

In our project we will create ruled-based agents which are modified types of simple agents. In order to create such agents we will import the needed files and classes from REPAST and use those in our implications. Different from simple agents, rule-based agents will not do the same patterns and will not do same things in simulation. Thanks to this upgrade, our simulation can be playable on different difficulties and also our simulation will provide a much more realistic feeling to its users. Also, improvements on agents’ behavior will provide us to generate different difficulty levels.
2.2. 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. This will be done through classes of REPAST which will be imported into our design. Furthermore, in order to give the needed realistic environment for our simulation, OpenMap will give us real coordinates of real harbors and etc. Moreover, we will use OpenMap in our project to choose real harbors from world. This will be done by using an interface in user interface menu.

![OpenMap Interface](image)

Figure 2.2: Sample of OpenMap Interface.

2.3. **Bridge Command**

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. Moreover we will change the original interface of Bridge Command in order to provide our users much more clean interface since Bridge Command provides lots of functions to its users which are not needed in our purpose. Though, we will use ships and other material of Bridge Command.

Figure 2.3: A sample of Bridge Command User Interface.
By putting this project into practice, we will grant different benefits for the users of it. There will be lots of benefits from this project for users which will be discussed briefly in the next paragraphs. Before starting stating those benefits, let us mention about the goals and objectives of our project.

There is a variety of goals about making such a project. Main goal of this project is, as it is mentioned in the introduction chapter, granting a realistic atmosphere for doing naval attacks and ambushes into harbors. There will be other usage areas of our framework for sure, but we will consider only in the scenario of breaking into a harbor with some defenses in there. By granting such a realistic simulated environment, project will provide its users, generally military people, a great chance to practice their skills in some certain areas such as using radar knowledge in order to avoid from defenses in a harbor, generating different strategies in different environments and in different weather conditions while making operation into a harbor and etc. Furthermore, by using this simulation, military will get rid of huge money waste in order to provide guns, fuel, vehicles and etc while making such kind of operations. There will be a huge saving because of this reason and that will be a good point in our project. Also environmental issues will be decreased since there will be no real usage of guns, vehicles and etc and that will be a nice point too.

There are lots of benefits for users of this product.

- Since its target market is generally military, this product will provide a great chance to increase their knowledge in making such kind of operations.
- Users’ experience in using radar of their vehicle will greatly increase since users have to check radar in order to avoid from AI USV’s while they are sneaking into harbor.
- Military troops that are using this product will face a realistic environment in their operations before they will get into such kind of an operation.
- Tactic knowledge of the users will greatly increase.
- Since there will be different difficulties of this simulation game, learning and gaining experience from this game will be very wide. Users just will not stay in a difficulty and when they will feel themselves ready for upper difficulties they will face with a harder AI.
Functionality of this project will be consists of lots of things but mainly they will enable players to interact with the system.

There will be lots of functionalities of this project but most important ones:

- Users can change the current harbor that they will try to make an operation by a system friendly user interface, which will be used by our GIS creator toolkit OpenMap.
- Users can change the difficulty of this project by deciding on the levels. There will be levels of this game namely easy, medium, hard etc. Changes in that manner will effect the number of enemies in the harbor and also it will change the behavior of those enemies. These enemies will be smarter in each level in order to make the game much more beneficial for the users.
- There will be lots of in game functionalities for users which will be supported by mainly Bridge Command.
- In emergency cases such as shutting down of the game, we will run another window simultaneously in order to prevent from such cases. This window will run if our program will collapse. However, this will be optional though.
- There will be options to pause the game and then replay the game. Moreover we will implement the functionality to save the game whenever users want to and whenever they want to load the game, they can do. This functionality will give users a chance to give some break in simulations in order to discuss the tactics and etc.

Users will generally reach those options from user interface of Bridge Command. Some of the functions that they can reach and user are shown in the table below.
<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>

**Table 2.1: Functionality table.**

Those functions are just some of the users' functions.

There are some use cases that people involved in the product. They are namely player use cases and administrator use case.

**Player Use Case:**

```
Player <look_around()> change_viewpoint() help() Bridge Command
```

*Fig 3.1: Player Use Case 1.*
Fig 3.2: Player Use Case 2.

Administrator functions are:

a) Creating a new scenario

By changing the data in framework, administrator can change the default scenario.

b) Changing type of agents

Administrator can change the type of the agent as he needed such as simple agent.

c) Error handling

Administrator maintains the usability of software by handling the errors that can happen through game time.

**Administrator Use Case:**
Since we are integrating REPAST and OpenMap open source toolkits, their relationship will be the most important part in that project. If we cannot accomplish that relationship in a proper way, our framework cannot carry out the needed options.

There will be relationships in the tools that we will use. First relationship will be among REPAST and OpenMap.

**REPAST – OpenMap Relationship:**

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.

Data-flow diagram can be seen in next picture.

---

**Figure 2.4: REPAST – OpenMap dataflow diagram.**
REPAST – Bridge Command Relationship:

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 2.5.

Figure 2.5: REPAST – Bridge Command dataflow diagram.

Bridge Command – OpenMap Relationship:

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 2.6.

Figure 2.6: OpenMap – Bridge Command dataflow diagram.
3. Design Considerations

Multi Agent Systems refer to a computer research domain that addresses systems which are composed of micro level entities -agents-, which have an autonomous and proactive behavior and interact through an environment, thus producing the overall system behavior which is observed at the macro level.

In our consideration, MAS is used as a programming paradigm to develop operational software. Since MAS is particularly suited to deploy distributed software systems that run in computational contexts in which a global control is hard or not possible to achieve, it is directly relational to simulational workspaces. MAS also represent modeling alternative for representing and simulating real-world or virtual systems which could be decomposed in interacting individuals.

At the same time, computer simulation is a unique way of designing, testing and studying both theories and real computer systems, for various purposes. As Shannon [Shannon, 1975] stated:

“The process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system and/or of evaluating various strategies (within the limits imposed by a criterion or a set of criteria) for the operation of the system.”

Since the opportunities of using both MAS and simulation are numerous, precisely because they can be applied and/or coupled in a wide range of application domains, our project integrates both of them in order to make a simulation that runs on the specific framework that includes REPAST and OpenMap.
3.1. Design Assumptions, Dependencies and Constraints

3.1.1. Design Assumptions

The software is assumed to be used generally for military services, namely navy forces. In order to match the requirements of navy forces, there will be a simulation which is inspired from Bridge Command. It will be used for its graphical environment. We will put our agents into that environment. Moreover, there will not be multiplayer option, so there will be one user at each time using the simulation.

3.1.2. Design Dependencies

It is considered that the software will run on Windows machines. Therefore, it will be developed on Windows machines. However, it is also considered to be applied to Unix machines since we use Java Virtual Machine.

3.1.3. Design Constraints

-Hardware Constraints

Performance is a key topic for this project because it works with user interactively and it requires real time decisions and reactions. The performance constraints are considered in two ways, namely decision based performance and reaction based performance. Both of them should satisfy appropriate time requirement so that reaction based task works accurately with decision based task. Time is an important factor to consider, so the hardware of system should be able to respond its needs.

Since our project is a simulation software, hardware of computers should provide required infrastructure for visual representation of simulation. Moreover, since there will not be any database relation, PC should have its own adequate hard drive capacity in order to be able to save and replay a session with simulation.

Thinking of all these constraints, the system on which our simulation will work should have at least the following hardware requirements:

*CPU: 1.6 GHz dual core Intel or AMD Processor*
-Software Constraints

We will implement our designs in Java. Moreover, it may require some knowledge about Blitz3D when we prepare graphical environment and enhanced user interface since Bridge Command is in that language. Basicly, Java Virtual Machine will be used. Java Coding Standards will be applied to which Repast and OpenMap adapt are going to be used.

3.2. Design Goals and Guidelines

Although, our primary aim of the software is to integrate two discrete toolkits, REPAST and OpenMap, we will start with implementing our user friendly interface. This interface will be different from Bridge Command’s original user interface since it is not satisfy our needs. After finishing our first task, which will be an intial start for project, we will come to our most important goal of the project, which is to build the infrastructure of our framework. This will be the most challenging part of work since there is not any other projects which use same framework. Rest of work will be mostly designing agents. This part of our design will take some time since we will try to use GIS info from OpenMap with our agents and generating such kind of agents will take some time. After creating needed agents for our purpose and generating the sample environment, we will be implementing our scenario. At that stage, we will take advantage of object oriented programming paradigm of Java. Class definitions that are already available in REPAST will be used as an extension to create our simple agent class. At this point, our guideline will be object oriented paradigm of Java.

In general we have several design guidelines that we take account:
1. **Useful**: Result of the project will be useful to people in a specific area namely navy forces. Project has to be beneficial for those people.

2. **Fast**: Software product should run with no delay time. Every second is important in nowadays. Also it has to run in a real time manner since it is a simulation game.

3. **Simple**: It is important for people to keep the design and user interface simple.

4. **Engaging**: Once a person uses that product, he should want to try again, so it must be a challenging task to do.

5. **Innovative**: After implementing our scenario, it should be open for new ideas and together with new tools to implement creating innovative products.

6. **Universal**: Design should be something that covers principles which have standards around the world.

7. **Profitable**: Design should be for today's and tomorrow's business.

8. **Beautiful**: User interface or graphical components of design should be attractive to people, delight the eye without distracting the mind.

9. **Trustworthy**: Product should give assurance to people who use it. It is important to make users relieved.

---

**Software System Attributes**

**Usability**: The system should have easy and minimum number of user interfaces. Since our graphical data generator tool Bridge Command has an user interface it does not completely satisfy our needs has much more detail than our needs. So we will implement a new friendly interface in order to satisfy this need and improve our projects usability. 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 information about how to set up and run the simulation. This menu can be reachable from user interface by a button called “help” in our design.

**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.

Scalability: Since there is one user in this project, we are not thinking about scalability and its requirements.

4. Data Design

4.1. Data Description

There will be no database implementation in our software project. Instead, we will use some XML files to keep track of agent’s moves according to their coordinates. Data structure of the system is based on temporary XML files, basic temporary “.out” files and temporary class objects as well. Let’s explain this data structures in detail:

File Structure Of The System:

a) Agent.XML:

In this file, we will store current coordinate information and state information of agents. This file will be updated at regular time intervals and it will be used in the “Save” feature of the system. It will look like:

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AgentInformation simulationName="Boat Simulation">
<Agent name="Agent 1">
<State>Left</State>
<Coordinate X>22,56</Coordinate X>
<Coordinate Y>35,56</Coordinate Y>
</Agent>
</AgentInformation>
```
b) .OUT files:

These files will store the information of agents’ coordinates in line-by-line fashion. They do this by extracting .OUT files at some regular time intervals. Files will be named as 1.out, 2.out etc.

Data Structure Of The System:

Data structure of the system will be composed of some object instances that are created from different class types. These data will be kept in memory and will be used in transitions between Repast and OpenMap. These are:

a) 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.

b) 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.

c) Coordinate:

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

e) **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.

### 4.2. Data Dictionary

In this section, we will provide object-oriented description of the data objects. We will do this by giving detailed explanation of data objects and major data types in the data structure.

**Data Objects:**

Most important classes in our data structures are given below.

**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 (A stern):** It will enable player for moving back. It is similar to a two state machine.

5. **System Architecture**

There are three distinct modules which behave differently according to their purposes and responsibilities, namely REPAST, OpenMap, and Bridge Command. REPAST 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. 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. BRIDGE COMMAND will be used as a tool for providing the simulation environment of the virtual components.

![Figure 5.1: Block Diagram of System Architecture as a whole](image)

Each module at the system has special purposes. Some of these modules have same component, but some of them have unique components.
5.1. Architectural Design

5.1.1. REPAST

Mainly, REPAST will be used for creating-manipulating agents according to the information which are taken from OPENMAP and for providing framework which will run our simulation.

Figure 5.1.1. REPAST COMPONENT DIAGRAM
5.1.2. OPENMAP

Mainly, OPENMAP will be used for taking geographical information which are requested and providing these information to the REPAST module. While OPENMAP is taking geographical information, it will manipulate those information according to some decision conditions.

Figure 5.1.2.a: Data Management of OPENMAP
Figure 5.1.2.b: OPENMAP COMPONENT DIAGRAM
5.1.3. BRIDGE COMMAND

Mainly, BRIDGE COMMAND will be used for providing the simulation environment of the virtual components. While BRIDGE COMMAND is creating this simulation environment, it will collect data from other modules. After collecting data from other modules, it can present variety of simulation environments.

Figure 5.1.3: BRIDGE COMMAND COMPONENT DIAGRAM
5.2. Description of Components

5.2.1. Information Gathering

Information Gathering component aims to get GIS information, agents’ states and agents’ conditions. As an input, it takes the GIS information from OpenMap, agents’ states and the agents’ conditions from Bridge Command. This component uses these inputs to be agents’ data. The implementation of this component is achieved by agent classes. If any crash (es) occurs in the REPAST, this situation will be reported to Error Management component.

5.2.2. Event Scheduler

Event Scheduler component aims to organize concurrent movements of the agents while states or conditions of the agents are changing. This component takes input from Information Gathering component and BRIDGE COMMAND module. When the states or the conditions (intelligence level) of the agents change, new GIS information of the agents are taken from OPENMAP, after that new GIS information are placed in the agents’ data. Then to arrange the concurrent state change Event Scheduler component is used. When Event Scheduler component is active, GIS information already exist in the agents’ data. As a consequence; this component takes states and the conditions of the agents as the input and creates the behaviour of the agents. The implementation of these components is achieved by agent classes. If any crash (es) occurs in the REPAST, this situation will be reported to Error Management component.

5.2.3. Agent Creation and Modeling

Agent Creation and Modeling component aims to create agents or modeling the agents which are already exist. Both procedure takes input from Event Scheduler component. Since we are interested in behaviours and the GIS information of the agents while creating or modeling the agents, this component collaborates with both of Event Scheduler component and Information Gathering component. The implementation of these components is achieved by agent classes. If
any crash (es) occurs in the REPAST, this situation will be reported to Error Management component.

5.2.4. Framework Provider

Framework Provider component aims to provide the agents which are already created or modeled by Agent Creation and Modeling component to BRIDGE COMMAND. Thus, new GIS information and the behaviours of the agents can be shown. If any crash (es) occurs in the REPAST, this situation will be reported to Error Management component.

5.2.5. Error Management

Error Management component aims to detect any crash (es) occurs in the REPAST and tries to fix this crash by activating second REPAST. While second REPAST is active, Error Management component takes care of the first (problematic) REPAST, and whenever the first (problematic) REPAST is fixed, it is activated again. This component collaborates with the all components which are included by REPAST.

5.2.6. Database Updater

Database Updater component aims to hold and update the geographical database of OPENMAP in order to adjust the real environments. Thus the GIS information of the real environments can easily be taken. This component has satellite related interfaces, thus inputs are taken from satellites.

5.2.7. Decision Management

Decision Management component makes necessary GIS information to be taken from Database Updater. Decision Management component enables this by taking states and the conditions of the agents from BRIDGE COMMAND module as the input and deciding which GIS information are necessary according to these states and conditions. As a consequence; this
component interacts with BRIDGE COMMAND module. Agent classes are implemented in this component.

5.2.8. Data Manipulation

Data Manipulation component picks up the necessary GIS information from Database Updater according to the input taken from Decision Management component. In other words, this component collaborates with both of Database Updater component and Decision Management component.

5.2.9. GIS Provider

GIS Provider component provides the GIS information taken from Data Manipulation component to the agents as the agents’ data. Thus, agent classes are implemented in this component also.

5.2.10. Scenario Gathering

Since BRIDGE COMMAND is a simulation program, it has variety of scenarios. Scenario Gathering component is user related component. Thus, Scenario Gathering component is used for determining which scenario will be displayed on the screen according to the input which is taken from user.

5.2.11. Data Collector (State Return)

After Scenario Gathering component takes the input from user, Scenario Gathering component output is taken by Data Collector component as the input. Since, at the beginning, BRIDGE COMMAND module already has uninitialized agents, this component interacts with both of OPENMAP and REPAST modules to initialize the agents and the environment according to the selected scenario. After the initialization of the agents and the environment, this
component interacts with both of OPENMAP and REPAST modules whenever the states or the conditions of the agents change.

After taking the agents’ data, agents’ behaviours and the GIS information of the environment from OPENMAP and REPAST modules, these data, behaviours and GIS information will be given from this component to Graphical Management component as the input. This component collaborates with Scenario Gathering and Condition Management components and this component also interacts with Graphical Management component. Agent classes are implemented in this component.

5.2.12. Navigator

This component takes user (user of the simulation) commands for navigating the ship. Thus, this component has no AI (Artificial Intelligence) and agent classes are not implemented in this component. This component interacts with Graphical Management component.

5.2.13. Condition Management

This component can change the agents’ and the environment’s condition. If agents’ condition (Intelligence level of the agent) is changed, this component interacts with Data Collector component. Then Data Collector component interacts with OPENMAP and REPAST, and then behaviours and the states of the agents change. After that; information will be given to the Graphical Management component.

If environment’s condition (weather etc.) is changed, this component interacts with Graphical Management component.

5.2.14. Graphical Management

This component consistently renders the environment and agents according to the information taken from REPAST and OPENMAP. This component collaborates with Condition Management component and Navigator component. This component also interacts with the Data Collector component.
5.3. Design Rationale

We chose this particular decomposition of the system, because each major sub-module (REPASTM, OPENMAP, BRIDGE COMMAND) has a special responsibility and decomposition of the sub-modules like this best describes the responsibilities of each. Some duties are common to components, so instead of creating new components permanently it is logical to combine these components to reduce the execution time. Thus less costly and more efficient software is created with our decomposition. Moreover, we chose this decomposition because extendibility feature of the software of this decomposition type is higher. We did not choose other designs when we thought cost and efficiency problem of the software.
Example code for the implementation of Bridge Command is something like that:

```plaintext
FileName="boot.x"
ScaleFactor=1.3038
YCorrection=-17
AngleCorrection=0
Filename2="radar.x"
X2=0.427
Y2=99.126
Z2=-28.382
ScaleFactor2=1
weapon_x=0.00
weapon_y=43.713
weapon_z=105.0
NumberOfLights=5
LightX(1)=26.4
LightY(1)=59.2
LightZ(1)=105.4
LightRange(1)=6
LightRed(1)=255
LightGreen(1)=0
LightBlue(1)=0
LightStartAngle(1)=-1
LightEndAngle(1)=112.5
LightX(2)=-26.2
LightY(2)=53.2
LightZ(2)=105.4
LightRange(2)=6
LightRed(2)=255
LightGreen(2)=0
LightBlue(2)=0
LightStartAngle(2)=47.5
LightEndAngle(2)=361
```

Figure 5.3: Creating Ships in Bridge Command
6. User Interface Design

In this section, we will describe user interface of our software project.

6.1. Overview of User Interface

There will be 4 main menus associated with our software project in the user interface. These are:

a) File Menu:

This menu will provide some utilities for file saving, file opening, replaying and program exiting functionalities.

b) Agent Menu:

This menu will be used for manipulating agents. By using this menu, user can create agents, change location of them and delete them.

c) Map Menu:

This menu will be used for map manipulation. By using this menu, user can locate harbors, change location of them and pick objects in the map for adjusting its attributes.

d) Help Menu:

This menu will be used for helping program users. It will include “help index” and “about” section.

6.2. Screen Images

Figure 6.2.a: Main theme
Figure 6.2.d: Map Menu

Figure 6.2.e: Help Menu
6.3. Screen Objects and Actions

a) File Menu

By using this menu, user can do some file operations. These are:

Saving:

User will save the last positions of the agents by using this functionality.

Opening:

User will open previously saved simulation environment by using this functionality. Agents’ and objects’ locations will be received from file.

Replaying:

Replaying of current simulation could be seen by using this functionality.

Closing:

Closing of current simulation will be done by using this functionality.

b) Agent Menu

By using this menu, user can do some agent operations. These are:

Creating Agent:

Agents will be created by using this functionality.

Changing Location of Agent:

Agent location will be changed by using this functionality.

Deleting Agent:

Agent will be deleted by using this functionality.

c) Map Menu
By using this menu, user can do some map operations. These are:

**Locating Harbor:**

User will locate harbor by using this functionality.

**Changing Location of Harbor:**

Harbor location will be changed by using this functionality.

**Choosing objects For Manipulating:**

User will pick objects in the map and reach attributes of the object by using this functionality.

**d) Help Menu**

By using this menu, user can reach info about usage of software. Some fields into his menu are:

**Help Index:**

User will get detailed explanation of a software topic-by-topic by using this field of menu.

**About:**

This portion will give some information about authors, version etc

7. Detailed Design

7.1. Detailed Component Design

7.1.1. Component Diagram
Figure 7.1: Component Diagram

Our design is divided into components. There exist fourteen components namely:

- **Decision Management**
- **Data Manipulation**
• GIS Provider
• Database Updater
• Scenario Gathering
• Data Collector (State Return)
• Condition Management
• Graphical Management
• Navigator
• Information Gathering
• Event Scheduler
• Agent Creation and Modeling
• Framework Provider
• Error Management

7.1.2. Decision Management Component

Decision Management component makes necessary GIS information to be taken from Database Updater. Decision Management component enables this by taking states and the conditions of the agents from BRIDGE COMMAND module as the input and deciding which GIS information are necessary according to these states and conditions.

![Decision Management Table]

```
Decision Management

#get_GIS_info();
#get_agent_state();
#get_agent_condition();
#decide_GIS();
```

Figure 7.2: Decision Management Component

Functions of Decision Management component:
# get_GIS_info(): Decision Management component of framework takes necessary GIS information from Database Updater component by this function.

# get_agent_state(): Decision Management component of framework gets agents current state through Bridge Command by using that function.

# get_agent_condition(): Decision Management component of framework gets agents current condition on simulation environment with this function.

# decide_GIS(): Decision Management component of framework decides on necessary GIS information for agents after using the functions below by using decide_GIS() function.

### 7.1.3. Data Manipulation Component

Data Manipulation component picks up the necessary GIS information from Database Updater according to the input taken from Decision Management component.

![Data Manipulation](image)

**Figure 7.3: Data Manipulation Component**

Functions of Data Manipulation component:

# get_GIS_info_from_DatabaseUpdater(): This component collaborates data flow between Database Updater and Decision Management by using this function. It picks up the necessary GIS information from Database Updater.
7.1.4. GIS Provider Component

GIS Provider component provides the GIS information taken from Data Manipulation component to the agents as the agents’ data.

![GIS Provider](image)

Figure 7.4: GIS Provider Component

Functions of GIS Provider component:

- `# get_GIS_info_from_DataManipulation():` GIS Provider component of framework takes GIS info from Data Manipulation component by using this function.

- `# give_agent_data():` After taking needed GIS information, GIS Provider component provides these data to agents by using this function.

7.1.5. Database Updater Component

Database Updater component aims to hold and update the geographical database of OPENMAP in order to adjust the real environments.

![Database Updater](image)
Figure 7.5: Database Updater Component

Functions of Database Updater component:

# store_GIS_info(): Database Updater component of framework uses this function in order to store real world coordinates that are taken from OpenMap toolkit.

# update_GIS_info(): According to agents positions in simulation environment this component update their real world coordinates by using this function.

7.1.6. Scenario Gathering Component

Scenario Gathering component is used for determining which scenario will be displayed on the screen according to the input which is taken from user.

```
Scenario Gathering
# select_scenario(): return string
# generate_scenario(): return string
```

Figure 7.6: Scenario Gathering Component

Functions of Scenario Gathering component:

# select_scenario(): Scenario Gathering component of framework uses this function in order to get the input from users in order to decide on which scenario will be displayed.

# generate_scenario(): By using this function, Scenario Gathering component of framework generates selected scenario environment.
7.1.7. Data Collector (State Return) Component

Data Collector Component interacts Scenario Gathering and it is also the place where every data is being brought.

```
Data Collector (State Return)

# get_input_from_Scenario_Gathering():
# get_agentBehaviour():
# get_agent_data():
# generate_required_data(): return string
# initialize_agents():
# initialize_environment():
```

Figure 7.7: Data Collector (State Return) Component

Functions of Data Collector (State Return) component:

- **# get_input_from_Scenario_Gathering():** In order to create selected simulation environment, this component works with Scenario Gathering component. It takes outputs of Scenario Gathering in order to initialize agents, and the environment.

- **# get_agent Behaviour():** While creating scenario, this component takes needed behaviours of agents from REPast by using this function.

- **# get_agent_data():** While creating scenario, this component takes needed data of agents from REPast by using this function.

- **# generate_required_data():** In order to keep track of agents conditions and states, Data Collector component of framework interacts with OpenMap and REPast all the time and act according to input that comes from them. After taking new states and conditions, Data Collector component uses this function to generate new data.

- **# initialize_agents():** Data Collector component uses this function in order to initialize agents, after deciding on which scenario to be in use.
# initialize_environment(): Data Collector component uses this function to initialize environment, after deciding on which scenario to be in use.

### 7.1.8. Condition Management Component

This component is responsible for changing the agents’ and the environment’s condition.

![Condition Management](image.png)

**Figure 7.8: Condition Management Component**

Functions of Condition Management component:

- **# get_agent_condition():** By using this function, Condition Management component of framework gets current agent information from Data Collector component.

- **# get_environment_condition():** By using this function, Condition Management component of framework get current environment information from Data Collector component.

- **# update_agent_condition():** After getting current conditions of agents, our component will decide on its further condition. By using this function, it updates agents’ conditions.

- **# update_environment_condition():** After getting current conditions of environment, our component will decide on its further condition. By using this function, it updates environment's condition.
7.1.9. Graphical Management Component

This component consistently renders the environment and agents according to the information taken from REPast and OPENMAP.

<table>
<thead>
<tr>
<th>Graphical Management</th>
</tr>
</thead>
<tbody>
<tr>
<td># render_agent():</td>
</tr>
<tr>
<td># render_environment():</td>
</tr>
<tr>
<td># get_input_fromConditionManagement():</td>
</tr>
<tr>
<td># give_output_toConditionManagement():</td>
</tr>
</tbody>
</table>

*Figure 7.9: Graphical Management Component*

Functions of Graphical Management component:

# render_agent(): Graphical Management component of framework uses this function to display agents on simulation environment.

# render_environment(): Graphical Management component of framework uses this function to display environment.

# get_input_fromConditionManagement(): By using this method, this component gets inputs from Condition Management component about conditions of our agents, environment and etc.

# give_output_toConditionManagement(): By using this method, this component gives its output to Condition Management.

7.1.10. Navigator Component

This component takes user commands for navigating the ship.
Functions of Navigator component:

# get_input_from_user(): This component takes user input by this method.

# give_output_toGraphicalManagement(): After taking user input, this component gives new conditions to Graphical Management component.

7.1.11. Information Gathering Component

Information Gathering component aims to get GIS information, agents’ states and agents’ conditions.

Functions of Information Gathering component:

# get_GIS_info(): By this method, Information Gathering component takes GIS information from OpenMap.
# get_agent_state(): By using this method, this component takes agents' states from REPAST.

# get_agent_condition(): Information Gathering component gets agents' conditions from REPAST toolkit by using this function.

# report_error(): In order to give feedback to Error Management component, this component uses this method.

### 7.1.12. Event Scheduler Component

Event Scheduler component aims to organize concurrent movements of the agents while states or conditions of the agents are changing.

![Event Scheduler](image)

**Figure 7.12: Event Scheduler Component**

Functions of Event Scheduler component:

# get_agent_state(): In order to organize concurrent movements of the agents, Event Scheduler component gets agents' states from Information Gathering component by using this method.

# get_agent_condition(): In order to organize concurrent movements of the agents, Event Scheduler component gets agents' condition from Information Gathering component by using this method.

# create_agent_behaviour(): After getting information about agents' states and conditions, this component creates agents' new behaviour by using this function.
# report_error(): In order to give feedback to Error Management component, this component uses this method.

## 7.1.13. Agent Creation and Modeling Component

Agent Creation and Modeling component aims to create agents or modeling the agents which are already exist.

![Agent Creation and Modeling](image)

**Figure 7.13: Agent Creation and Modeling Component**

Functions of Agent Creation and Modeling component:

- # get_input_from_EventScheduler(): By using this method, this component gets input from Event Scheduler method.

- # get_input_from_informationGathering(): By using this method, this component gets input from Information Gathering method.

- # create_agent(): After getting inputs from various components of framework, this component creates necessary agent classes by using this method.

- # report_error(): In order to give feedback to Error Management component, this component uses this method.
7.1.14. Framework Provider Component

Framework Provider component aims to provide the agents which are already created or modeled by Agent Creation and Modeling component to BRIDGE COMMAND.

![Framework Provider](image)

**Figure 7.14: Framework Provider Component**

Functions of Framework Provider component:

- # give_agent_to_BridgeCommand(): By this method, our component gives information about agents position and informations to Bridge Command, in order to create simulation environment.

- # report_error(): In order to give feedback to Error Management component, this component uses this method.

7.1.15. Error Management Component

Error Management component aims to detect any crash(es) occurs in the REPAST and tries to fix this crash by activating second REPAST.

![Error Management](image)

**Figure 7.15: Error Management Component**
Functions of Error Management component:

# get_error_data(): By using this function, our component takes input from failures in any part of other components.

# activate_second_REPAST(): In crashing problems, this component provide a second running simulation game to its user in order to get rid of this problem. Though this will be optional.

# fix_error(): By using this method, Error Management component fix any errors that can happen.
7.2. Class Diagram

![Overall Class Diagram](image)

Figure 7.16: Overall Class Diagram
8. Libraries and Tools

8.1. 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

8.2. Toolkit for Geographic Information

Name of the Software: OpenMap
Version Number: 4.6.5
Mnemonic: 22.7 MB
Source: http://openmap.bbn.com/cgi-bin/license.cgi

8.3. Ship Simulation Programme

Name of the Software: Bridge Command
Version Number: 4.2
Mnemonic: 275 MB
Source: http://www.bridgecommand.co.uk/download.php
9. Time Planning (Gannt Chart)

9.1. Term 1 Gantt Chart
Figure 8.1: Gantt Chart Term 1
9.2. Term 2 Gantt Chart

Figure 8.2: Gantt Chart Term 2
10. Conclusion

This document is prepared to explain the detailed design of the project ABVCSF taken by our group, Küp Şeker.

At the earlier steps of the project design, it is implied that what our problem is and what its scope and purpose will be. After that components of the project have been researched. Since our design will be different from others that are in the market currently by means of toolkits used, each toolkit has been tested and explored. It is considered that which parts of which toolkit will help us. Moreover, it is discussed that what our design should do, what features it will provide, which components it will include, what the user interface will look like and so on.

After when we get an initial idea about project, we get into more detail about system architecture and data models. Each module of the system is described. Constraints of the project is discussed. Afterwards we provide a brief and detailed information of those modules of the system, their functionalities and etc. Afterwards, we talk about user interface in detail. We give screenshots from our newly designed user interface.

Finally, as a plan for the future through the end of the semester, all aspects of the project would be agreed on. Therefore, we will focus on the prototype that we will present together with the user interface design of the project. It will be a good practice and motivation to make a prototype of project for us when we start the implementation part in the second semester since it will consist new topics for all of us such as artificial intelligence and developing simulation framework.