Middle East Technical University

Ceng 491
Software Requirements Specifications

Massively Multiplayer Online Role Playing Game Project
Virtual Turkey

MECAC
Cinar Kilcioglu
Mert Degirmenci
Umit Cavus Buyuksahin

December 5, 2010
4.1.2 Relationships ......................................................... 22
4.1.3 Complete Data Model ............................................... 23

5 Behavioral Model and Description 23
   5.1 Description for Software Behavior ............................... 23
   5.2 State Transition Diagram .......................................... 24

6 Planning 26
   6.1 Team Structure .................................................. 26
   6.2 Estimation (Basic Schedule) ..................................... 26
   6.3 Process Model .................................................... 26

7 Conclusion 28
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Virtual Turkey</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Sample Quest</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>General View of the System</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Client Software Use Cases</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Server Software Use Cases</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Sign-Up Use Case</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Play Case</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>Sign-Up Use Case</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Manage Accounts Use Case</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Sign-Up Use Case</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>Manage NPCs Use Case</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>The Complete Data Model</td>
<td>18</td>
</tr>
<tr>
<td>13</td>
<td>Quest Data Object</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>NPC Data Object</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>Character Data Object</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Vehicle Data Object</td>
<td>20</td>
</tr>
<tr>
<td>17</td>
<td>Account Data Object</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>Treasure Data Object</td>
<td>22</td>
</tr>
<tr>
<td>19</td>
<td>Entity Relationship Diagram</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>State Transition Diagram</td>
<td>24</td>
</tr>
<tr>
<td>21</td>
<td>Game Architecture</td>
<td>25</td>
</tr>
<tr>
<td>22</td>
<td>The Spiral Model</td>
<td>26</td>
</tr>
<tr>
<td>23</td>
<td>Gantt Chart</td>
<td>27</td>
</tr>
</tbody>
</table>
1 Introduction

This document contains the software requirements for “Virtual Turkey” which is a massively multiplayer online role playing game (MMORPG). The approach used in this specification is adapted from IEEE recommended practices [1]. This document also abides the standards presented [2]. MECAC, the project group, assumes full responsibility of the requirements presented in this document.

1.1 Problem Definition

Turkey has variety of places with cultural inheritance to visit. In order to bolster touristic interest in Turkey, we will develop an online multiplayer game which will provide culture enthusiasts around the world the opportunity to preview sightseeing places in Turkey. To meet the requirements of an MMORPG, the game being developed is expected to support thousands of players concurrently. The project will later be published under GNU license when it reaches the maturity of other well-known open source MMORPGs.

1.2 Purpose

This software requirements specification intends to provide complete description of all requirements of “Virtual Turkey”. The requirements suggested in this document will serve as a guideline throughout the development process of this project. The end-product will be tested against the requirements to ensure the quality of the software produced.

1.3 Scope

This document addresses the functionality, external interfaces, performance, attributes, and design constraints of the MMORPG to be developed. However, the requirements presented in this document does not impose any design or implementation details.

1.4 User and Literature Survey

There are some open-source MMORPG projects currently available. Most popular and well-known open-source project is WorldForge[3].

The WorldForge project aims to produce an open source framework for MMORPG. The intent of the project is to create a widely used development framework and set of libraries by motivating interested developers.
Besides WorldForge, there are many MMORPGs that can be played free of charge online such as Allegiance\textsuperscript{[4]}. It provides a mix of real-time strategy and player piloted space combat gameplay. Initially developed by Microsoft Research, the game was later released under a shared source license in 2004\textsuperscript{[5]} and is now maintained and developed by volunteers\textsuperscript{[6]}.

Despite numerous open-source and free games released, a commercial MMORPG is still dominating the market. In April 2008, World of Warcraft (WOW) was estimated to hold 62\% of the MMORPG subscription market\textsuperscript{[7]}. Although this ratio may have slightly changed during the last two years, WOW is still the most popular MMORPG.

As mentioned in the problem definition, potential users of “Virtual Turkey” are the people interested in Turkey’s monuments, and the culture enthusiasts.

1.5 Definitions and Abbreviations

\begin{tabular}{ll}
CPU & Central Processing Unit \\
DoS & Denial of Service \\
GNU & GNU is not Unix \\
GPU & Graphics Processing Unit \\
GUI & Graphical User Interface \\
IP & Internet Protocol \\
MMORPG & Massively Multiplayer Online Role-Playing Game \\
MPI & Message Passing Interface \\
NPC & Non-Player Character \\
RPM & Revolutions per minute \\
RSA & Rivest, Shamir and Adleman \\
SRS & Software Requirements Specifications \\
\end{tabular}

1.6 References


[2] CSS.06 - *Yazılım Gereksinimleri Belirtimi Standardı*. Rev.7.0

1.7 Overview

The report contains seven sections. First section introduces the project “Virtual Turkey”. In the second section, the game is explained with a high-level perspective. The details of the project is started to be given in section 3. This section explains the software requirements to a degree that enables designers to design this system, and testers to test it. Domain for the software and behavioral model is described in section 4 and 5, respectively. Finally, planning of the project is presented and the report is concluded.
2 Overall Description

2.1 Product Perspective

The MMORPG project, Virtual Turkey, mainly serves the purpose of introducing the worth seeing monuments all around Turkey. The map of Virtual Turkey with missions is given in Figure 1. The player will be traveling the country in order to collect coins and the gold scattered over some secret places, as it can be seen from Figure 2. While visiting and learning about the invaluable historical places of Turkey to carry out the quests, the player will have more chance to gain treasure; however, the treasure has to be hidden due to non-transportable nature of it, which makes the game more mysterious and riveting. Each player is planned to have several attributes to make the game-play more realistic. Player needs these attributes in order to use the quest system, the player-class system and the trade system. For instance, a player might need an item in order to complete a quest, or a gold to trade.

Virtual Turkey is an independent and totally self-contained system. The major parts of the system are clients and the server. Components of the server are NPCs, physics engine, network and database which holds the client information. Clients also have physics engine, game loop and the communication layer. Clients and the server will communicate periodically to initiate and progress the MMORPG. Block diagram, showing the major components, is shown in Figure 3.
2.2 Product Functions

Actors of the main functions in the system are the user and the administrator. Users has access to the clients software, and the administrator has access to the server software.

User has three main use cases in the client software, which are shown in Figure 4.
Administrator has three main use cases in the server software, as shown in Figure 5.

![Server Software Use Cases Diagram](image)

**Figure 5: Server Software Use Cases**

Below, each use case is explained in detail.

2.2.1 User Use Cases

Use Case: Sign-Up

![Sign-Up Use Case Diagram](image)

**Figure 6: Sign-Up Use Case**

**Diagram:**

**Brief Description:** This use case describes how the user interacts with the client software to create an account on the server database.

**Initial step-by-step description:**

1. User runs the client component of MMORPG.
2. User clicks the “Sign-up” button.
3. User fills the form for user-id, password, and e-mail.
4. If the user-id does not exist in server database the user account is created.
Use Case: Play

![Diagram of User leading to Play]

Figure 7: Play Case

**Diagram:**

**Brief Description:** This use case describes how user logs-in to the system, plays the game and logs-out.

**Initial step-by-step description:**

1. User runs the client component of MMORPG.
2. User clicks Log-in button.
3. User fills the form for user-id and password.
4. If the server authenticates the user-id and password, the user proceeds to the next step.
5. User clicks “join-game” button, and plays the MMORPG.
6. User clicks log-out button and disconnects from the server.

Use Case: Update

![Diagram of User leading to Update]

Figure 8: Sign-Up Use Case

**Diagram:**
**Brief Description:** This use case explains how the user updates his account information.

**Initial Step-By-Step Description:**

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. If the user id authenticated, he clicks the “update” button.
4. User updates his information such as username, password, e-mail address.
5. User clicks “log-out” button and disconnects from the server.

### 2.2.2 Administrator Use Cases

**Use Case: Manage Accounts**

![Diagram](image)

**Diagram:**

**Brief Description:** This use case describes how administrator manages accounts of users.

**Initial step-by-step description:**

1. Administrator logs in to the server computer.
2. Administrator clicks “Accounts” button in server GUI.
3. GUI shows the accounts in tabular form with associated actions.
4. Administrator updates accounts and clicks “commit” button.
5. Administrator logs out from the server computer.
Use Case: Monitor Network Traffic

Diagram:

**Brief Description:** This use case explains how the administrator monitors the network traffic.

**Initial step-by-step description:**

1. Administrator logs in to the server computer.
2. Administrator clicks “Monitor” button in the server GUI.
3. GUI shows number of online players, network overhead on the server, and the communication delays in clients.
4. Administrator views the status of network traffic.
5. Administrator logs out from the server computer.

Use Case: Manage NPCs

Diagram:

**Brief Description:** This use case explains how the administrator manages non-playing characters.
**Initial step-by-step description:**

1. Administrator logs in to the server computer.
2. Administrator clicks “Manage NPCs” button in the server GUI.
3. GUI shows non-playing characters and their status in a tabular form.
4. Administrator updates NPCs and clicks “commit” button.
5. Administrator logs out from the server computer.

### 3 Specific Requirements

This section describes all the software requirements to a level of detail sufficient to enable designers to design and testers to test a system which satisfies this specification.

#### 3.1 Interface Requirements

##### 3.1.1 User Interfaces

There are two user interfaces of “Virtual Turkey”. These are the client interface for the players and the administrator interface for managing the MMORPG. Following sections describe the logical characteristics of each interface.

**Client User Interface:** Clients will interact with the system by means of a personal computer. After deployment of the client software to users personal computer, the user will be able to connect the persistent world of “Virtual Turkey” and interact with other clients. The users of the game will also be able to sign-up through the client software.

**Administrator Interface:** The administrator user interface will enable interaction between the administrator of the game and the software. The graphical user interface will allow the administrator to monitor the network traffic, online players, malfunctions within the game logic and the client software.

##### 3.1.2 Hardware Interfaces

The client and the server components of the software has different hardware interfaces. Following paragraphs describe the hardware interfaces for the two components.
**Client Hardware Interface:** The client component of MMORPG will rely on CPU and GPU to leverage the computational demand of networking, physics engine, and graphics of the game. The CPU will belong to x86 family of instruction set architectures. The target client software is intended to operate on 64-bit processor with 32-bit backward compatibility. For optimal experience for players, client software will try to offload the graphics and physics calculation to GPU whenever possible. Client software will also require Ethernet card to communicate with the server.

**Server Hardware Interface:** The server-side software will be interfacing with an x86 processor to leverage network and database tasks. For the clients to have seamless experience with MMORPG the software should interface at-least 15000 RPM disk drive and internet connection speed of 100 Mbit/second or more.

### 3.1.3 Software Interfaces

Both the client and the server component of MMORPG will be an application for Windows NT family of operating systems. As both components will be developed with C# programming language, Microsoft’s .NET 4.0 software framework will be needed. XNA 4.0 runtime libraries will be used for client-side graphics computation. The physics engine of client will be leveraged by JigLibx which is specifically designed for XNA. For network communication, the client and the server software will depend on LidGren library.

### 3.1.4 Communications Interfaces

The interaction between server and client will be maintained on TCP channel. All other communications will be carried out on shared memory. The components within the server will use MPI to communicate over shared memory.

### 3.2 Functional Requirements

Each major function along with data flow and requirements of it will be presented in this section.
3.2.1 Sign-Up

**Description:** This function of the system specifies how user signs-up to MMORPG.

**Basic Data Flow:**

1. User runs the client component of MMORPG.
2. User clicks “Sign-up” button.
3. User fills the form for user-id, password, and e-mail.
4. The user account is created.

**Alternative Data Flow 1:**

1. User runs the client component of MMORPG.
2. User clicks “Sign-up” button.
3. User fills the form for user-id, password, and e-mail.
4. The user id exists in the database. Server denies the clients request.
5. The GUI shows an error message asking for user to select different user-id.
6. The sign-up user interface goes back to (3).

**Alternative Data Flow 2:**

1. User runs the client component of MMORPG.
2. User clicks “Sign-up” button.
3. User fills the form for user-id, password, and e-mail.
4. Password length is less than 6 characters. Server denies clients request.
5. The GUI shows an error message asking for user to select longer password.
6. The sign-up user interface goes back to (3).

**Functional requirements:**

1. REQ-1: Server should not accept the same user-id twice when signing-up users.
2. REQ-2: Server should not accept the passwords shorter than 6 characters when signing-up users.
3.2.2 Play

**Description:** This function specifies the process of user logging-in, playing the game, and the signing out process. It does not however, specify any content within the game logic, nor does it specify the graphics within the game.

**Basic Data Flow:**

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills the form for user-id and password.
4. Server authenticates the user-id and password, and the user proceeds to next step.
5. User clicks “join-game” button, and plays the MMORPG.
6. User clicks “log-out” button and disconnects from the server.

**Alternative Data Flow 1:**

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills the form for user-id and password.
4. Server does not authenticate the user-id and password entered.
5. The log-in user interface goes back to (3).

**Alternative Data Flow 2:**

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills the form for user-id and password.
4. Server authenticates the user-id and password, and the user proceeds to next step.
5. User clicks “join-game” button, and plays the MMORPG.
6. User loses its network connection while playing MMORPG.
7. Server logs out the user from the game, and saves-off its state.

**Functional requirements:**

1. REQ-3: Server should check the user-id and password entered by the user.
2. REQ-4: Server should log-out the user automatically which has disconnected from the network.
3.2.3 Update

*Description:* This function explains how the user updates his/her account information.

*Basic Data Flow:*

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills the form for user-id and password.
4. Server authenticates the user, and the user clicks the “update” button.
5. User updates his/her information such as username, password, e-mail address, and the server accepts the request.
6. User clicks log-out button and disconnects from server.

*Alternative Data Flow 1:*

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills it user-id and password form.
4. Server does not authenticates the user, and the log-in screen goes back to (3).

*Alternative Data Flow 2:*

1. User runs the client component of MMORPG.
2. User clicks “Log-in” button.
3. User fills the form for user-id and password.
4. Server authenticates the user, and the user clicks the “update” button.
5. User updates his/her information, and clicks the “update” button.
6. Server does not accept the request, showing the user the reason.
7. User clicks “log-out” button and disconnects from the server.

*Functional requirements:*

1. REQ-5: Server should check the new user-id entered by the user in case it already exists.
2. REQ-6: Server should check the new password in case it is shorter than 6 characters.
3. REQ-7: Server should check the new e-mail to verify that its a valid e-mail address.
3.2.4 Manage Accounts

Description: This function describes how administrator manages accounts of the users.

Basic Data Flow:

1. Administrator logs in to the server computer.
2. Administrator clicks “Accounts” button in server GUI.
3. GUI shows the accounts in tabular form with associated actions.
4. Administrator updates accounts and clicks “commit” button.
5. Administrator logs out from the server computer.

Alternative Data Flow 1:

1. Administrator logs in to the server computer.
2. Administrator clicks “Accounts” button in the server GUI.
3. GUI shows the accounts in a tabular form with associated actions.
4. Administrator updates accounts and clicks “commit” button.
5. Server rejects the changes explaining the administrator the error occurred.
6. The GUI goes back to (2).

Functional requirements:

1. REQ-8: Server GUI should not accept two changes at a time to prevent malicious action of the administrator.

3.2.5 Monitor Network Traffic

Description: This function of the server software specifies how the administrator interacts with the system to monitor the network traffic.

Basic Data Flow:

1. Administrator logs in to the server computer.
2. Administrator clicks “Monitor” button in server GUI.
3. GUI shows number of online players, network overhead on the server, and the communication delays in clients.
4. Administrator views the status of the network traffic.
5. Administrator logs out from the server computer.
**Functional requirements:**

1. **REQ-9:** Server GUI should show the number of online players in the monitor panel.
2. **REQ-10:** Server GUI should show the communication delays for each user currently logged into the system.

### 3.2.6 Manage NPCs

**Description:** This function specifies how the administrator manages non-player characters.

**Basic Data Flow:**

1. Administrator logs in to the server computer.
2. Administrator clicks “Manage NPC” button in server GUI.
3. GUI shows non-player characters and their status in a tabular form.
4. Administrator updates NPCs and clicks “commit” button.
5. Administrator logs out from the server computer.

**Functional requirements:**

1. **REQ-11:** Server GUI should enable administrator to add/remove non-player character to the game.
2. **REQ-12:** Server GUI should enable administrator to change the status or the location of a non-player character.

### 3.3 Non-Functional Requirements

#### 3.3.1 Performance Requirements

In order for “Virtual Turkey” to be massively multiplayer, the server should support 1000 users concurrently. The delay of communication between server and the user should not exceed 0.1 second. The database transaction should not consume more than 10

In order for the MMORPG to be maintainable, the GUI of the server component should be robust for the administrator to update information or monitor the network traffic. The administrator should be able to log-on to the server within 5 seconds even when the server supports a thousand players online. When the administrator clicks “Manage Accounts” or “Manage NPCs” buttons, the GUI should show the requested information within 10 seconds and commit the changes within a minute. Monitoring network traffic should not increase the overhead on server-side as such information can directly be computed on-the-fly. When
the administrator clicks monitor network traffic, the GUI on the server component should show the current traffic summary within 10 seconds.

3.3.2 Design Constraints

Security: The communication between the server and the client component of MMORPG should be encrypted with RSA algorithm. Server should also check the cheating case with offloading major physics calculations to server-side. In order to prevent abuse of the server component, software on the server of MMORPG should block the recurring requests from the same IP address. Server should log common proxy addresses as well to check if the client is performing DoS attack.

Software System Attributes:

1. Maintainability: The modification of the source code should be disabled. The extensions should be applicable directly without modification to the back end. The patching service should be regarded as a different component in case extensions would be offloaded to a different server.

2. Portability: Both the client and the server component of the software should be portable. The target hardware platform for client component is unknown. Only requirement for the client software to be deployed is the operating system that belongs to Windows NT family. The server side of MMORPG should be independent of specific hardware or software configuration.

3. Scalability: Most important software system attribute of “Virtual Turkey” is its scalability. Deploying additional servers and dividing the persistent world should not require additional extension on the client software.

4 Data Model and Description

4.1 Data Description

The complete data model is given in Figure 12. The focus of the model is on Character and NPC data objects. Account, Vehicle, and Treasure data objects are associated with Character object. NPC object, on the other hand stores the information about the trade items of it. Non-playing characters has no associated vehicles or accounts as it can be observed from the figure. Following sections describes all the data objects and their major attributes in more detail.
Figure 12: The Complete Data Model
4.1.1 Data Objects

In this section a brief description of each data object is given. For each data object, function and semantics associated with it are summarized. This section also describes major attributes of data objects.

**Quest**  This data object represents quest given by non-player characters. Database needs to store the identifier of the quest which is only major attribute of it.

**id:**  The unique identifier of the quest data object.

**NPC**  The NPC data object - as its name suggests - represents a single non-player character in “Virtual Turkey” MMORPG. Besides its unique identifier, it has three major attributes namely position, owner, and type.

**id:**  The unique identifier of the NPC data object.

**owner:**  The owner character of the NPC data object.
**Character**  This data object represents the character of the user associated with the account. The user character may have associated Vehicle or Treasure data objects.

- **name:** The unique identifier of the Character data object.
- **account:** References corresponding Account data object.
- **last played:** The time that the character last played “Virtual Turkey”.
- **created on:** The time that the character has been created.
- **position:** The location of the character in the persistent world of MMORPG.
- **quests completed:** The quests that the character has completed successfully.
- **quests in progress:** The quests that the character has attempted to complete.
- **is online:** The boolean attribute showing whether the character is connected to persistent world or not.
- **vehicle:** The current vehicle of the character that references a Vehicle data object.

**Vehicle**  The current vehicle of the character that references a Vehicle data object.

Software Requirements Specifications for Virtual Turkey 20
**Vehicle**  This data object represents the vehicle in “Virtual Turkey”. Only characters can use the vehicles in the MMORPG to transport their character between different locations of the persistent world. Each vehicle can be associated with only one character.

- **id:** The unique identifier of the Vehicle data object.
- **character:** References the corresponding character object that owns the vehicle.
- **position:** The current location of the Vehicle in the persistent world of MMORPG.

![Account Data Object](image)

**Account**  This data object stores the account information of players. The account data object does not store the character information. However, it references a character data object. This approach enables players to have multiple characters in MMORPG.

- **e-mail:** The email address of the account owner.
- **password:** The password of the account owner.
- **name:** The nick name chosen by the account owner.
- **address:** The address of the account owner. This attribute is optional.
- **last signed on:** The last date when the account holder has logged-in.
- **created on:** The date when the account was created.
- **online:** Boolean attribute showing if the account owner is connected.
- **confirmation code:** The confirmation code for account owner.
- **confirmed:** Boolean attribute showing whether the account was confirmed or not.
Treasure

This data object represents the treasure items that the characters poses. The treasure data objects have associated value and position in the virtual world. This data objects are intended to be traded for items by characters.

- **id:** The unique identifier of the treasure data object.
- **value:** Value of the treasure data object.
- **position:** The location of the treasure data object in the persistent world of MMORPG.

### 4.1.2 Relationships

This section describes the relationship between the data objects described in the previous section.

**NPC - Quest:** Each NPC may be associated with one or more quests. Those quests are to be completed by characters. However, a quest may not be associated with multiple NPCs.

**NPC - Item:** Each non-playing character may own one or more items. However, an item can not be owned by multiple non-playing characters. NPCs are able to trade the items with playing characters.

**Character - Item:** Each character may own one or more items. However, an item can not be owned by multiple characters. Characters can trade the items and treasures with other characters or NPCs.

**Character - NPC:** Each non-playing character is owned by only one character. An NPC without an owner or a character without an NPC can exist.

**Character - Vehicle:** Each character can only own one vehicle in the MMORPG. A vehicle can not be associated with multiple characters.

**Character - Account:** Each character can be associated with at most one account. However an account owner may own multiple characters.
**Character - Treasure:** A character may collect zero or more treasure, however a treasure can only be owned by a single character.

### 4.1.3 Complete Data Model

This section describes a complete data model merging data object descriptions with relationships explained in previous sections. Figure 19 shows the entity relationship diagram of the data model which provides a conceptual representation of data. Major data objects are NPC, quest, item, treasure, character, account, and vehicle. The relationships between data objects provide both semantical and relational information.

![Figure 19: Entity Relationship Diagram](image)

### 5 Behavioral Model and Description

#### 5.1 Description for Software Behavior

The game has two main parts, clients and server. The relationship between these two parts is accomplished by package transferring between each part’s own communication layer.

Game loop, which belongs to clients component, interacts with NPC part of the server component and during the interaction, information is passed to NPC through communication.
layers. Information can be gain gold, next position of character, vehicles etc. It also initiates the game to start.

Database inside the server component stores the information such as account information of the users, last position of the players, and their attributes. It also has a connection with server physics engine, which enables the physical system of the game.

### 5.2 State Transition Diagram

![State Transition Diagram](image)

Figure 20: State Transition Diagram
Figure 21: Game Architecture
6 Planning

6.1 Team Structure

“Virtual Turkey” Project Team does not have any team leader that directs the project. The power of the team comes from the motto “Individually, we are one drop. Together, we are an ocean”. Each member of the team puts all of his effort to the project.

The team has weekly meetings with Assoc. Prof. Veysi Isler, who is the advisor of the project, at MODSIMMER. The team also meets with Research Assistant Umit Rusen Aktas to discuss about the progress of the project. Members of the team study the topics that have been determined before with fully participation every Tuesday and Thursday.

6.2 Estimation (Basic Schedule)

The plan is to prepare the detailed design report and prototype demo till the end of the semester. Second semester mainly will be based on configuration management, system design development, and system testing.

The detailed schedule of the project is given in Figure 23.

6.3 Process Model

The spiral model is used as a software development process, which combines elements of both design and prototyping-in-stages. The project team benefits from the spiral model’s ability to combine top-down and bottom-up concepts. The spiral model used in the project is given in Figure 22.

Figure 22: The Spiral Model

http://en.wikipedia.org/wiki/Spiral_model
<table>
<thead>
<tr>
<th>Number</th>
<th>Task</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Management</td>
<td>2/10/2010</td>
<td>2/10/2010</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Web Page</td>
<td>12/12/2010</td>
<td>27/5/2011</td>
<td>167</td>
</tr>
<tr>
<td>9</td>
<td>Server - Client Analysis</td>
<td>29/11/2010</td>
<td>11/12/2010</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Database Analysis</td>
<td>29/11/2010</td>
<td>20/12/2010</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>Network Analysis</td>
<td>29/11/2010</td>
<td>11/12/2010</td>
<td>43</td>
</tr>
<tr>
<td>12</td>
<td>NFC Analysis</td>
<td>29/11/2010</td>
<td>7/1/2011</td>
<td>49</td>
</tr>
<tr>
<td>13</td>
<td>AI Analysis</td>
<td>29/11/2010</td>
<td>10/1/2011</td>
<td>52</td>
</tr>
<tr>
<td>14</td>
<td>System Configuration</td>
<td>15/2/2011</td>
<td>1/14/2011</td>
<td>46</td>
</tr>
<tr>
<td>15</td>
<td>Server Configuration</td>
<td>16/2/2011</td>
<td>22/2/2011</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>Database Optimization</td>
<td>11/2/2011</td>
<td>18/2/2011</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>AI Configuration</td>
<td>27/2/2011</td>
<td>1/4/2011</td>
<td>8</td>
</tr>
<tr>
<td>35</td>
<td>Database Test</td>
<td>7/5/2011</td>
<td>10/5/2011</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>Alpha Test</td>
<td>14/5/2011</td>
<td>16/5/2011</td>
<td>3</td>
</tr>
<tr>
<td>40</td>
<td>Finalization</td>
<td>21/5/2011</td>
<td>27/5/2011</td>
<td>7</td>
</tr>
<tr>
<td>41</td>
<td>Post Production</td>
<td>21/5/2011</td>
<td>27/5/2011</td>
<td>7</td>
</tr>
</tbody>
</table>
7 Conclusion

This analysis report shows software team’s approach to MMORPG problem. User interaction with the system is defined in addition to project constraints. Market research presented in this document covers popular open-source MMORPGs. Scheduling and timeline have also been specified in this document. This specification will hopefully constitute the basis for design, development, and testing of the project.