CSCE 431 - Project Software Requirements Specification

Team BMORPG (Barely Multiplayer Online Role-Playing Game)

Brandon Adame

James Freyman

Josue Martinez

Stephen Simmons

Luke Yeager

February 20, 2012

Jaakko Järvi, Instructor

Tolga Cifti, T.A.

1. Intro

1.1 Purpose

Our purpose is to create a game in which multiple users can interact in a competitive and social environment. The players will connect to a server that pairs clients together in a secure and reliable fashion. The server also calculates all in game interactions to facilitate fair game play.

The game is intended for players of age ten or more who are interested in a game with a strategic, competitive environment.

1.2 Scope

The game implements two primary software products: the Client and the Server.

The Client serves primarily as a GUI for interaction with the user. The client takes input from the user and transmits it to the server. It is not expected to store any character information, ensuring validity of information.

The Server connects all clients together and keeps track of multiple games simultaneously. Once paired, players rely on the server for in-game calculations, and when the game is over, the Sever notifies the Client that they may update their character statistics.

The game is primarily intended to provide place where multiple game players can battle each other and eventually increase their character's skill.

1.3 Definitions, acronyms, abbreviations

Client – Program running on the player's computer. The Client sends the Player's commands to the Server.

Server – Program running on the server. The Server is responsible for connecting Clients together to create a game, and for monitoring the game while in progress.

RPG (Role Playing Game) - A game in which players assume the role of their characters in a fictional setting. Players can increase their player attributes through integrations and battle in the virtual world.

MMORPG (Massively Multi-player Online RPG) - An RPG based in a virtual environment that allows a huge number of players to take part in the game.

2 Overall description

2.1 Product perspective

- System Interfaces - This product is self-contained and operates independent of any larger systems.

- User Interfaces - The client side application will be the main interface for users. It needs to include a page to customize characters, and the option to be matched to another player to battle. Once in a battle, users need to pick a move from a set list, or change their weapons according to what they have in their inventory. There may also be a feature that allows two battling users to chat, in which case a chat window will be needed to display messages and a field to take user input.
- Hardware Interfaces - The server will be constantly listening over a certain port waiting for clients to log in (exact port numbers will be determined at a later date). Once users are paired to battle, they will communicate with the server on a certain port. Each game thread will use a different port.
- Software Interfaces - This application is designed to run on a Windows operating system with .NET Framework 4 installed. We will use a SQL database to store player and game data.
- Operations - In the case that a user gets disconnected during a battle, he will forfeit the game. It is possible that a feature will be added in the future to allow a user to reconnect within a certain amount of time.

2.2 Product functions

- Network Connection - Throughout the game system, secure sockets will be used. We aim to design a system that cannot be easily hacked, but which offers real-world security. This is especially important when sending login passwords over the network.
- Client connection - When the Client program starts up, players have the choice to either create a new account or log-in to an existing account. When logging into an existing account, the Client simply sends their username and password to verify their identity, then the Server either accepts or denies the request. When creating a new account, the Client sends all the information necessary to specify a new account, and the Server creates the account, then notifies the Client.
- Pairing players - The Server will pair players together by looking at their levels and trying to create the fairest game possible from the list of currently connected Clients.
- Player characteristics - After each game, the Player has the option to upgrade their player characteristics. The amount of points they have to spend on new weapons and abilities depends on their level, the level of their opponent, and the outcome of the match.

2.3 User characteristics

This game is intended for all players of at least ten years of age with a computer and access to the Internet. There is no preexisting technical skills or knowledge required to play the game.

2.4 Constraints

This project is intended to be an academic exercise, demonstrating that we can design a system of programs with secure reliable network communication. We do not intend to develop a system which can handle a large number of concurrently connected clients. Therefore, we will be using a personal computer for the server rather than spending money on a production-level server.

- Regulatory policies - At most 130,000 players can simultaneously be participating in battles.
- Hardware limitation - The number of available ports limits the total number of players that can connect to the server.
- Interfaces to other applications - In addition to the server and client interfacing, the server portion of the code will interact with a SQL database.
- Parallel operations - Since each game is independent from the rest, the Server will allocate a new thread for each game. In this manner, each game will run concurrently and will not affect the run-time performance of any other game.
- Signal handshake protocols - We will be using TCP sockets to ensure reliability of data transfer. The handshake procedure will be clearly defined in documentation, with all network communication being sent as JSON objects.
- Reliability requirements - The client program needs to be able to handle the loss of connectivity to the server (server goes down/upgrading components). The server needs to handle the loss of connectivity to a client without crashing the game for those two players. The loss of a single game should not cause the entire server to fail.
- Safety and security considerations - The security of the server needs to be hardened and verified so that an attacker cannot deny service to clients, obtain control of the server, or gain access to the database of users.

2.5 Assumptions and dependencies

- The project assumes all code will be executed on a Windows platform with .NET Framework 4 installed as the client will be written in C# using Windows Presentation Foundation.
- Software also assumes access through any given firewall to allow for outgoing socket connections to the server.
- Client software requires there is an active server to connect to. Client has no standalone functionality.

3.0 Specific Requirements

3.1 External Interfaces

- Connection window
    - Allows the user to enter a username and password to log into the service. If the user does not have a pre-existing account, there will be a button available to create a new account. The button will launch a new window form with username and password text input boxes. After either the user enters their pre-existing username and password or creates a new one, the respective windows will disappear and the game window will launch.
    - For the duration of development, the window will also contain IP and port input text boxes for debugging usage. These will not be visible in the final iteration.
- Game Window

    - The game window will contain 4 interaction buttons on the bottom right of the screen consisting of: Attack, Defend, Special, Equip. Each button, when clicked, will spawn an item selection pane to the right of the buttons containing selectable text options. These options will relate to items, types of attacks, or types of armor to be selected.
    - There will be a file bar on the top of the window with dropdown menus as follows: File, Help.
    - File will contain an exit option that will terminate the web connection and close the program.
    - Help will contain an option that spawns a window containing instructions on how to play the game.
- User Profile Window
    - The user profile window will display core player statistics on the right side of the screen and a dynamic list of usernames that are available to play. The user can then click on a username and send an invite to play a game. If the user is selected, a popup window will appear asking the second player if they would like to accept the game connection. If the user clicks yes, the game window will launch and the game will start. If the user clicks no, the popup window closes and control is returned to the original profile window.
- Shop Window
    - The shop window will contain a list of items that can be purchased by the user. The user is allowed to purchase any item on the list so long as they have sufficient money. Money is a variable held by the server and displayed on the shop window screen, and it changes as the user purchases items.
- Level-Up Window
    - The level-up window allows the user to select specific upgrades to their base character upon reaching a preset level. These options will be dynamic and delivered by the server and displayed to the user. The user can then select their options, which will be sent back to the server.

3.2 Functions

- Client side application
    - The client side application will allow users to submit commands to the server, such as updating items, joining a game, or making a move during a game.
    - The client will allow a user to log in if he has an existing account, or create one if he doesn't.
    - The client will receive messages from the server and display it in the GUI so the user can see a visual representation of the game statistics.
- Server side application
    - The server side application will receive messages from clients, and process them depending on what function is sent.
    - The server will use a database to store up to date information about users and games.

- The server will receive a move message from a client, and analyze it to make sure the move is legal, and the player's statistics on the client match what is stored on the server.
- For each active game, the server will store the state of the game, and the current statistics of the two players.
- The server will declare a forfeit if a player is disconnected from the server, or if a client sends a message saying it quits.

3.3 Performance Requirements

Most performance requirements for this project are limited by the Internet connection between the connected computers. Here, we assume that communication between computers occurs in less than 0.5 seconds.

- The Server shall allow at least 20 connected Clients at any time, with up to 10 games running concurrently, while meeting all other timing requirements.
- When two or more Clients are connected, each Client shall be matched by the Server in no more than 15s.
- During the game, communication between the clients (chat messages, notification of actions, player disconnects, etc), shall occur in no more than 3s.
- Since the main load of data from the server will occur in game initialization, the Client should not require any network communication to decide which move to make next in-game. Thus, the action selection menus should have instant response time (< 0.1s).

3.4 Logical Database Requirements

- A database is needed to store the login credentials of all the players registered to the game.
- The same database is needed to hold the progress that is made by each player (such as experience and leveling up).
- It will also be used to track the "inventories" of the players so that they can have a persistent set of items at their disposal for use in-game.

3.5 Software System Attributes

- **Reliability**
  Each program should handle network connection issues robustly, with well-defined error messages and notifications. The server should have a reliable Internet connection such that Clients can remain connected to each other during gameplay.
- **Availability**
  The server will constantly be running, as it needs to always be available for clients to connect to and play. In the case that a client quits or gets disconnected, the game will end; the client may reconnect to the server and begin playing another game at any time.
- **Security**
  As previously discussed, secure (TLS / SSL) sockets will be used for network connections. In addition, Clients will be required to authenticate themselves with a username and password

before connecting to other Clients. In-game, cheating will be thwarted by cross-checking each requested action against the server's list of actions that player is authorized to perform.

- **Maintainability**
  The client will be notified if it is an older version than what is the currently "active" version, and will be given a path to obtain the newest version of the client. If a new version of the server is developed, then the server will have to go down for a period of time while it is updated, leaving all clients unable to connect.
- **Portability**
  The software for the server will only run on a Windows machine capable of running the C# code. The client has the same attributes; however, a Windows machine is not necessary to connect to the server, so if a client was made for another platform (and was functionally identical to the Windows client), that client could coexist with the Windows client.