**Software Requirements Specification**

**for**

**Super Tic Tac Toe**

**Version 1.1**

**Prepared by Adam Bilby, Haseeb Amin, Garrett Clement, Young-Jin Choe**

**2/6/2017**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1.** **Introduction 1**

1.1 Purpose and Scope

1.2 Definitions, Acronyms and Abbreviations

1.3 References

1.4 Overview

**2.** **Overall Description 1**

2.1 Product Functions

2.2 User Classes and Characteristics

2.3 Constraints

2.4 Assumptions and Dependencies

**3.** **Specific requirements 2**

3.1 External Interface Requirements

3.1.1 User Interfaces 2

3.1.2 Hardware Interfaces 2

3.1.3 Software Interfaces 2

3.1.4 Communications Interfaces 3

3.2 Functional requirements

3.2.1 Functional requirement 3

3.2.2 Functional requirement 3

3.3 Performance Requirements

3.4 Software System Attributes

3.4.1 Availability 4

3.4.2 Security 4

3.4.3 Maintainability 4

3.5 Design constraints

3.5.1 Standards Compliance 4

3.5.2 Hardware Limitations 4

**Appendix A: Issues List 4**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Adam | 2/7/2017 | Worked on introduction, definitions, product features, user classes and characteristics, and constraints. | 1.0 |
| Haseeb | 2/8/2017 | Edited sections 1.4, 3.5 | 1.0 |
| Adam/Garrett | 2/8/2017 | Worked on functional requirements section. | 1.0 |
| Adam | 2/8/2017 | Worked on Use Case Diagram | 1.0 |
| Adam | 2/8/2017 | Worked on Use Case Diagram | 1.1 |
| Garrett | 2/10/2017 | Drew GUI Diagrams | 1.1 |
| Garrett | 2/16/2017 | Worked on sections 3.1.1, 3.1.4 | 1.1 |
| Adam | 2/17/2017 | Made revisions to 2.1 and 3.2 | 1.1 |
| Choe | 2/19/2017 | Made revisions to 1.1 and 1.2 | 1.1 |
| Haseeb | 2/20/2017 | Made revisions to 1.4 and 3.5 | 1.1 |
| Choe | 2/20/2017 | Made revisions to 1.3, 2.4, and 3.4 | 1.1 |

# **Introduction**

## **Purpose and Scope**

Super Tic-Tac-Toe 1.0 is a puzzle game in which two agents compete to place their respective markers in a row either horizontally, vertically, or diagonally on a five-by-five grid. There will be two types of agents: human and artificial intelligence (AI). There are two game modes: stand-alone and network. The stand-alone mode is for a human on a web browser to play against an AI on the web server. The network mode will allow two servers to play against one another on a peer to peer network. This game does not allow player vs player gameplay.

## **Definitions, Acronyms and Abbreviations**

Player: A human or artificially intelligent agent who plays the game.

Artificial Intelligence (AI): The player is a part of the software that is capable of optimally planning the next move in the game.

Stand-Alone Mode: A game between a human and an AI player. This is a accomplished with a browser/server architecture.

Browser/Server Architecture: There will be a server and client program that can be launched from a launcher.

Network Mode: A game between two AI players. This is accomplished with a peer to peer architecture.

Peer to Peer (P2P): Two servers communicate directly with one another.

Markers: Players are represented on the gameboard by either an ‘X’ or an ‘O’ marker.

The Board: A 5x5 grid which can be filled by markers.

Turn: Players take turns placing markers on the board. The player with the ‘X’ marker always goes first. A turn has a time limit, and the results of a turn are displayed for a minimum time before the next turn is allowed to take place.

Game: A complete sequence of turns leading to a win condition or a draw.

Win Condition: A game is won whenever a player manages to place five of their markers in a row and without interruption either vertically, horizontally, or diagonally.

Draw/Double Loss: If no player manages to achieve the win condition, and no more moves are possible, then the game results in a double loss or draw.

## **References**

<https://learn.uco.edu/d2l/le/content/154871/Home>

SE Lecture Powerpoints

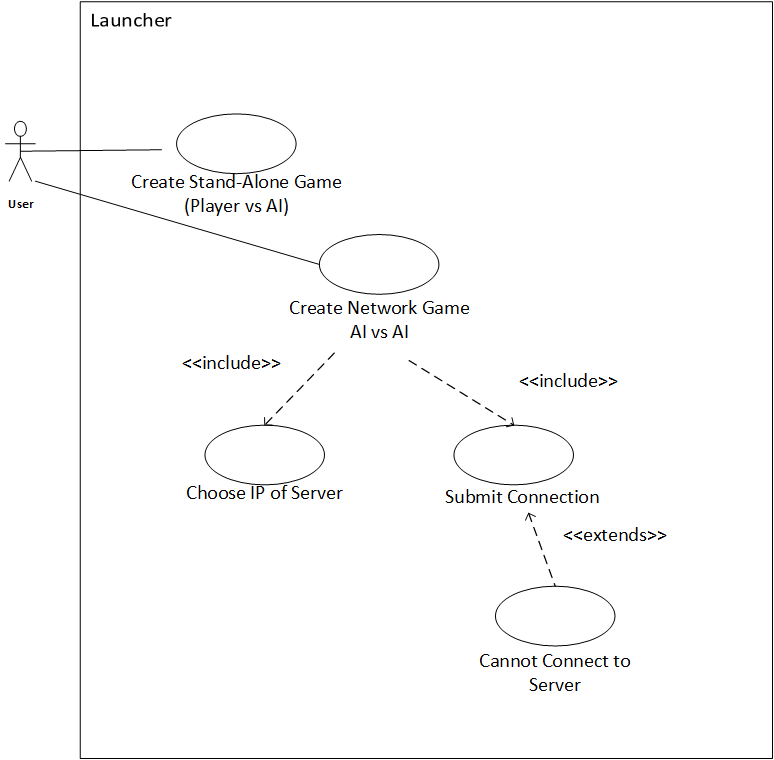
## **Overview**

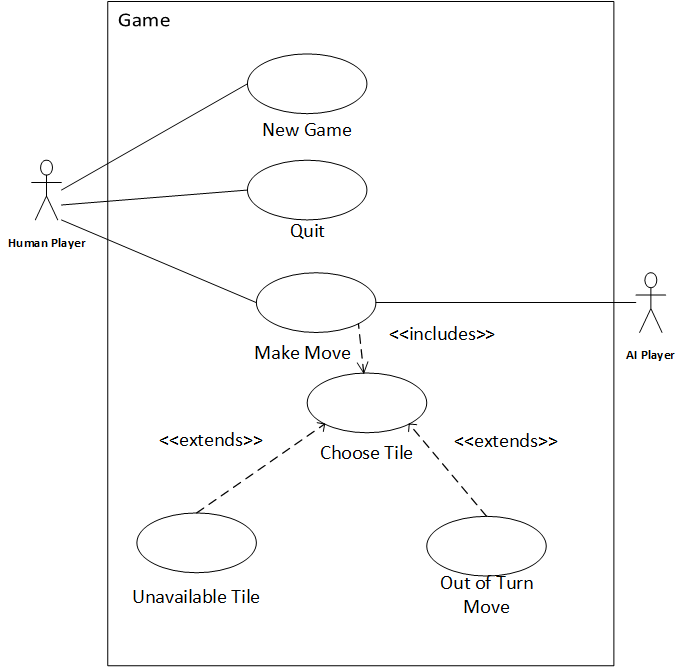
In the remainder of the document we will cover the overall description and the specific requirements. In the overall description we will discuss product functions, user classes and characteristics, constraints, and assumptions and dependencies. In the specific requirements we will discuss external interface requirements, functional requirements, performance requirements, software system attributes, and design constraints.

# **Overall Description**

## **Product Functions**

Super Tic-Tac-Toe allows for a user of the software to set up one of two game modes. The launcher is the first thing a user will see when they open the program. They will be presented with the choice of either launching a stand-alone game, which will take them straight to a game, or initiating a network game, which will ask them for an IP address to type in and submit. If this goes well a game between two AIs will be displayed. Otherwise an error will be displayed that the server was not connected too.





## **User Classes and Characteristics**

The typical user will be educated and computer literate. Special needs are not anticipated.

## **Constraints**

This product will be designed with Java SE 8 in mind, and so the only limitations will be the minimum requirements for this and the Netbeans 8.2 IDE. The developers are working with mid-market computers which exceed the needs of this project by far.

Our team will need to allow our servers to communicate with other team’s servers. This will require a standardized protocol for communicating game objects. Java will most likely be popular with the class which has readily available libraries for parsing XML and JSON packets sent over the internet. However we will be bound to the consensus no matter what.

## **Assumptions and Dependencies**

The communication protocol between team servers will either need an adapter built for it or our software will have to change. XML will be our first choice for over the internet communication as serializing objects with it is easy. The second choice for communicating between two AI is to use sockets.

# **Specific requirements**

## **External Interface Requirements**

### **User Interfaces**

The GUI will be created using Java’s built in swing libraries. The user will be required will be required to have Java installed on their computer to interface with the GUI.

The launcher will initially consist of two clickable buttons (see Figure A.). The Player vs AI button and the AI vs AI button. Once the AI vs AI button is clicked the launcher will display a IP address text box and connect button (see Figure B.). The user will be required to enter the IP address of the opponent in the textbox and click connect to play a AI vs AI game.

Error messages will be shown via java dialog boxes. (see Figure E.)

Figure A.) Figure B.)

initLauncher.pngAIvsAIPress.png

If the user clicks the Player vs AI button the Gameboard GUI (see Figure A) will be displayed. The GUI will consist of a 5x5 grid of clickable tiles, a new game button, and a quit button. The player will be able to click a tile on a the grid to make a move, given it is their turn and the move is viable. The New Game button will be used to reset the game board. Once the player is ready to quit he/she will simple click the quit button which will navigate them back to the launcher GUI (see figure C).

If the user clicks the AI vs AI button the Gameboard GUI will once again appear, but with slight modifications. (see Figure D.) The new game button will not appear and the player will not be able to interact with the grid tiles. The player will be able to quit the AI vs AI match at any time by pressing the quit button. The quit button will still navigate the user back to the launcher GUI.

Figure C.) Figure D.)

PvAGameBoard.png AvAGameBoard.png

Figure E.)

ErrorMessage.png

### **Hardware Interfaces**

The primary pieces of hardware needed for this software are the mouse and monitor. The mouse is used to interact with the user interface and interact with the game. The monitor will display the game and user interface. Standard monitors and mouses should do.

### **Software Interfaces**

This will be targeted towards the windows operating system.

The software will be developed using Java 8 SE and will use a variety of standard libraries to achieve this.

### **Communications Interfaces**

The program will use a client-server architecture for networking. The server will allow up to a total of two clients to pair with it at one time. There will be no security protocols built into the network communication.

We will use serialized XML game objects in order to share game objects with other servers. An adapter will be built to allow parsing of other software’s game objects so that our team’s servers can play against theirs.

## **Functional requirements**

### Artificial Intelligence

#### ***Introduction***

The goal of the AI is to optimally make the next move on their turn of the game for the purpose of achieving the win condition. If achieving the win condition is the ultimate goal of this planning, however if at some point that becomes impossible it will play towards double loss.

#### ***Inputs***

The AI receives the Game object. The game object has the state of the board, which is an array of characters with twenty five (25) places, either being blank, ‘X’, or ‘O’.

#### ***Processing***

The game object will be parsed. Once the game object is parsed the AI will examine the current state of the grid and based on the minimax algorithm determine the next best possible move.

#### ***Outputs***

<*Describes the outputs (destination, quantities, error messages, and the like).>*

If the AI receives a null game object it will create an exception indicating the possibility of connectivity issues.

### New Game

#### Introduction

The new game functionality is initiated when its corresponding button is pressed. This button will start a new game, even if a new game is currently active. This will be considered a loss.

#### Inputs

This initiates a brand new game, so the only input required is the user pressing the button. It will delete the old game’s state entirely.

#### Processing

If the current game exists, it will be deleted, and replaced with a new game.

#### Outputs

The new game will begin.

#### Introduction

The player uses their mouse to choose a tile. The player is capable of clicking and moving the mouse as much as they want, but the tiles are disabled when the player’s turn is not active. When the player chooses a tile which is empty on their turn, the symbol corresponding to them is placed in that tile.

#### Inputs

The position of mouse during the click relative to the tiles on the board.

#### Processing

If the tile is disabled due to it being the other player’s turn (the AI), or because it is already filled with a symbol, then nothing will happen.

#### Outputs

The empty tile will be filled with a symbol corresponding to the player. The player’s turn is over after this event.

### Create Stand-Alone Game

#### Introduction

The client will be responsible for sending and receiving information from the sever. For example, once the player has made a move the client will send information about the move to the sever. The client will also be responsible for parsing and formatting the data sent by the server and then relaying this information to the game so that it can adjust accordingly.

#### Inputs

Once the player makes a move the current state of the board will be sent to the client.

#### Processing

The client will than parse the information that it has received from the game into an agreed upon format. Once the information has been formatted it will then send the information to the server. Once the client receives information from the server it parse and format it.

#### Outputs

In the case that the client sent data to the server it will wait patiently for the server to send information about the opponent's next move. In the case that the client received the information it will send the formatted data to the game client.

### Create Network Game

#### Introduction

The user initiated a stand alone game.

#### Inputs

There are no inputs.

#### Processing

The game object and local AI are created to play against.

#### Outputs

A graphic user interface displaying the game appears.

## **Performance Requirements**

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

As of this version of the documentation there are no known performance requirements.

## **Software System Attributes**

The main idea here is to judge the ability of the AI to play the game optimally. We will use a Minimax algorithm and AB-pruning in order to judge what the optimal policy is and use testing to make sure that the AI always plans according to the optimal policy.

### **Availability**

If the game experiences disconnects or lag, it will result in a draw of the disconnected player.

### **Security**

There is no need for security, as there is very little commercial value to protect.

### **Maintainability**

The application will be easy to maintain in its design will allow for easy implementation of new functionalities. In addition, we will implement low coupling so that there will not be many ripple-effects whenever a component is changed or added.

## **Design constraints**

When designing we must use UML CASE tools for our modeling. We must also use Microsoft Project to track progress, as well as a version control system for our software.

### **Standards Compliance**

As of this version there is no standard of which to follow. In the future, all teams will decide on a standard for connectivity between each other.

### **Hardware Limitations**

We are limited to designing our software to run on a standard personal computer.

**Appendix A: Issues List**