# Software Requirements Specification

**Windmill Software**
Steven Cornella
Armon Entezari
Dustin Ingram
Aaron Rosenfeld
Michael Vadovszki

March 13, 2010
Revision 1

## Approval Sheet

| | |
|---|---|
| Steven Cornella | Date |
| Armon Entezari | Date |
| Dustin Ingram | Date |
| Aaron Rosenfeld | Date |
| Michael Vadovszki | Date |

## Revision History

| Date | Description | Revision |
|---|---|---|
| February 8, 2010 | Initial specifications | 0 |
| March 13, 2010 | Revised use-cases | 1 |

# Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to explain all necessary requirements for Windmill's chess software in the IEEE standard. This include functional, non-functional, constraints, use case specifications, and graphical prototypes of our chess software.

## 1.2 Scope

This document is the initial version (1.0) of the requirements specifications for Windmill's chess software. Software developers and testers are the intended readers of this document.

## 1.3 Definition, Acronyms, and Abbreviations

**Bishop** One of six chess pieces in a chess game. It is permitted to only move in a diagonal direction on a chessboard.

**Checkmate** A chess game state wherein one player's king will be inevitably captured.

**Chess** A board game played between two players on a chess board with sixty four squares. Each square may hold only one chess piece.

**GUI** Acronym for Graphical User Interface. It provides a graphical front end for computer programs.

**Internet** A large worldwide system of connected networks.

**IP Address** Unique 32-bit number that identifies any computer connected to the Internet.

**Port** A 16-bit number that indicates a communication channel on a specific machine.

**Java** An object oriented computer programming language.

**King** One of six chess pieces in a chess game. Each player receives one king at the start of the chess game. It can move in any direction on a chessboard. The objective in a chess game is to protect it from capture by the opponent.

**Knight** One of six chess pieces in a chess game. Each player receives two knights at the start of the chess game. It moves in two possible unique ways, two horizontally and one vertically or vice versa.

**Queen** One of six chess pieces in a chess game. Each player receives one queen at the start of the chess game. In a chess game, it can move in any direction.

**Rook** One of six chess pieces in a chess game. Each player receives two rooks at the start of the chess game. It can move either horizontally or vertically any number of squares.

**Swing** An API library for providing a GUI to Java programs.

**Thread** Code that runs within the address space of a single process.

### 1.4 Overview

Following this introduction, an overall description, use case diagrams, non-functional and functional requirements, and graphical user interface prototypes are presented.

## 2 Overall Description

### 2.1 Product Perspective

Chess has been one of the most widely known and competitive games since its creation centuries ago. While the rules of the game haven't changed, the people who play chess and the places chess is played have. To keep the game popular and convenient to play in this modern age, a digital successor to the original board game is required.

### 2.2 System Interfaces

Windmill's chess software focuses on simplicity of design and ease of play. Therefore there are no separate client/server interfaces; it utilizes one universal client program to automatically connect players.

### 2.3 User Interface

The Windmill client has a very simple interface leveraging Javas Swing to provide a simple and enjoyable game of chess across a standard network. The interface provides all the functionality needed to connect to another client and play a standard rules-compliant game of chess.

### 2.4 Hardware Interface

The chess client, because its running on Java and using standard libraries should run on any computer with an up-to-date version of Suns JVM, a mouse, keyboard, and network interface for playing against remote clients.

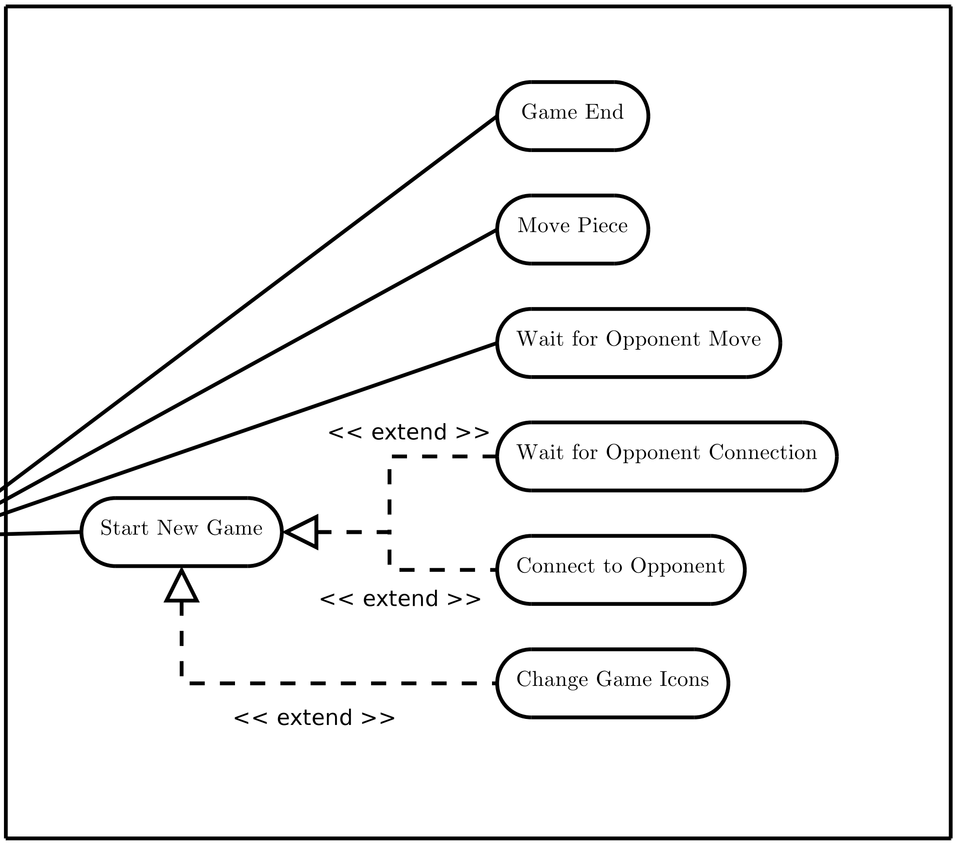
### 2.5 Product Functions

Windmills chess software will provide the interface, graphics, and logic required to play a standard game of chess over a network connection with another chess client.

### 2.6 User Characteristics

The only knowledge required is the basic knowledge of how to play chess. The simple and intuitive interface will allow users to jump right into a game without any difficulty.

## 3 Use Cases



### 3.1 Start New Game

**Preconditions**

None.

**Main Flow**

The use case begins when the system presents the player with a connection dialog containing two methods of play (S-1). The user may choose icons for the chess pieces (S-2). The system then attempts to establish a connection to the remote player (S-3, E-1).

**Subflows**

- **S-1 Method Selection** The player shall select either “Wait for an Opponent” or “Connect to an Opponent.” Waiting for an opponent moves the system to the “Wait for Opponent Connection” use-case. “Connect to Opponent” moves the system to the “Connect to Opponent” use-case.
- **S-2 Icon Selection** The player shall have the ability to select “Change Game Icons” before starting the game. Doing so will move the system to the “Change Game Icons” use-case.
- **S-3 Connection** The player shall select “Ok” in the connection dialog to either wait for an incoming player connection or attempt to establish a connection based on the selection of S-1.

**Alternative Flows**

- **E-1 Connection Failure** The program was unable to establish a connection. The system presents a dialog explaining the cause of the error shall be shown and return to the connection dialog.
- **E-2 Program Closed** The program has been closed and the program shall terminate.

## 3.2 Wait for Opponent Connection

**Preconditions**

The user has selected “Wait for an Opponent” in the “Begin Game” use-case and pressed “Ok.”

**Main Flow**

The system shall prevents further actions (S-1) until another user connects (S-2).

**Subflows**

- **S-1 Connection Dialog Lock** The user shall not be able to interact with the connection dialog until a connection is established.
- **S-2 User Connects** Another user connects to the system (as specified in the “Connect to Opponent” use-case.) The game board shall be shown and the connection dialog hidden and the system moves to the “Move Piece” use-case.

**Alternative Flows**

- **E-1 Program Closed** The program has been closed and the program shall terminate.

## 3.3 Connect to Opponent

**Preconditions**

The user has selected “Connect to an Opponent” in the “Begin Game” use-case and pressed “Ok.”

**Main Flow**

The system shall attempt to connect to the specified IP and port (S-1). The connection may succeed (S-2) or fail (E-1).

**Subflows**

- **S-1 Attempt Connection** The system shall attempt to establish a connection to the designated IP and port. If successful, the system moves to (S-2). If unsuccessful, the system moves to (E-1).
- **S-2 Connection Successful** The system shall display the board, hide the connection dialog and move to the “Wait for Opponent Move” use-case.

**Alternative Flows**

- **E-1 Connection Error** There was an error with the connection or the connection timed out while waiting for a response from the user. The game presents an error message and returns to the “Start New Game” use-case.

## 3.4 Move Piece

#### Preconditions

It is the local user's turn.

#### Main Flow

The user selects a piece and the system highlights the allowed moves on the board (S-1). The player then drags-and-drops a piece onto an allowed square (S-2). This information is communicated to the remote player (S-3). If the move is not allowed, the system goes to (E-1). In the case of network problems on either end, (E-2) is triggered.

#### Subflows

- **S-1 Highlight Moves** The user shall click a piece. The positions on the chess board that are allowed for the selected piece shall be highlighted by changing to a different color.
- **S-2 Allowed Move** The user shall drag the piece onto an allowed position. The icon for the associated piece shall be moved to that location by the system.
- **S-3 Communicate Move** The move shall be communicated to the remote system to update the remote chess board.

#### Alternative Flows

- **E-1 Invalid Move** The player has moved the piece to a disallowed position. The piece shall be moved back to the original location by the system.
- **E-2 Network Error** If either players' network connection fails, the game shall terminate.

## 3.5 Wait for Opponent Move

#### Preconditions

It is the remote player's turn.

#### Main Flow

The remote player is making a move (use-case "Move Piece" from their perspective). The local board is locked (S-1) and the player cannot move. Upon receiving a move from the opponent (S-2), the local system moves to the "Move Piece" use-case.

#### Subflows

- **S-1 Game Board Lock** The game board shall be locked and the user is prevented from interacting with the board or game pieces.
- **S-2 Receive Move** The remote user has moved (use-case "Move Piece" use-case from their perspective) and the local system shall move to the "Move Piece" use-case if a win is not found. If it is, the system shall move to "Game End."

#### Alternative Flows

- **E-1 Network Error** If either players' network connection fails, the game shall terminate.

### 3.6 Game End

**Preconditions**

The game has been won by either player (S-1),

**Main Flow**

Both players' boards shall be locked (S-1) and notified of who won (S-2). The game will then terminate.

**Subflows**

- **S-1 Game Board Lock** The game board shall be locked and the user prevented from interacting with the board or game pieces.
- **S-2 Win Notification** Both players shall be notified of which player won the game. An "Ok" button shall be displayed which terminates the game when clicked.

**Alternative Flows**

None.

### 3.7 Change Game Icons

**Preconditions**

The user has selected "Browse" in the "Custom Icon Directory" box on the connection dialog.

**Main Flow**

The user selects a directory on the local machine (S-1) and the dialog closes (S-2).

**Subflows**

- **S-1 Directory Selection** The user shall select a directory from their local machine containing icons for the chess pieces.
- **S-2 Dialog Close** The system shall close the dialog and load the icons. The user shall be returned to the connection dialog.

**Alternative Flows**

None.

## 4 Non-Functional Requirements

### 4.1 Quality

#### 4.1.1 Usability

The chess game shall be easy to understand and use. All user input shall occur instantly ($< 1$ sec.) on the local machine. All transmissions to a remote client shall be invoked instantly ($< 1$ sec.) and shall appear on the remote machine in an amount of time not much greater than ($< 1$ sec.) the latency between hosts.

#### 4.1.2 Stability

The chess game shall not crash with normal use. All errors shall alert the user or gracefully terminate the program.

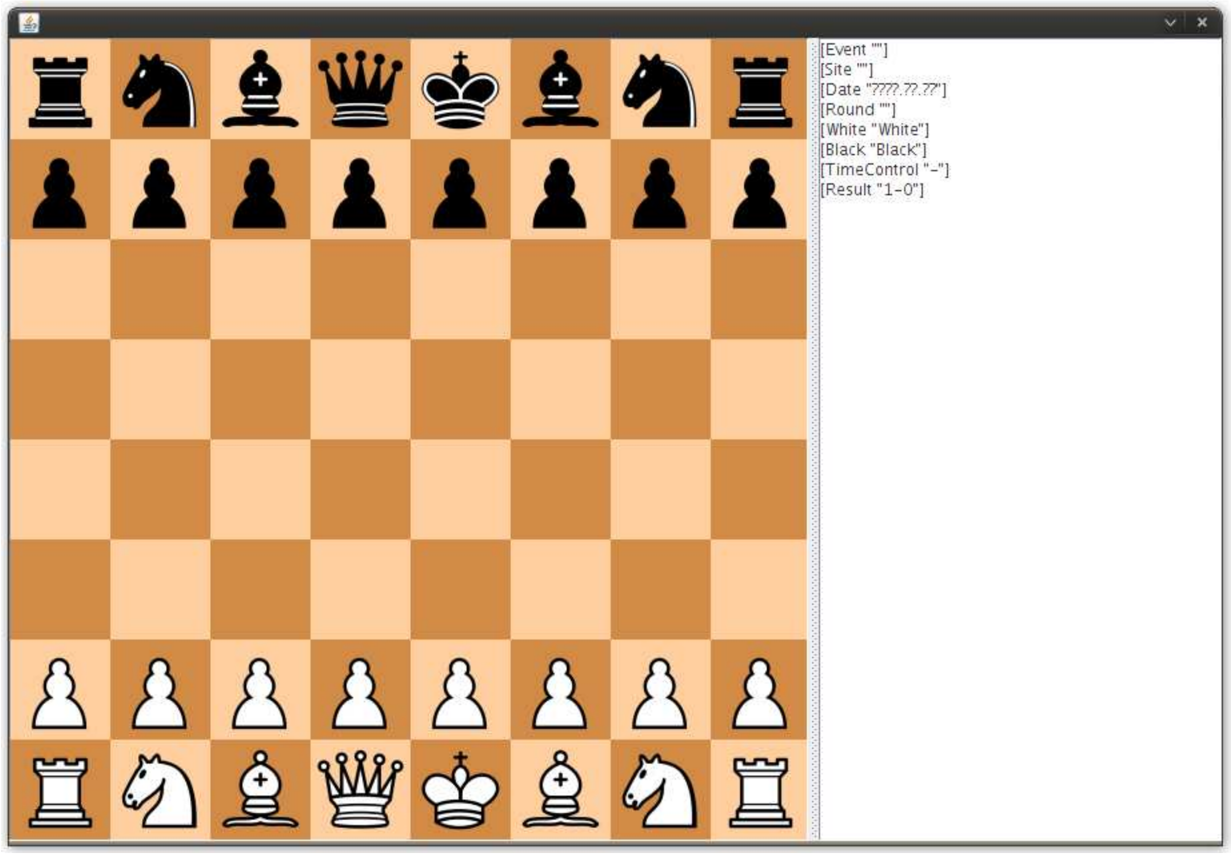
### 4.2 Performance

#### 4.2.1 Memory and CPU

The chess game shall not consume more than 128 Mb of memory nor utilize more than ≈ 1% of CPU cycles.

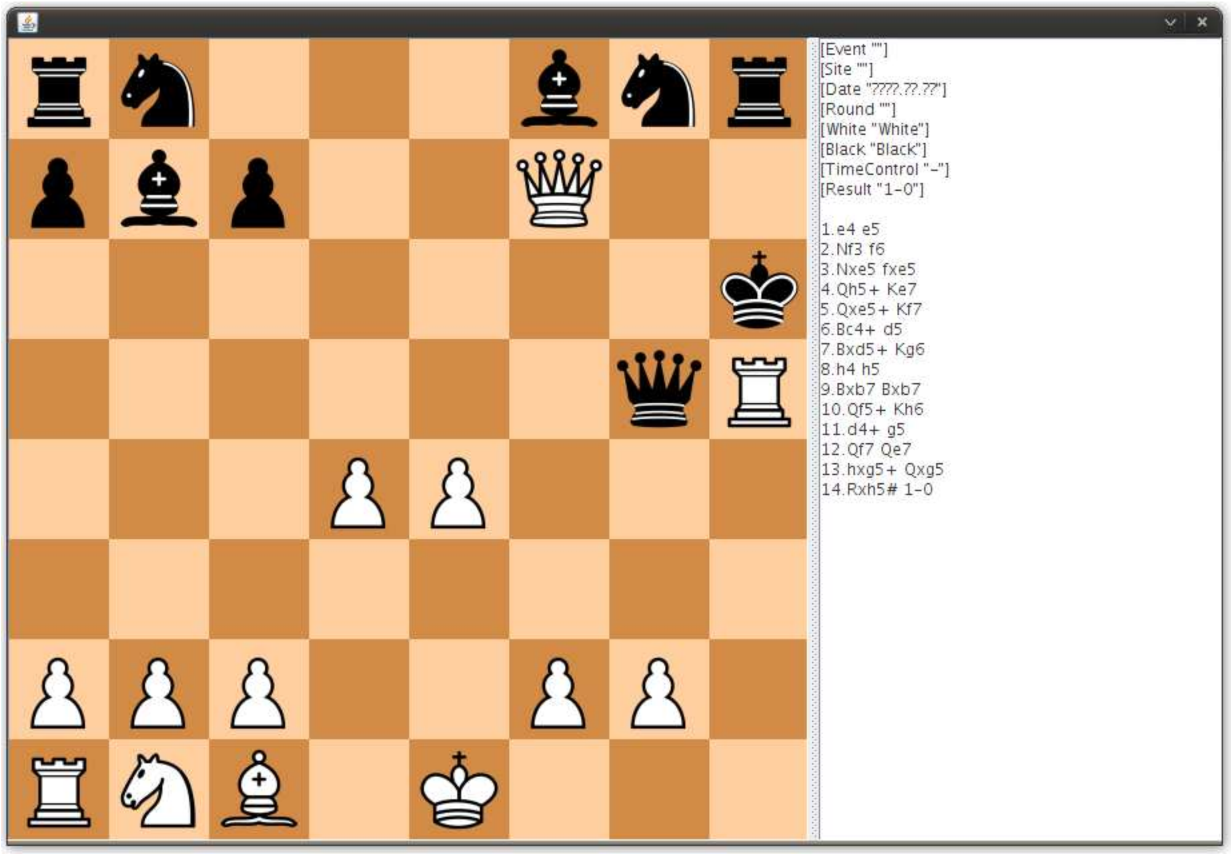
## 5 Graphical User Interfaces

### 5.1 Before Gameplay



```
[Event ""]
[Site ""]
[Date "????.??.??"]
[Round ""]
[White "White"]
[Black "Black"]
[TimeControl "-"]
[Result "1-0"]
```

## 5.2 During Gameplay



```
[Event ""]
[Site ""]
[Date "????.??.??"]
[Round ""]
[White "White"]
[Black "Black"]
[TimeControl "-"]
[Result "1-0"]

1.e4 e5
2.Nf3 f6
3.Nxe5 fxe5
4.Qh5+ Ke7
5.Qxe5+ Kf7
6.Bc4+ d5
7.Bxd5+ Kg6
8.h4 h5
9.Bxb7 Bxb7
10.Qf5+ Kh6
11.d4+ g5
12.Qf7 Qe7
13.hxg5+ Qxg5
14.Rxh5# 1-0
```