Software Requirements Specification

for

Chess Chaos

Version 1.0 approved

Prepared by Drew Grubb

Texas State University

Committed March 7, 2018

Table of Contents

1. Introduction

1.1 Purpose

This document specifies all the functional and nonfunctional requirements for Chess Chaos, a program that will be designed specifically for versatility to gain professional programming experience and habits.

This document covers the entire stand-alone system and will be used for the declaration of this software prior to development.

1.2 Document Conventions

Lists are generally made in a high priority to low priority manner unless otherwise specified.

The priorities for each requirement is stated in the Features section of this document (4). Although Chess Chaos follows the waterfall method and all features will be implemented with that in mind, the priorities are considerably arbitrary and purely for the sake of professional practice.

1.3 Intended Audience and Reading Suggestions

This document is intended for all students, developers, testers, documentation writers, project managers, employers, and anyone else who might find this information useful.

This entire project is built from the Requirements phase to the Testing phase by Drew Grubb alone, so the suggested reading orders below are somewhat arbitrary and based off my personal recommendation.
I suggest all audience members read the entirety of the Introduction section (1) first, as it gives a rough overview of any parts missed or jumped around by the reader.

Suggested Reading Order:
- Developers
    - Section 2: Grasping the product perspective, goals, and constraints.
    - Section 3: Project interfaces and design concerns.
    - Section 4: Project features and priorities.
- Testers and Students
    - Section 2: Grasping the product perspective, goals, and constraints.
    - Section 4: Project features and priorities for the sake of testing.
- Project Managers and Employers
    - Section 2: Grasping the product perspective, goals, and constraints.
    - Section 3: Project interface and design concerns.
    - Section 4: Project features and priorities.

There is an Appendix Glossary at the very end of this document for readability and consistency.

1.4 Product Scope

Chess Chaos is a stand-alone program that will display and run several variations of Chess based on the input of the user.

The objective of this project is to gain experience in software engineering and object-oriented programming using the waterfall design methodology via building a personal portfolio. This program will be uploaded completely Open Source to GitHub for anyone to view and download.

There are several programming goals for this project:

- Create a working chess framework with all the included pieces and move sets with the primary focus of versatility and efficiency.
- Create a Player vs. Player game of Chess
- Create a Player vs. AI game of Chess
- Create an AI. vs. AI game of Chess
- Create a variety of game options such as a professional Timed Mode, multiple AI algorithms, as well as several Demo games to show off the versatility of the backend programming.

1.5 References

GitHub Repository for this Project
https://github.com/drewgrubb0/Chess_Chaos

Contains:

- Software Requirements Specification (This document)
- Software Design Document
- Source code and associated project files
- Test Plan
- Any licensing information (MIT License)

Personal References:

My LinkedIn (Hire me)
https://www.linkedin.com/in/drew-grubb/

2. Overall Description

2.1 Product Perspective

Chess Chaos is a new, self-contained project that shall replicate the standard rules, gameplay, and visual experience of chess on a virtual machine.

Chess Chaos will provide a fresh experience through the addition of expanded game modes such as AI vs. AI or a Random Mode.

2.2 Product Functions

The product must:
- Provide a Graphical User Interface
- Provide a menu system that changes the game settings when the user navigates through it.
- Provide a functioning game of chess that follows Section 4 features and requirements.

The user must be able to:
- Interact with the Graphical User Interface with the Mouse buttons
- Easily understand and execute important functions
- Play a complete game of Chess without encountering compromising issues

2.3 User Classes and Characteristics

Chess Chaos operates under a single authorization of "User". This user can interact with the program using the mouse and keyboard and is expected to be able to navigate a menu and know the rules of chess to fully experience the qualities of this product.
For Player vs. Player versions of Chess Chaos, the single user is responsible for the manual alternating of turns.

2.4 Operating Environment

Chess Chaos is not platform dependent and can run on any system that has a Java Runtime Environment installed on the machine.
Chess Chaos will not have any issues running on JRE's built later than 1995 due to the AWT GUI library being the most recent library that will be used.

To install a JRE on your machine, visit
http://www.oracle.com/technetwork/java/javase/documentation/install-windows-64--153423.html

2.5 Design and Implementation Constraints

Chess Chaos may experience hardware limitations on extremely slow machines. Although this should not affect the playability of the game, these constraints could lead to slower algorithm execution.

2.6 User Documentation

For this project, I will be following the basic Javadoc documentation style to provide simple and readable online assistance via the Java Platform API. These Javadoc comments shall be present above every class and every method that requires explaining beyond its name. For more advanced algorithms, comments shall be presented within the source code to guide the reader through the process.

2.7 Assumptions and Dependencies

Chess Chaos is built on the assumption that the machine attempting to run this software has enough memory space to run it. Chess Chaos is installed under the assumption that the machine has a Java Runtime Environment (JRE), and will fail to load or run if a JRE is not present.

3. External Interface Requirements

3.1 User Interfaces

The primary interaction between Chess Chaos and the user come from mouse interaction with a set of various Graphical User Interfaces. For example, the options available to the user upon entering the game will differ significantly from the different stages of the menu.
Chess Chaos shall contain three main Graphical User Interfaces

1. Menu GUI:
This is where the user selects their settings or game type. The menu shall consist of up to four different buttons depending on the current state of the selections made by the user:

Initial Screen: [Play Chess | Play a Replay | Quickplay]
Board Screen: [Standard Chess | Random Chess | Etc]
Players Screen: [Player | Random AI | Hard AI | Etc]
Settings Screen: [Casual | Timed Mode]

2. Player vs. Player GUI:
Layout:
Chess Board: Left 65% of the screen
In Game Settings: Right 35% of the screen

Buttons:
Clicking on a piece "selects" it.
Clicking on a tile moves a selected piece to that tile (if allowed), and switches the turn
Forfeit button
Restart button
Pause button (in timed mode only)

3. AI vs. AI GUI:
Layout:
Chess Board: Left 65% of the screen
In Game Settings: Right 35% of the screen

Buttons:
Cancel/Quit Game
Pause/Resume Game
Reset Game
Undo Moves

3.2 Hardware Interfaces

Chess Chaos is supported on any machine that supports the Java Virtual Machine and has a Mouse Input mechanism and a display screen.
Chess Chaos is touchscreen compatible.

3.3 Software Interfaces

Chess Chaos shall be written using Eclipse Oxygen using JDK 1.8 on the windows platform. There will be no external libraries used in this program.

4. System Features

Priorities:
1 – High: System shall not run without this feature
2 – Medium: Usability will be difficult without this feature.
3 – Low: Feature primarily for convenience.
4 – Optional: Feature primarily for extra options.

4.1 Menu Navigation

4.1.1 Description and Priority

Chess Chaos shall change the setup of the Chess game depending on the navigation of the menu by the user. The UI for the menu can be found in Section 3.1

This feature will have priority 1

4.1.2 Stimulus/Response Sequences

Navigation will be determined by buttons clicked via the Mouse Listener system.

4.1.3 Functional Requirements

REQ-1: This program shall use the Mouse Listener to allow the user to click buttons
REQ-2: This program shall automatically switch around the states on button click
REQ-3: When the final setting is clicked, start the correct game with all the previously selected settings.

4.2 Basic Chess Functions

4.2.1 Description and Priority

Chess Chaos shall provide all the important features of basic chess as well as game mode specific game options.

This feature will have priority 1

4.2.2 Stimulus/Response Sequences

Chess activated by selecting all the necessary settings in the menu state.

4.2.3 Functional Requirements

REQ-1: White moves first, and the turn is alternated automatically.

REQ-2: When clicked, pieces hover possible moves
REQ-3: Possible moves are updated for each specific piece
REQ-4: When spaces are clicked, pieces are moved to that spot if allowed
REQ-5: The game will end when Checkmate is determined.
REQ-6: The game will limit possible moves when the King is in Check.
REQ-7: Special moves will need be accounted for: Castling, En Passant, and Promotion.

4.3 Artificial Intelligence

4.2.1 Description and Priority

Chess Chaos will contain different types of AI based off Random algorithms or Minimax Algorithms.

This feature will have priority 2.

4.2.2 Stimulus/Response Sequences

AI Players are only initiated if Player vs. AI or AI vs. AI is selected in the Player Selection screen.

4.2.3 Functional Requirements

REQ-1: Easy AI – Random Algorithms
REQ-2: Medium AI – Random/Minimax Algorithm

4.4 Different Game Modes

4.3.1 Description and Priority

Chess Chaos will contain different game modes and game types such as Timed Mode, Casual Mode, and Chess Chaos.

This feature will have priority 4.

4.3.2 Stimulus/Response Sequences

Different Game Modes will be initialized by the different settings in the menu.

4.3.3 Functional Requirements

REQ-1: Timed Mode will contain player specific timers that count down.
REQ-2: Timed Mode will force a player to forfeit upon hitting 0:00
REQ-3: Chess Chaos will feature 4 players

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Although there are no specific performance requirements for this project, the system shall not cause any noticeable lag on a standard computer.

Timers shall be run based off System Time, and as such shall perform consistently without a dependency on update speed.

5.2 Safety Requirements

Chess Chaos shall be built with safety in mind, and as such shall work to prevent possible harms that could result from eye strain, seizures, or frustration.

5.3 Security Requirements

Chess Chaos shall not handle or collect any personal data from the users or the System. This program has no security or privacy concerns.

5.4 Software Quality Attributes

The goal of Chess Chaos' quality characteristics shall be to provide a smooth user experience that shall not crash due to a user error or display failure. All text shall be Anti-Aliased by the API, and the game shall run at 30TPS/FPS to the best of the system's ability.

This software is Open Source under the MIT license, and will be available for download at any time after production and design.

Chess Chaos will be portable and playable on any major, standard Operating System.

5.5 Business Rules

Beyond the internal turn-based system that shall prevent the user to from "cheating" by affecting AI computation, there are no operating principles for this program that would prevent any users from performing similar actions under all circumstances.

Appendix A: Glossary

Listed by relevance with a secondary listing alphabetically

Chess Terms and Pieces:

Bishop: A piece that can move diagonally across any number of squares until it is stopped by the edge of the board or the presence of another piece. Can capture the first enemy piece it runs into.

Capture: The act of removing an enemy piece from the board by occupying its space with your own Piece.

Castle: A special move where, if the King has not been moved and either of the corresponding team's Rooks have not been moved, the King may move two squares towards a Rook and the Rook may move to the other side of the King.

Check: To make a move that places your opponents King in the **Line of Sight** of a piece, forcing them to move reactionarily.

Checkmate: To make a move that places the enemy King in Check where they cannot make a move to get themselves out of check, therefore ending the game in a victory.

En Passant: A special move where, if an enemy Pawn has just performed a double jump, a friendly pawn next to that pawn can perform a **Capture** in basic pawn style, sliding forward one space and horizontally one space.

King: A piece that can move one space in any direction or **Castle** if the state of the board allows it. The goal of the game is to force your opponent's King into **Checkmate**. Can capture an enemy piece provided doing so would not result in a **Check**.

Knight: A piece that makes its moves in an L shape (one space vertically & two spaces horizontally or visa versa). Can jump over pieces and directly capture pieces.

Line of Sight: If a move from a piece would knock out another piece, it is commonly said that the second piece is in the Line of Sight of the first piece.

Pawn: The lowest rank piece that can move forward two spaces on its first move and only one space every move after that. **Capture** is allowed only if the enemy piece is one space "in front of" and one space horizontally the pawn.

Promotion: The act of moving your pawn all the way to the opposing border throughout the course of the game, which allows you to switch out your pawn with a piece of any type, regardless of whether it is a duplicate.

Queen: The strongest piece in Chess, the Queen can move with the combined motion of the Rook and the Bishop.

Rook: A piece that can move any number of spaces horizontally or vertically until it is stopped by a border or piece. Can capture the first enemy it encounters in any direction.

Stalemate: An end game situation in which a player can make no legal moves but is not in check. Results in a draw.

<u>Interface and Integration</u>

AI: (Artificial Intelligence) A software developed replication of human intelligence, allowing the software to operate without direct, manual control.

API: (Application Programming Interface) An interface that provides development tools and communicates with software components with regards to building a specific program.

GUI: (Graphical User Interface) An interface that allows users to interact with software with textless, visual tools, such as windows, scroll bars, and buttons.

IDE: (Integrated Development Environment) A component of software development that allows programmers and users to analyze the software in question to looks for defects and attempt to debug the software.

Java: A programming language that relies heavily on object-oriented design and a general-purpose implementation of various software components.

JRE: (Java Runtime Environment) An application that allows users to develop and run coded software that relies on the Java programming language.

UML: (Unified Modeling Language) A language used to develop diagrams, serialize objects across multiple language databases, and visualize the design of software.

Appendix B: Analysis Models

All models are present in the Software Design Document, available on the public GitHub repository for this project (See 1.5: References).