SE 3XA3: Software Requirements Specification
ChessAce

Team 18, Team MIF
Jerry Ke, kex1
Harry Fu, fuh6
Morgan Cui, cuim2

October 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to re-implement the open source package ChessOOP. Large-scale video games are popular at present, but during the business days, most of the users, including students and employees, cannot enjoy such games with their insufficient fragmented time. Fortunately, people still have traditional board games like chess to utilize such spare time. Furthermore, chess can also help establish and enhance the relationship between different people. For example, two board game fans meet and choose to play chess during their study break. However, setting up the physical board and pieces usually depletes the limited time and the interest that appears occasionally. In this case, a portable, player versus player (PVP) chess program is needed to allow users to play it face to face anywhere anytime.

1.2 The Stakeholders

1.2.1 The Client

The client of the project is an external group who is acting as a evaluator and final viewer before its deployment. Such group of people may be interested in reverse-engineering or re-implement the project ChessAce in the later future.

1.2.2 The Customers

Consumers are individuals who will be interested in consuming the game content. The consumers, which members involve with all ages, who have the internet access to download the game, and the hardware support to interact with the game.

1.2.3 Other Stakeholders

Any other individuals, groups or organizations who have an interested in the game. Such entities mush have feasible tool to either interact with or investigate or modify the program following the software licensing.

- **Developer**: Members of MIF team are the developers for this project.

1.3 Mandated Constraints

1.3.1 Solution Constraints

- The program shall be implemented with Java language.
- The program shall be able to download to and execute on any Windows, OS X and Linux OS with Java JDK.

1.3.2 Implementation Environment of the Current System

- The program shall be able to use any integrated PC display adapter to display graphic component.

1.3.3 Partner or Collaborative Applications

ChessAce does not have any direct partner or collaborative applications but does rely on Eclipse and some image editor to be implemented. It also relys on a personal computer supporting Java in order to execute.

1.3.4 Off-the-Shelf Software

For the product to execute, the following off-the-shelf software is required:

- Java

Java is free-to-download from online resource.

1.3.5 Anticipated Workplace Environment

The antocopated workplace environment is anywhere except extreme condition such as storm, rain or any other conditions that are capable of damaging the cooperating hardware.

1.3.6 Schedule Constraints

ChessAce must be completed with functional software and appropriate documentation in one university semester by December 8.

1.3.7 Budget Constraints

The budget is not applicable to this project, since we are using our own laptop to complete the code development. No addition resource is needed.

1.4 Naming Conventions and Terminology

- Functional Requirements: Requirements that describes what the product will do
- GUI: Graphical User Interface
- ChessOOP: name of an open-source chess game program
- Non-functional Requirements: Requirements that describe qualities that the product will have

1.5 Relevant Facts and Assumptions

1.5.1 Relevant Facts

The application is intended for use by two users on their local machine. The length of the overall codes is approximately 1500 lines. The open-source project that the application is based upon has a very short revision history and part of documentation, and lacking a specific user manual.

1.5.2 Business Rules

The only rules we've established is that everyone should contribute approximately the same into the development, and members should try to take jobs if they feel they are stronger in certain areas.

1.5.3 Assumption

It will assume that the users will have general knowledge of computers. If the any advanced knowledge is needed, it will be explained in a user manual. The developer has assumed that all the developing tool, such as IDE and image processing, etc, will be free. Also, the team has assumed that all the users are able to gain internet access in order to download the project.

X The open-source project that the application is based upon has a very short revision history and part of documentation, even lacking a specific user manual. This needs to be addressed in reimplementation. It will assume that the users only have basic general knowledge of computers. In order to use the application, any elements or features of the application whose operation requires more advanced knowledge will be explained in a user manual.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

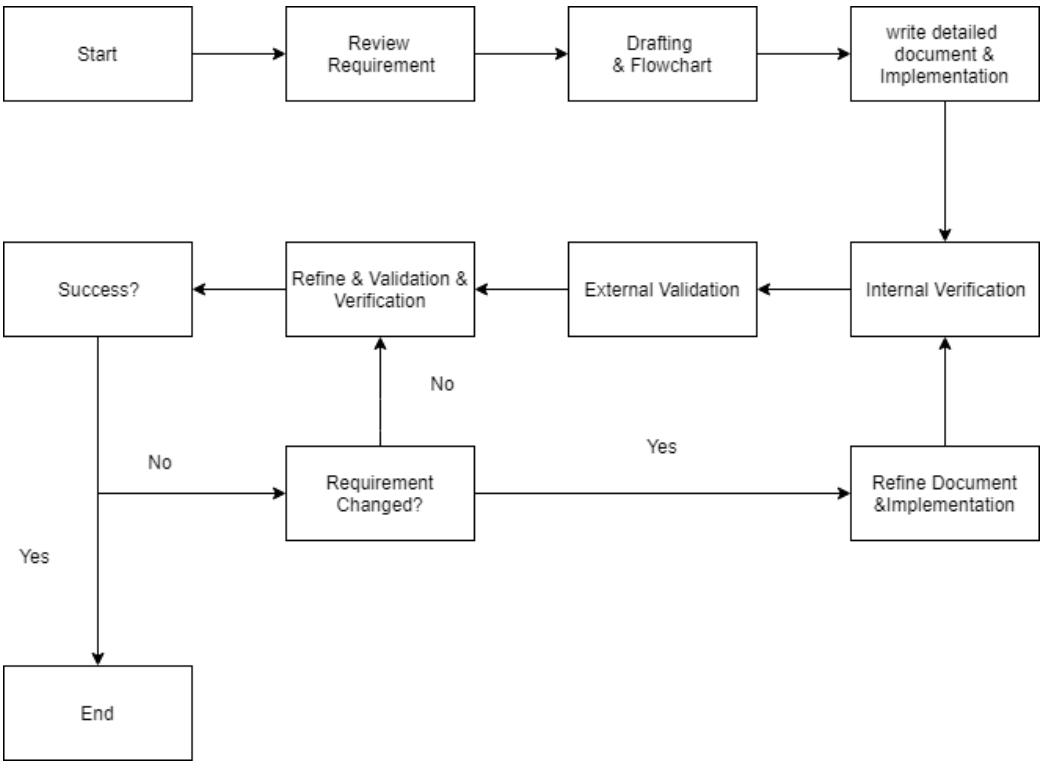

Figure 1: Flow chart for context of work.

2.1.2 Work Partitioning

Event Number	Event Name	Input	Output
1	ChessAce Creation	Developer code	Java GUI Window
2	ChessAce Movement Audio	Microphone, Audio file	Audio output device
3	ChessAce Pieces Creation	Drawing Tool	Java GUI Window
4	ChessAce Unit Elimination	Developer code	Java GUI Window
5	ChessAce Checkmate&Victory	Developer code and Image File	Java GUI Window

Table 2: Work Partitioning 1

2.1.3 Individual Product Use Cases

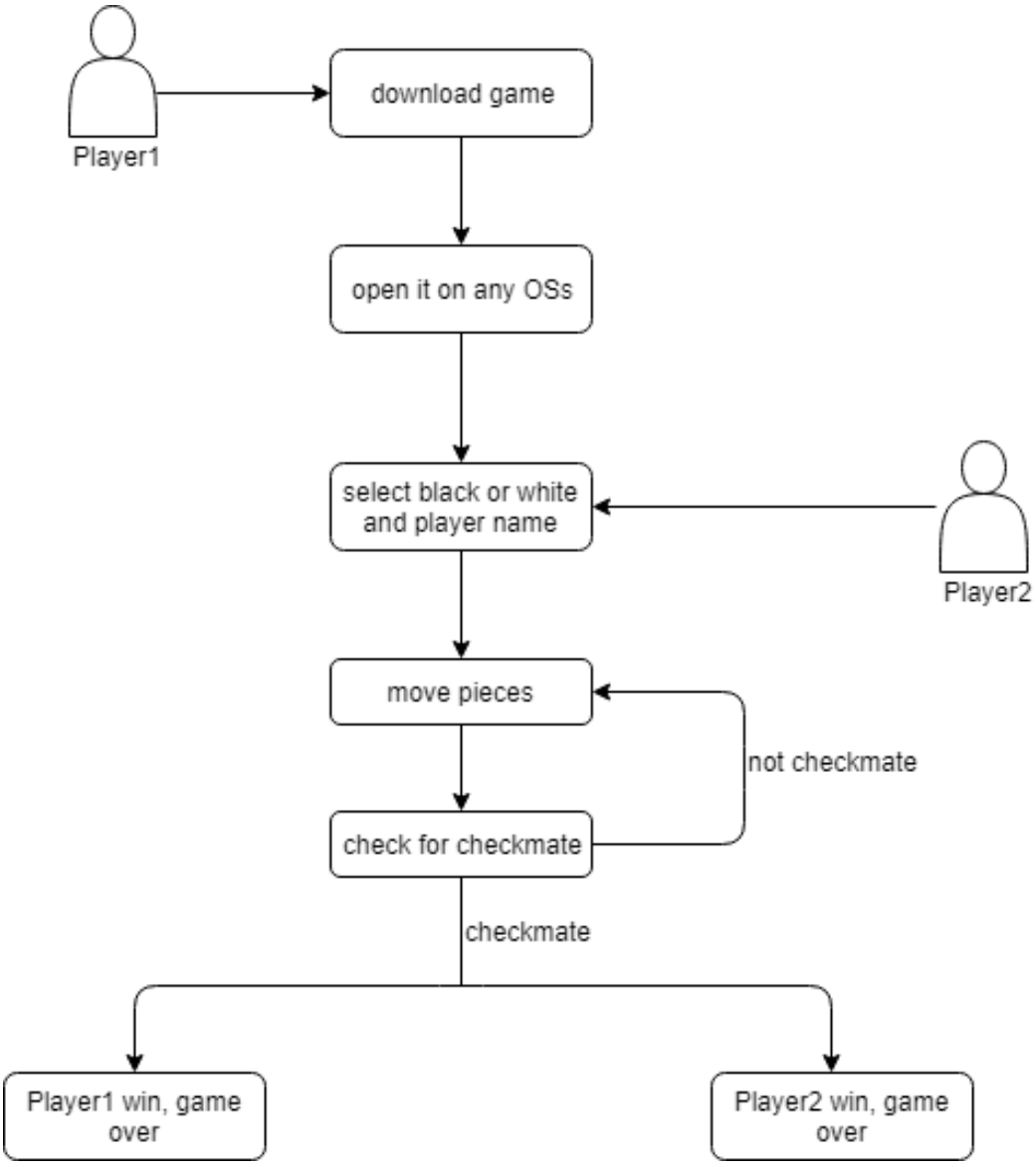

Figure 2: Individual Product Use Case.

2.2 Functional Requirements

- **Requirement number**: FR1
  The user must be able to move pieces on chess board.
  Rationale: If the user cannot play this game on a standard chess board, then the program is not performing its basic function.

- **Requirement number**: FR2
  The system must show possible movement for chess pieces to user.
  Rationale: To allow the user to choose the possible movement for pieces.

- **Requirement number**: FR3
  The system must perform unit elimination when user move one piece to the opponents piece.
  Rationale: To allow this game to perform the basic chess rules.

- **Requirement number**: FR4
  The user interface must show timer and current player.
  Rationale: To allow the user to manage their time and know whose term it is.

- **Requirement number**: FR5
  This game must let two players to play at the same time.
  Rationale: The purpose of this game is this game is a player versus player offline chess game.

- **Requirement number**: FR6
  When one of the players have no movement can be done, the system show checkmate.
  Rationale: To allow this game end and show the result.

- **Requirement number**: FR7
  The program must save the latest record of the game.
  Rationale: If the program cannot save the latest recording, then the program is not performing its required behaviour and is malfunctioning.

- **Requirement number**: FR8
  The player who is playing white is the first one to move.
  Rationale: If not the player who playing white to move first, then the program does not comply with the chess rule.

- **Requirement number**: FR9
  Users shall be able to download the executable jar and run it on any platform.
  Rationale: If this cannot be achieved, then the program is not performing its required behaviour.

- **Requirement number**: FR10
  Users shall be able to start a game once two users are available.
  Rationale: To meet the goal of this project which is a PVP game.

- **Requirement number**: FR11
  Users shall be given the choice of who plays black and white.
  Rationale: To improve the projects performance.

- **Requirement number**: FR12
  The inactive player may request to undo the prior move.
  Rationale: To allow this game to perform the basic chess rules.

- **Requirement number**: FR13
  There shall be no more than one undo request per turn.
  Rationale: To meet the general chess rule.

- **Requirement number**: FR14
  Captured pieces shall be displayed in a captured pieces box.
  Rationale: To improve the projects readability.

- **Requirement number**: FR15
  A player must be given a confirm dialog before forfeiting.
  Rationale: To avoid the user forfeiting the game by wrong click.

- **Requirement number**: FR16
  The active player must receive information about the remaining time to make the next move.
  Rationale: To remind the user left time and give a constraint to the program.

- **Requirement number**: FR17
  The active player shall select a piece by clicking it.
  Rationale: To formulate the rules of the program.

- **Requirement number**: FR18
  When a piece is selected, all legal moves for that piece are highlighted.
  Rationale: To improve the readability of the program.

- **Requirement number**: FR19
  When a piece is selected, the active player may select another piece by clicking it.
  Rationale: To formulate the rules of the program.

3 Non-functional Requirements

3.1 Look and Feel Requirements

- The chess board of the product must look like the standard 8*8 chess board.

- The game has to clearly label the pieces with white and black to distinguish sides.

- The pieces shall have distinguish shapes to show their identities.

3.2 Usability and Humanity Requirements

- The set up menu shall be easy for anyone over 6 years old to use.

- The game section shall be stratight forward for anyone who has a basic idea of chess play.

- For players who never played chess before, an integrated rule manual shall be provided to assist.

3.3 Performance Requirements

3.3.1 Speed Requirement

- The initialization of the game shall not take more than 10 seconds.

- All operation performed by player shall respond with a display change to indicate either a valid or an invalid operation under 0.5 seconds.

3.3.2 Safety Critical Requirement

- This execution process of this software is not related to any obvious physical hazards.

3.3.3 Precision Requirement

- The selected chess piece must move to the user selected valid square.
- The player can only move the pieces of the pre-selected side.

3.3.4 Reliability and Availability Requirement

- The software must be able to initialize and operate normally, not necessarily continuously, at any time.

3.4 Operational and Environmental Requirements

3.4.1 Expected physical environment

- The software will be used by people, who will be sitting down, inside a climate controlled environement.

3.4.2 Expected technical environment

- Players will interface with the software through either a desktop or laptop, powered by windows, OS X, Linux or other operating system that has JAVA JDK.

3.4.3 Partner Application

- ChessAce must cooperate with the hardware integrated with the PC, such as CPU and RAM.

3.5 Maintainability and Support Requirements

3.5.1 Maintainability

- Each piece, chess board and restriction should all have their unique module.
- Modules should keep the property of high cohesion low coupling.

3.5.2 Portability

- The program is expected to run on Linux, OS X, Windows and other operating systems with a Java JDK.

3.6 Security Requirements

- ChessAce shall not access unnecessary data on the user's device, and any accessed data, which is not generated by the program, shall not be able to upload, download to any cloud server or personal device.

3.7 Cultural Requirements

- ChessAce will not use any text, images, audio or other medias that will offend the ethnic group that purchase it.
- ChessAce shall show a disclaimer explaining that any similarity to any cultural, ideology or political figure is coincidental.

3.8 Legal Requirements

- ChessAce shall comply with all national and federal software regulation laws.
- ChessAce intelligent software shall comply with all relevant software standards.
- ChessAce intelligent software shall comply with all relevant privacy acts.

3.9 Health and Safety Requirements

- After 2 hours of continuous usage, prompt message will suggest a rest with a frequency of 15 minutes.
- After 3 hours of continuous usage, ChessAce will automatically terminate.

4 Project Issues

4.1 Open Issues

Automated Testing: The ability to determine the correctness of the player's chess movement needs to be implemented to enable automated testing. The ability to determine chess plate setting and checkmate (starting a game and ending game) needs to be implemented to enable automated testing.

Understanding code structure of existing open-source project: The project that the application is based on has undocumented code with some unimplemented basic features, so analyzing its code structure is necessary to understand how the program functions.

4.2 Off-the-Shelf Solutions

There are many currently available product that are similar to the application. We could find many existing online or offline chess video game online, such as chess with friends, chess titans, Sparkchess. These chess game are all similar to our project which could help us to understand the basic rules of chess and to implement our player versus player offline chess game.

4.3 New Problems

N/A ATM

4.4 Tasks

The project's task is set by the deliverable outline for Software Engineering 3XA3. The final demonstration and documentation will need to complete by December 5th, 2018. Deliverables include ProjectApproval, DevelopmentPlan, ProblemStatement, Requirements Document Revision 0, Proof of Concept Demonstration, Test Plan Reversion 0, Design and Document Revision 0, Revision 0 Demonstration, Final Demonstration Revision 1, Peer Evaluation of Oher Team Final Demo and Final Documentation Revision 1.

4.5 Migration to the New Product

Requirements will be phased in over time in order of the priorities listed in this documentation, starting with High Priority requirements. The new product will be tested on three devices, running Windows 10, Linux, and Mac OS 10. The product should pass the current battery of automated unit testing as well.

4.6 Risks

One of the risks is that the project may crash because the player may do some misoperation such as move the chess pieces out of chess plate or did some movement which breaks the chess's role. These operations may cause the software crash which also means that the testing portion of the project will heavily rely on the user's operation and chess movements. The other risks may be a memory leak and make the whole system slowing down. This means the testing portion of the project will heavily rely on checking the memory state to make sure no stack overflow happens.

4.7 Costs

N/A ATM

4.8 User Documentation and Training

The product will contain a user manual which will include instructions on how to use the application, the rules of chess movement and it will include details on all of its features and options.

4.9 Waiting Room

The GUI needs to be improved and the functional and non-functional requirements need to be implemented.

4.10 Ideas for Solutions

Using GUI development software could help improve the GUI.

5 Appendix

N/A ATM

5.1 Symbolic Parameters

N/A ATM