Software Requirement Specification for "The Jungle Game" Project

Author: HAN Jiaming, PEI Yuxing, LYU Yunfei, MA Suizhi

Version: V3.2, released on October 10, 2022

HKPU 22 Fall , Software Engineering

Project Website: https://hirsun.github.io/COMP3211_Wiki/

GitHub Repository: https://github.com/pyx-2021/Jungle

Contents

1 Preface

1.1 Design Objectives

This software requirement specification (SRS) accurately describes the requirements and essential features for developing the project "The Jungle Game". Referring this specification will facilitate developers to better understand the users' requirements, baseline information on the target software and the features of each component, as well as help in coding at and maintaining in the future. The specification is described in various formats including text, tables and flowcharts.

1.2 Release Iteration Description

This specification is the 3rd official public version (V3.2) of the project, released on October 10, 2022.

SRS shall be modified after requirements change. Therefore, this article might not be the latest. Readers may access the newest documents on the project's online wiki at https://hirsun.github.io/COMP3211_Wiki/task/srs.html. GitHub also keeps track of where changes have been made and the old version.

1.3 Expected Readers

The intended readers of this article are

- Product Managers
- Developers
- Customers of Deliverables
- Scholars of Software Engineering.

1.4 Abstract

In this Requirement Specification, we first make an overall introduction about the project. Then, in Glossary, we explain some critical vocabulary that is frequently used in this document. For the "User Requirements Definition" part, we summarize the main functionality needed in this project and introduce the critical rules of this game. Additionally, we involve a figure to show the System Architecture intuitively. Besides, we list some functional requirements to describe from the view of programmers. Some other requirements are also mentioned including product requirements and organizational requirements.

2 Introduction

2.1 Background

This project is a task whose main deliverable is a game, for teaching and evaluation purposes in HKPU's Software Engineering course for undergraduate in 2022 Fall. Members shall form groups on their own to complete the project in a standardized software development and project management procedure.

2.2 Overview

The main deliverable of this project is a game called "The Jungle Game". This game is a competition between 2 players on a board, which will result in a winner. Developers are required to design software according to game rules and deliver the product to customers.

3 Glossary

- Developers: All designers, programmers and testing engineers involved in the development of this project.
- The Jungle Game: A modern Chinese board game with an obscure history. More descriptions are available at Wikipedia[1].
- Github: An Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project [2].
- Git: A free and open-source software for distributed version control, tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development [3].
- Model-View-Controller (MVC): A software design pattern that divides software system components into three parts. Model is the data structures specified for the software system and the logic of the running system. View is to render and display components in Model in a proper format. Controller makes respond to inputs and modify the system state.[4].
- Maven: A popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. The tool provides allows developers to build and document the lifecycle framework [5].
- Gradle: A build automation tool known for its flexibility to build software. A build automation tool is used to automate the creation of applications [6].
- Docker: An open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications [7].

4 User Requirements Definition

4.1 General Functionality [8]

The major delivery expected by the customers should be a game.

- Its gameboard shall be a jungle terrain with dens, traps "set" around dens, and rivers.
- Each player should be able to controls eight game pieces representing different animals of various rank.
- Stronger-ranked animals should have the ability to capture ("eat") animals of weaker or equal rank.

The player who is first to maneuver any one of their pieces into the opponent's den should win the game. An alternative way to win shall be to capture all the opponent's pieces.

4.2 Critical Rules [1]

4.2.1 Board

A board of the Jungle game should consist of seven columns and nine rows of squares (Figure 1).

- Pieces shall move on the square spaces as in international chess, not on the lines as in Chinese Chess.
- Pictures of eight animals and their names should appear on each side of the board to indicate the initial placement of the game pieces.

After initial setup, these animal spaces have no special meaning in gameplay. There are several special squares and areas on the Jungle board:

1. The dens (Figure 2) shall be in the center of the boundary rows of the board and are labeled as such in Chinese.
2. Traps (Figure 3) shall be located to each side and in front of the dens and are also labeled in Chinese.
3. Two water areas or rivers (Figure 4) shall be in the center of the board: each comprises six squares in a 2X3 rectangle and is labeled with the Chinese characters for “river”.
4. Single columns of ordinary shall land squares on the edges of the board and down the middle between the rivers.

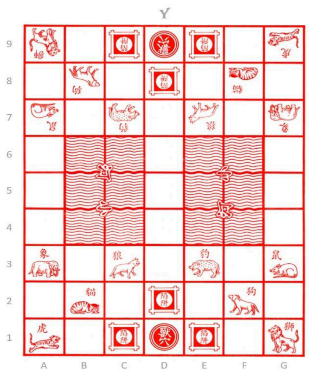

Figure 1: A typical Jungle gameboard showing the locations of starting squares, the den, rivers, and traps.

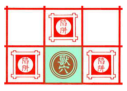
Figure 2: The den highlighted in green.

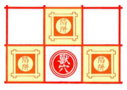
Figure 3: The traps highlighted in yellow.

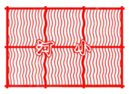
Figure 4: One of the rivers.

4.2.2 Pieces

Each side should have eight pieces represent different animals, each with a different rank.

- Higher ranking pieces should have the ability to capture all pieces of identical or lower ranking.
    - One exception: The rat should have the ability to capture the elephant, while the elephant shall not have the ability to capture the rat.
- The animal ranking, from highest to lowest, should be as shown in Table 1.
- Pieces should be placed onto the corresponding pictures of the animals which are invariably shown on the board.

Table 1: Pieces and their ranks.

Rank	Piece (en)	Piece (cn)
8	Elephant	象
7	Lion	獅
6	Tiger	虎
5	Leopard	豹
4	Wolf	狼
3	Dog	狗
2	Cat	貓
1	Rat	鼠

4.2.3 Movement

Players should alternate moves.

1. During their turn, a player should move.
2. Each piece should moves one square horizontally or vertically (not diagonally).
3. A piece shall not move to its own den.

There are special rules related to the water squares:

1. The rat should be the only type of animal that is allowed to go onto a water square.

2. The rat shall not capture the elephant or another rat on land directly from a water square.
3. A rat on land shall not attack a rat in the water.
4. The rat shall have the ability to attack the opponent rat if both pieces are in the water or on the land.
5. The lion and tiger pieces should jump over a river by moving horizontally or vertically.
    (a) They should move from a square on one edge of the river to the next non-water square on the other side.
    (b) Such a move shall not be allowed if there is a rat (whether friendly or enemy) on any of the intervening water squares.
    (c) The lion and tiger should be allowed to capture enemy pieces by such jumping moves.

4.2.4 Capturing

Animals capture the opponent pieces by "eating" them.

A piece should capture any enemy piece which has the same or lower rank, with the following exceptions:

1. A rat should be able to capture an elephant (but not from a water square);
2. A piece should have ability to capture any enemy piece in one of the player's trap squares regardless of rank.

4.2.5 Objective

The goal of the game should be to move a piece onto the den on the opponent's side of the board or capture all the opponent's pieces.

4.3 Player Characteristics

1. Should be suitable for players of various ages, who have some logical thinking skills.
2. Two players should alternately use the terminal to operate the game via the command line.

5 System Architecture

In the system developing phase, software engineers are supposed to develop the project using Model-View-Controller paradigm. This partitions the whole system into three sets of components. Model should define object data structure for chessboard, pieces, player, and their correspondent attributes. Controller should define the setting rules for the game, the process and mechanism of the program running flow, and appropriate controlling functions to make the whole system robust and maintainable. Controller should also interpret user inputs and check the validness of the commands. View should determine the display patterns of the game, and receive input from the user, and may have a well-accepted UX patterns. Prototype object class diagrams for Controller and View are shown in figure 6 and figure 7.

The project lifecycle should be managed by a proper tool. Alternatives are Maven and Gradle, which are used for automatically analyzing, building, and packaging. In the deployment phase, the software is supposed to be platform irrelevant and should not make at least assumption to the machine environment as possible. Docker is suggested.

To consider the system architecture in a client-server view, Controller should play the role of server, being active to manipulate data and execute core functions, and View should play the role of clients being the front-end to receive various form and inputs. The system should also contain a database layer of any form. Local file storage may be fine enough, but in-memory storage may not be robust to unexpected disturbs. In this architecture, the core functions like rule checking, chessboard updating, and game setup should be properly encapsulated. Users are not suggested to directly access these functions, but to indirectly get access to them by an intermediate layer consisting of managers.

This game might be divided into three critical parts, as shown in Figure5.

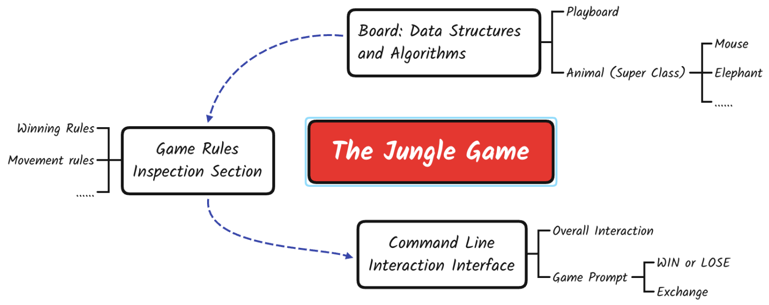

Figure 5: Critical parts of the Game.

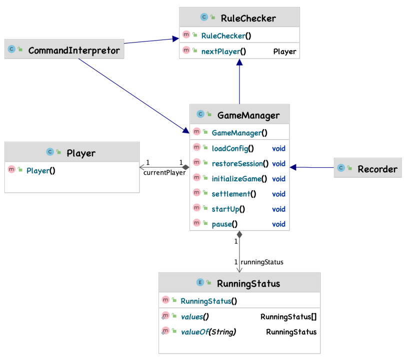

Figure 6: object class diagram of Controller.

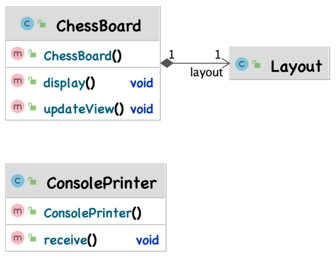

Figure 7: object class diagram of View.

6 System Requirements Specification

6.1 Functional Requirements

6.1.1 Board

1. The system shall display a command-line character-based board on the screen.

   A character-based board is necessary for users to know the game straightforwardly, and no need for an individual GUI.

2. The system shall display all surviving animals' positions in character form on the command-line board on the screen.

   It is necessary to let users know the current positions of their own/opponents' animals to make the next step decision.

3. The system shall display all special squares and their areas on the board.

   A special square has special attributes that distinguish it from other normal squares, which need to be informed to the users.

4. The system shall display all pieces with their names and sides in character at the correct position.

   Users need to know the position of all different animals on both sides because the movement and capture rule is different for different animals.

5. The system should display a title.

   It is helpful to let users know what program they are running to avoid mistakes.

6. The system should display the rule when requested by users.

   For new users, it will be useful for introducing them to the game. For old users, it will be not annoying because it will only display when requested.

6.1.2 Pieces

1. The system shall display all pieces with their names.

   Different pieces have different functions and ranks.

2. The system shall record the positions and ranks of all pieces.

   The rank is fixed for individual pieces, and it is necessary for capturing.

3. The system should display the special function of the selected piece.

   Sometimes users may forget the special function, and the system can remind them.

6.1.3 Movement

1. The system shall support taking turns when playing.

   Each user must move once and only once for each turn and after that pass to the opponent player.

2. The system shall allow users to select a valid piece to move

   The user can only move a piece when the piece is valid: it is his/her piece, and it has not been captured.

3. The system shall allow users to take a valid move.

   The piece can only move for one step, and it cannot be moved out of the board, to its den, to the river (if it is not a rat), across the river (if it is not a lion or tiger), and no own side piece.

4. The system shall display the updated board after each movement.

   The updated board is an essential reference for players.

5. The system shall report an error when an invalid selection of a piece is made.

It is necessary to let users know the selection of the animal is not allowed in the rule and the selection cannot be processed.

6. The system shall report an error when an invalid move is made.

   It is necessary to let users know the move of the piece is not allowed in the rule and the movement cannot be executed. For example, moving out of the board, moving to the water if not rat, moving across the river if not lion and tiger or there is a rat in the route, moving to a position already accommodated by another own side piece.

7. The system shall start capturing the process if a movement is attempting to move to a position with an opponent's piece.

   The opponent's piece needs to be captured by the comparing of rank.

8. The system shall report the winner of the game when a movement makes a piece go into the opponent's den.

   One winning objective of the game.

9. The system should provide explanations for the error of invalid selection.

   It is helpful to let users know why the piece cannot be selected. For example, selected opponent's piece, selected dead piece, or do not select any piece.

10. The system should provide explanations for the error of invalid movement.

    It is helpful to let users know why the movement is considered invalid. For example, moving out of the board, moving to the water if not rat, moving across the river if not lion and tiger, moving to a position already accommodated by another own side piece.

11. The system should provide warnings to the attempt of a movement that moves to a more powerful piece.

    If a piece is moved to a position that already has an opponent's more powerful piece, the moving piece will be captured immediately without any compensation.

12. The system should display the player side on the screen.

    It can help to let users know which side is playing.

6.1.4 Capturing

1. The system shall be able to determine if a capturing is happening.

   When a piece moves to a position that has already been occupied by the opponent's piece, we say a capturing is happening.

2. The system shall be able to determine whether the capturing is valid by comparison of ranks.

   Except the rat can capture the elephant, the animal with a higher rank can capture the animal with a lower rank.

3. The system shall be able to report an error when a rat is on the way to jumping from tigers and lions.

   According to the rules, the tiger cannot jump when a rat is in the river.

4. The system shall remove the captured animal after capturing process is completed.

   When a capture process is considered valid, the animal of the original position (i.e., the enemy animal) is eaten, and should be moved out of the game.

5. The system shall report an error when a rat is trying to capture elephants or another rat across water and land.

   According to game rule, rats are not allowed to capture rats or elephants directly from water to land or from land to water.

6. The system shall be able to process the capture process when the piece is in a trap.

   According to the game rule, a piece that is in a trap can be captured by any other piece.

7. The system shall announce the winner when all animals of one side have been captured.

   One of the winner's objectives.

8. The system shall update and display the board after capturing process.

   It is essential for users to know the updated positions of the pieces.

9. The system should be able to provide the reason why the capturing process cannot be finished.

   For new users who are not familiar with the rule, it can be a reminder.

10. The system should be able to send notifications of the capturing process.

    It can be a reminder for opponents that the piece has been captured and to change their strategies accordingly.

6.2 Non-functional Requirements

6.2.1 Product requirements

- The interface of the game should be easy to learn for an adult with basic computer experience.
- The latency of the system should be imperceptible (usually within 0.1s).
- The disk space of the game should be small enough for most home PC.
- The game should be able to run on an easy-setting Python environment.
- The game should run safely and not contain any virus or malware.

6.2.2 Organizational requirements

- The system should be environmentally and not take unnecessary computation to consume power.
- The system should be distributed digitally to avoid package waste.
- The system should be available for continuous development to avoid duplicated programming.

Users of the system should easily know which side he or she is in.

6.2.3 External requirements

- The system shall not save any unnecessary personal data.
- The system shall comply with local and international laws.
- The system shall not send user statistics for marketing purposes.
- The system shall provide basic cyber security protection in case of a leak of data.

6.3 API Interface MVC Structure

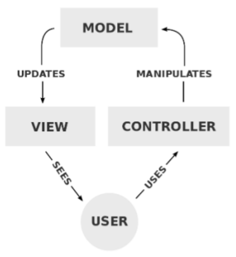

Figure 8: MVC Model

MVC Model is shown as Figure8.

- Model： Data structure of the game board Data structure of animals
- VIEW： Command-line terminal
- Controller: Backend Process

6.3.1 Updates

1. The basic structure of the board
2. Current/updated position of individual pieces
3. Notifications of events including errors, warnings, and notifications

6.3.2 Manipulates

1. User input of the select process
2. User input of the move process
3. User input of the capture process
4. User modifications of the commands after receiving notifications
5. User input of functional operations (such as exit, start a new game, etc.

Reference

1. https://en.wikipedia.org/wiki/Jungle_(board_game) ↩ ↩
2. https://en.wikipedia.org/wiki/GitHub ↩
3. https://en.wikipedia.org/wiki/Git ↩
4. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller ↩
5. https://en.wikipedia.org/wiki/Maven ↩
6. https://en.wikipedia.org/wiki/Gradle ↩
7. https://en.wikipedia.org/wiki/docker ↩
8. Course Project Description of HKPU's Software Engineering for UG, 22 Fall ↩

Indexes