MonoPolyU

Software Requirement Specification

The Monopoly Game

Authored by:

- Dominicus Dylan HARYOTO
- Haoran BAI
- Tao FU
- Weichen NING

Preface

This Software Requirements Specification (SRS) document is intended for the development team, project stakeholders, and the instructor overseeing the COMP3211 Software Engineering course at the Hong Kong Polytechnic University. The purpose of this document is to provide a comprehensive and detailed outline of the functional and non-functional requirements for the command-line-based Monopoly game that is to be developed as part of the course project.

In addition to defining the specifications for the software, this SRS will also address various aspects such as user roles, game mechanics, and system interactions. It will outline the anticipated user experience, detailing how players will interact with the game and what functionalities will be provided to enhance gameplay.

The SRS will be structured to cover all relevant areas, including

- Introduction: An overview of the project scope and objectives.
- Overall Description: A summary of the game environment, user characteristics, and assumptions.
- Functional Requirements: Detailed descriptions of the game's features and functionalities.
- Non-Functional Requirements:
- Performance metrics, usability criteria, and other quality attributes.
- System Architecture: An outline of the software architecture and technologies to be utilized.

This document is intended to evolve as the project progresses, incorporating feedback from team members and stakeholders to refine and clarify requirements. It will be reviewed periodically to ensure its relevance and accuracy, adapting to any changes in project scope or objectives.

Introduction

The purpose of this document is to specify the requirements for a command-line-based Monopoly game, referred to as "the game." Developed as part of the COMP3211 Software Engineering course at the Hong Kong Polytechnic University, the game aims to provide an engaging and interactive experience for 2 to 6 players, following the traditional rules of Monopoly.

This SRS outlines both the functional and non-functional requirements of the game, ensuring that all project stakeholders have a clear understanding of its intended features and design constraints. It will also be the main guideline for the whole project.

The game will allow players to engage in strategic turn-based gameplay involving property transactions, dice rolling, and financial management within a structured turn-based format. Key features include user commands for player movement, property management, and game state persistence through saving and loading functionalities.

Assumptions for this project include:

- All players have a basic understanding of Monopoly rules.
- The game will be developed using Java, ensuring compatibility with standard libraries for these languages.
- The user interface will be command-line based, emphasizing ease of use and clarity of interaction.

Aligning to the requirements outlined in this document, the development team aims to create a functional, reliable, and user-friendly software application that practices the software engineering theories, meets the project requirements, and provides an enjoyable gaming experience.

Glossary

1. Command-Line Interface (CLI): A text-based user interface used to view and manage computer files and run software programs by typing commands.
2. Develop: In software terms, refers to the creation and refinement of software applications.
3. Dice Roll: The act of simulating the throw of two four-sided dice, a fundamental game rule used to determine the number of positions a player moves forward on the gameboard.
4. Gameboard: The basics of Monopoly game, consisting of the path where the players stand on, and the squares containing different functions.
5. Gameboard Designer: A role or functionality within the game allowing customization and creation of new gameboard layouts as per user preferences or requirements.
6. Game Rules: The rules and methods designed for interaction between players and the gameboard, which includes the movement of players on the board, the decisions of players, and the effects of squares on the board.
7. HKD (Hong Kong Dollar): The official currency of Hong Kong, used as the unit of money in game transactions.
8. Interface: In computing, an interface can refer to the point of interaction between a computer and a user or between two software components.
9. Java: A high-level, class-based, object-oriented programming language.
10. Load Game: The process of retrieving and initiating game data from a previously saved state, allowing players to continue a game from where they left off.
11. Maintainability: A characteristic of software that defines the ease with which it can be modified to correct faults, improve performance, or adapt to a changing environment.
12. MVC (Model-View-Controller): An architectural pattern used in software engineering to separate the application logic into three interconnected components (model, view, controller), each handling specific aspects of the application.

13. Monopoly Game: A multiplayer economics-themed board game where players roll dice to move around the gameboard, buying properties and walking into squares. Players collect rent from their opponents, with the goal of driving them bankrupt.
14. Player: A participant in the Monopoly game, characterized by the index, position on the gameboard, owned properties, and total money.
15. Properties: Locations on the gameboard that can be bought and owned by the players. Each property has an associated cost, rent, and price.
16. Saving and Loading Game State: Features that allow the game's current state to be saved to a file for later retrieval and continuation of the game.
17. SRS (Software Requirements Specification): A document that captures a complete and clear description of the behavior of the system to be developed.
18. Template: A pre-designed layout or structure, indicating the predefined gameboard layout in this document.
19. Turn-based Gameplay: A style of gameplay mechanic where players take turns making their moves or decisions.
20. User Commands: Instructions inputted by the player through the command line to interact with the game, such as rolling dice or buying properties.
21. Verification: The process of checking data, documents, or software for correctness, completeness, and compliance with standards and specifications.

User Requirements Definition

Functional Requirements

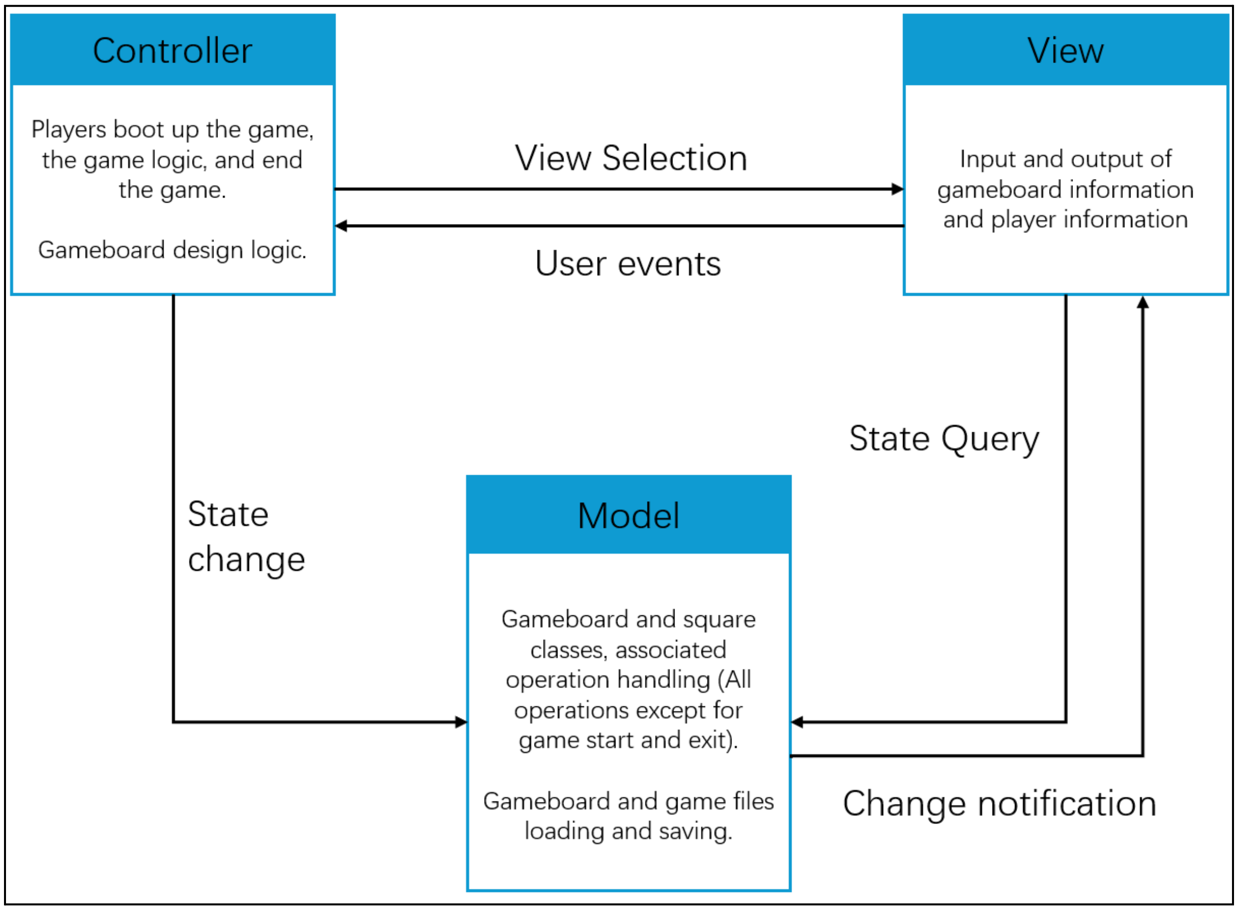
1. Game Initialization
    a. The game shall allow players to start a game with an existing game board.
    b. The game shall allow players to specify the number of players and the name of the players.
    c. The game shall be able to generate random names for the players.
2. Player Actions
    a. The game shall allow players to roll two four-sided dice and move forward the number of positions as the sum of points of these two dice.
    b. The game shall display the status of each player and each square for every player.
    c. The game shall allow players to query the next player.
3. Game Rules
    a. The game shall initialize the players with HKD 1500 and no property.
    b. The game shall let the players start from the first square.
    c. The game shall allow the players to advance squares clockwise and return to square 1 after reaching square 20.
    d. The game shall enable players to purchase unowned properties when they land on them by deducting the property's price from their current money.
    e. The game shall deduct rent from players who land on properties owned by others, based on the rent amounts specified for each property.
    f. When a player lands on the Income Tax square, the game shall deduct 10% of the player's current money (rounded down to the nearest 10).
    g. When a player lands on a chance square, the game shall take an effect on the player's money that ranges from maximum gain of HKD200 to a maximum loss of HKD300 (money in multiples of 10).
    h. When a player lands on a Chance square, the game will randomly select a number in multiples of 10 from a range of -300 to 200 (both inclusive), representing both losses and gains. The maximum gain is HKD200, while the maximum loss is HKD300.
    i. Every time a player passes the Go square, including landing exactly on it, the game shall add HKD1500 to that player's money as a salary.
    j. Landing on the Free Parking square shall have no effect on the player's financial status or position in the game.
    k. When a player lands on the Go to Jail square, the game shall move the player directly to the Jail square.
    l. If a player lands on the Jail/Just Visiting square without being sent to Jail, no action shall be taken.

    m. If a player is in Jail, they must either pay a fine of HKD 150 or roll doubles to get out. If they fail to roll doubles within three turns, they must pay the fine and then move the number of spaces shown on their third roll.
    n. The player shall retire when his/her money becomes negative, and his/her properties will become unknown.
    o. The game shall automatically end if only one player remains or after 100 rounds of play.
    p. The game shall determine the winner based on either the only one remaining, or the one with the most amount of money at the end of (100 rounds) game. Multiple winners are possible when multiple players have the same amount of money.
4. Saving and Loading Game State
    a. The game shall allow players to save the current game state to a file.
    b. The game shall allow players to load a game from a file and continue the file.
5. User Commands
    a. The game shall provide a list of valid user commands, and their formats as prompted.
    b. The game shall handle invalid user inputs gracefully by displaying an appropriate error message and prompting the user on how to provide a valid command.
6. Gameboard Designer
    a. The game shall allow gameboard designers to create a new gameboard from the given template.
    b. The game shall allow gameboard designers to modify properties of an existing gameboard. The gameboard designer could specify to update the name, price, or rent of a property.
    c. The game shall allow the gameboard designers to save the modified gameboards.
    d. The game shall allow the gameboard designers to review the properties of the accessed gameboard.

Non-functional requirements

1. Usability
    a. The command-line interface of the game should be intuitive, with clear instructions and navigation on how to play and interact with the game such that a player or designer could use no more than 5 minutes to be able to understand and execute all operations with no more than 5 mistakes.
2. Reliability
    a. The game should automatically handle and recover from unexpected errors 95% of the time, providing a clear error message and guidance for recovery.

System Architecture


Controller
Players boot up the game, the game logic, and end the game.
Gameboard design logic.
View Selection
User events
View
Input and output of gameboard information and player information
State Query
State change
Model
Gameboard and square classes, associated operation handling (All operations except for game start and exit).
Gameboard and game files loading and saving.
Change notification

Graph 1: MVC Architecture of the Command-Line-Based Monopoly Game

As shown in Graph 1 of the architecture designed for the project, it has been decided to adopt an MVC architecture pattern for the task to allow multiple ways of viewing and interacting with data. It enables the data to be modified independently of how it is represented, and vice versa. Additionally, it allows the same data to be displayed in various formats, ensuring that any changes made to one representation are reflected across all of them.

Controller: The interactions between the players, designers, and the system are managed by this part. It facilitates the sharing of user data with the View and the Model after booting up the system and processing updates.

View: The player and gameboard data will be presented through this component, either upon request or during critical situations.

Model: The player data and gameboard data will primarily be managed within this component. It will handle the operations requested by users and update the data or files accordingly.

System Requirements Specifications

Functional Requirements

1. Game Initialization
    a. The game shall process the user input to load the corresponding gameboard.
    b. The game shall prompt the user to enter the number of players and the name of the players, which are used to instantiate the players.
    c. The game shall include a function to generate random names for players who choose not to enter a name.
    d. The game shall process the user input to load the corresponding saved game file and continue the game.
2. Player Actions
    a. The game shall include a function allowing players to simulate the action of throwing two four-sided dice and use the output to modify player position data.
    b. The game shall include a function that is able to output the data of each player and each square for every player at any round.
3. Game Rules
    a. The game shall automatically initialize the money as HKD 1500 and property as an empty list for all players.
    b. The game shall initialize the position value as 1 (the first square) for all players.
    c. The game shall map the players' position to correct indices of squares, following the clockwise movement and returning to square 1 after reaching square 20.
    d. The game shall include the function allowing the player to purchase unowned properties and automatically decreasing the money according to the property's price.
    e. The game shall include the function to automatically decrease the property's rent from the money of a player A who landed on a property owned by another player B, while increasing the rent to the money of player B.
    f. The game shall include the function to automatically deduct the player money by 10% (rounded down to the nearest 10) when landing on the Income Tax square.
    g. The game shall include the function to randomly select a number in multiples of 10 from a range of -300 to 200 (both inclusive) when the player landing on a Chance square, representing both losses and gains. The maximum gain is HKD200, while the maximum loss is HKD300.
    h. The game shall include a function to automatically add HKD1500 to player money when the player passed or landed on the Go square.
    i. The game shall have no effect on the player's state (money, properties, and position) when landing on the Free Parking square.

    j. The game shall include the function to automatically change the player’s position to the Jail square and change the player status to “In Jail” when landing on the Go to Jail square.
    k. The game shall have no effect on the player’s state (money, properties, and position) when landing on the Jail square and without “In Jail” status.
    l. The game shall include the function to let the player choose to decrease the player’s money by HKD150 or roll doubles to get rid of the “In Jail” status. The player shall have three attempts to roll doubles; if they fail to do so within those attempts, their money will be automatically decreased by HKD150, and they will advance steps equivalent to their third dice throw from "In Jail" status.
    m. The game shall include a function to monitor the player's money. Once the data becomes negative, the player will be automatically retired from the game and the occupied properties will be released.
    n. The game shall include a function to determine the end of the game, either when only one player remains or when the number of rounds reaches 100.
    o. The game shall include a function to determine the winner(s), either by identifying the last remaining player, or by comparing the total amount of money of players at the end of the game. The winner is the one with the most amount of money in the latter case. If multiple players own the same amount of money, they are all listed as winners.
4. Saving and Loading Game State
    a. The game shall include the function to save all players’ data and the gameboard of the current game into a file.
    b. The game shall include the function to load the previously saved game data and continue the game.
5. User Commands
    a. When there are finite available options to be chosen by players, the game shall print out all available user options on the command line for users to check and choose.
    b. The game shall include exception handlers for each possible exception and print out the exception message, including what should be done next to handle it.
6. Gameboard Designer
    a. The game shall include the function for the gameboard designers to create a customized gameboard by editing the given property’s data, including name, price and rent.
    b. The game shall include the function for the gameboard designer to review the updated list of properties (if any) of the gameboard.
    c. The game shall develop the function for the gameboard designers to save the gameboards they modified.

Non-Functional Requirements

1. Maintainability
    a. The development of classes and methods should follow Model-View-Controller (MVC) architecture in separate packages, respectively.

2. Verification
    a. The unit tests should be run regressively before pushing to the master branch. The code line coverage shall be at least 95% for the Model package and 90% for the View and Controller packages.