

Jaybirds

Software Requirements Specification

Foundations of Software Engineering

Fall 2023

Akinwale | Brothers | Liu | Olson



Photo Credit: [The Johns Hopkins Hub](#)

1. Introduction

This document identifies the required components of a computer version of the popular board game Clue®.

2. Glossary

2.1 Game Terms

User: A single game player. See also the player.

Player: A single program user. See also the user.

Board Game: A contest played on a marked board, usually by moving tokens across it to achieve an objective.

Character: An identity assumed by each player that corresponds to a game token and game card.

Weapon: One of six items used (in theory only) to commit the crime that also corresponds to a game card.

Room: A place indicated on a game board depicting a large manor house that corresponds to a possible place that a (fictitious) crime was committed.

Card: A small, marked rectangle indicating various symbols in the game such as a character, weapon, or room.

Notebook: A private space on the screen that players use to record clues.

Hand: The group of cards assigned to a single player.

Deck: All of the cards in the game.

Suggestion: A player's claim that a single character, weapon, and room were part of the crime. The game's requirement that other players 'disprove' the suggestion generates clues that eventually enable a player to deduce the true character, weapon, and room of the crime.

Accusation: A player's final claim that a single character, weapon, and room were part of a crime. A player wins the game with an accurate accusation.

2.2 Computing Terms

Architecture: The layout of the software and hardware that will comprise the program.

Computer: A device typically operated by a single user to store and process information.

Server: A processor used to store data and provide it to other computers typically called clients.

Messaging Interface: A network, display, and protocol for passing messages between and among game players.

Python: A general purpose computer programming language.

3. System Architecture

The architecture includes separate computers for each player that interact with a server that controls the game state in a classic server-client model and with a python program. The server messages the players about the game state and clues, sometimes communicating with a specific player and sometimes with all players. The server also maintains individual player notebooks and displays the notebooks to the appropriate player only.

Subsystems

1. Player Computers:

Information: This subsystem includes the hardware and software on each player's computer.

Functions:

- Accept user input, such as moves and actions in the game.
- Display the game interface and relevant information to the player.
- Communicate with the server to send player actions and receive game state updates.

2. Server:

Information: This subsystem includes the server hardware and the server-side software.

Functions:

- Manage and control the game state, including player positions, scores, and clues.
- Handle incoming requests and messages from player computers.
- Distribute game state updates and clues to the appropriate players.
- Maintain individual player notebooks.

3. Python Program:

Information: This subsystem consists of the Python program used in the game.

Functions:

- Implement the game's logic and rules.
- Process player actions and update the game state accordingly.
- Generate and manage clues and game events.
- Communicate with the server to send and receive game-related data.

4. Player Notebooks:

Information: Each player has their own notebook containing game-related information, such as collected clues and personal notes.

Functions:

- Store and manage player-specific data.
- Display the notebook content to the respective player only.
- Update the notebook with new information as the game progresses.

5. Communication System:

Information: This subsystem handles communication between players and the server.

Functions:

- Establish and maintain network connections between player computers and the server.
- Transmit player actions, game state updates, and messages between players and the server.
- Ensure secure and reliable data transfer.

6. Game Interface:

Information: This is the user interface (UI) presented to players on their computers.

Functions:

- Display the game board, player avatars, and other game-related graphics.
- Provide controls and buttons for players to interact with the game.
- Show game state information, clues, and notifications.

7. Game Logic:

Information: This subsystem encapsulates the rules and game mechanics.

Functions:

- Enforce game rules and mechanics, such as player movement, scoring, and win conditions.
Generate random events or clues as needed.
- Manage the game's overall progression.

8. Messaging System:

Information: This subsystem manages the messages sent between the server and players.

Functions:

- Route messages from the server to the appropriate player or all players as needed.
- Handle message queuing and delivery.
- Ensure that players receive game updates and clues in a timely manner.

4. Functional Requirements

Player turn-taking and movement during the game is governed by a set of rules documented in the appendix. The software will support the following use cases:

User Action	Program Action & Functions
Use Case 1: Set Up a Game.	
	Depict setup view.
Declare intent to play	Confirm intent to play.

Declare # of players.	Request # of desired players. *
	Set “winning” condition—a character, weapon, and room.
	Depict all player pieces in their respective starting hallways.
Obtain a hand.	Deal remaining cards to players.
Review cards.	Display player hands.
	Branch to Take a Turn view.
*Rule #1: Player count must be $2 \leq x \leq 4$	
Use Case 2: Take Notes.	
Choose Notebook View.	Display notebook.
Mark clues.	Indicate Clues.
Close notebook view.	Branch to Player Turn.
Use Case 3: Take a Turn.	
	Depict Take a Turn view.
Indicate Notebook View if desired.	Branch to Notebook View.*
Indicate Make Accusation if desired.	Branch to Accusation View.**
Choose a location for move.	Display possible locations.
Select desired location.	Depict player at chosen location.
Indicate intent to make or decline making a suggestion.	Prompt for Suggestion.

*Rule #2: Ideally the player should be able to branch to his/her notebook at any point in the game.

**Rule #3: The player should be able to make an accusation at any point during his/her turn.

Use Case 4: Make a Suggestion.

	Depict Make a Suggestion View.
Indicate weapon and person.	Depict player suggestion.*
	Relocate person in suggestion to suggested room.**
	Branch to Respond to a Suggestion.

*Rule #4: The room choice = the current room of the player making the suggestion.

**Rule #5: The person indicated in the suggestion relocates to the room in the suggestion.

Use Case 5: Respond to a Suggestion.

	Depict Respond to a Suggestion View.
Choose to display a weapon, person, room or “none.”*	Depicts Response.
	Branch to Take a Turn.

* Rule #6: A response of “none” should be communicated to all players.

Use Case 6: Make an Accusation.

	Depict Make An Accusation View.
Choose weapon, person, room.	Check for winning condition.
	Branch to End Game or Wrong Choice.

Use Case 7: End A Game.

B. Player movement specifications:

Each player can move one space from a room to a hallway, through a secret passage, or from a hallway into a room. Each hallway can only hold one character. Rooms can hold multiple characters. If a hallway path is blocked to a player, s/he must choose another path. If every path is blocked to a player, s/he must remain in place.

The program moves the character named in a suggestion automatically and immediately to the room in the suggestion.

C. Suggestion and Accusation specifications:

A player may make a suggestion only if s/he has changed rooms or if the player was brought to a room due to a suggestion by another player. A player may enter and exit and re-enter a room as many times as desired. A player can make an accusation at any time during game play.

D. Non-functional Requirement:

The software should include the following quality attributes¹:

Reliable and Scalable: The software will function with 99% uptime—no more than 7 hours of downtime per month for updates and fixes. The software will support 2-3 players interacting in a single game and eventually grow to support 100 simultaneous games with separate players at the same time.

Flexible and Maintainable: The software will rely on continuous integration to support updates and bug fixes in a timely, non-disruptive manner.

Secure: The software will protect user account information with the highest industry standards. The program will shield the status of a game, user messaging, and the identities of players within a single game from outsiders to the game.

E. Implementation Constraints

Schedule & Pace: The software will demonstrate increasing functionality at three key increments to be delivered over a two month-time frame.

Cost: The software development will utilize free or nearly free tools for development, testing, integration and deployment due to the project team's limited student budget.

¹ Roger Pressman and Bruce Maxim, Software Engineering: A Practitioner's Approach, 9th Edition, (2020) 109, 201