CSCI-GA.2440: Software Engineering
Final Project Report

P2P (Peer-to-peer) Command Line Game Platform
Featuring Rock–Paper–Scissors

Shuidie Yao, Hui Duan
sy2328@nyu.edu, hd1021@nyu.edu

1. INTRODUCTION

In this project, we use Rational Unified Process (RUP) to develop a command line game platform with a P2P network as infrastructure. Besides a central database that stores user's login credentials and history scores, all other mechanics are peer to peer, including finding a co-player, sending message to each other, and handling game results.

For the first lifecycle, we implement the first game of the platform——Rock–Paper–Scissors. Users need a copy of our program and a terminal to access the platform. A player entering the game will find another player through the network, and the two will talk to each other once every round so that they can apply strategy to confuse the opponent, before showing their hand.

2. INCEPTION PHASE

In the inception phase, we played the role of stakeholders in a hypothetical computer game startup that want to find out the feasibility and profitability of an old-fashioned command line game that uses P2P technology. We focused on identifying the scope, cost and schedule of the project and working out its basic requirements and associated risks.

2.1. Basic Aspects

2.1.1. Scope

We decided to target players aged 40 and upwards for our command-line based game. Although people have become used to playing colorful, modern games, older players always remember the good old time of black-and-white text games. An old-fashioned command-line game platform has actually become a rarity now so a new one would certainly be appealing to the elder players.

We chose Rock–Paper–Scissors to be the first game to implement for our platform. The game has a long history, originating from as far back as the Ming Dynasty in China. Nowadays it has become well-known world-wide. It's a game proved both by time and scope, because it's simple yet challenging to win. An online version of the game can bring fresh experience to the broad group of potential audiences of elder players and Rock–Paper–Scissor fans. Without a need to learn the rules, players can join our game and have fun in just a few minutes. As a result we take advantage of the broadest game market of casual game players.

Finally, we chose to deploy the game on a P2P network because it has the following advantages:

- There is no central entity to control the game state, which means there is no need for a dedicated server to connect the client in order to constantly receives data, thus saving on operation costs.
- Message between users will be sent and received in a well-designed rule, so situations like losing connection can be better handled than the case with centralized servers. The game network will also be less susceptible to failures and attacks.

2.1.2. Cost

For this rock–paper–scissors game, we have made the following estimates for the various costs associated with its development and maintenance.

- ❖ We need two laptops with wireless connections which have access to a remote Linux system throughout the development procedure.
- ❖ We need a database server to store user credentials and records, if we decide to implement such a feature.
- ❖ Two programmers with experience in web development are needed to implement the P2P network for the platform and perform maintenance and patching after the product release. They do not necessarily need to have game development experiences since the game itself is very simple.
- ❖ Since the game is not mainstream, we need to hire a marketing team that specializes in advertising our product in online communities such as BBS boards that have a high concentration of our targeted players.

2.1.3. Schedule

We have decided to use a complete RUP life cycle for the development and deployment of the rock-paper-scissors game. We plan on using a single iteration for the inception, elaboration and transition phase but two iterations for the construction phase. The total life cycle should take a month to finish if we have good progress, otherwise we will skip the second construction phase. In more details, analysis and design will be performed mainly in the first week. Testing will start in the third week, while part of the program is still under developing, for earlier adjustments if necessary.

2.2. Requirements

We have worked out the following requirements for our game.

2.2.1. Operational Requirements

- System Features: Each player needs system security and should has access to his/her account in a secure way.
- Technical Supports: The developer team should mark reported bugs by priority and fix them in time. The team needs to schedule and implement system updates according to user feedback.
- Help Desk Supports: We need to set up a customer support email for players to report bugs and ask questions about sign in, system connection issues, etc. User feedback and data needs to be collected and directed to the engineering team.
- Hardware Supports: The engineer team should maintain a central database that contains all user info.

2.2.2. Functional Requirements

2.2.2.1. Player Module

- Login: Each player needs to have a unique pair of login username and password. The user also needs to setup an email address for verification.
- Forget username or password: players should be able to retrieve their username by providing identification information to the customer support and reset their password via a password reset email sent to their associated email account.
- Game History: Players should be able to review their game records, including game time, win rate, etc.
- Global Scoreboard: Players should be able to view a scoreboard which contains the top players currently in the P2P network in terms of winrate.

2.2.2.2. Game Module

- Game Menu: Once a user starts up the P2P game platform, a game menu needs to be provided for the user to choose the game and its playing mode.

- Pairing: After choosing a game, player needs to be paired by the P2P platform with another player who are also waiting to be paired for the same game.
- Game Helder: The actual game will be performed by a ‘game helder’, building a one-to-one connection between the pair of players. User reaction and other game data should be sent through this connection during the game procedure to decide the game result.
- Ending Choice: When the game ends, the users can request the opponent to play again or quit game.

2.2.2.3. P2P Network Module

The game platform needs a P2P network module which contains the network infrastructure of the platform and runs in the background to support other game functions.

- Initial Peer Nodes: We need to provide several peer nodes which will always be in the network. The software will contain a list of such nodes, as well as the first peer to contact for a new user to join in the network.
- Network Builder: After contacting the first peer in the network, the user node needs to construct its own network by finding other peers via the known peers in a recursive fashion. To avoid having too large a network, the build process should terminate after reaching a maximum number of peers that one node can remember.
- Pairing: players should be able to reach out to other nodes in the network and attempt to pair with them. If it detects another peer that is also actively pairing, they should succeed in paring with each other.

2.2.3. Input Requirements

Players should be able to interact with the keyboard to type commands to sign in, choose game, and perform various game operations.

2.2.4. Output Requirements

The game platform should provide feedback in the form of messages printed to the console for every change the game takes, which includes:

- Information on the opponent after pairing establishes.
- Updated scores after both players have submitted a move.
- Disconnect information if the opponent quits the game.

2.2.5. Process Requirements

The system should carry out the following processes:

- Data Storage and Integrity: The system should store all players’ detailed and necessary information in an online database. All incomplete and unfinished transactions should be reported.

- Data Validation: The game client programs should constantly validate the current game's state in order to prevent data loss or cheating.

2.2.6. Hardware Requirements

- A simple computer containing a command line terminal is required to play the game.
- Wireless connection is required to access the game platform.

2.2.7. Software Requirements

- Operating Systems: Any one of these is required:
  - Windows XP/7/8.1/10
  - Linux
  - Mac
- Python runtime: The system should contain a usable Python interpreter.
- Terminal Application: A command line terminal is required, including but not limited to:
  - Mobaxterm
  - qtconsole

2.3. Risks

We have determined the major risks for our project as

2.3.1. Technology Risks

This system needs specialized hardware and software to support and there may have several kinds of risks, such as hardware unavailability, storage size underestimate, requirements changes.
To prevent these risk, we need to fully understand our requirements and prepare well.

2.3.2. Organizational Risks

The time frame might be interrupt by sudden difficulties, especially during the develop and test process. For example, an unexpected bug could cause extra several hours that suspend the plan.

2.3.3. People Risks

For team members, they may not have fully skills to develop the whole game or fix all bugs. In this case, team members should ask professional people for advice. On the other hand, the team member may unavailable at critical times.
To prevent this kind of risk, we may start the project as early as possible and extend our timeframe to develop this software.

2.3.4. Requirements Risks

Due to the lack of experience in software development, the team might underestimate or miss some of the requirements. Also, the requirements might change when the game platform used by

actual players. In this case, the team member need to grab the problem at the first sign and handle it.

2.3.5. Estimation Risks

During the development, teams may found that they are underestimate the time required to development the software and the rate of defect repair.
To prevent this risk, we suggest the develop team have professionals to have their project examined. If for some cases it happens, the team members should adjust their plan and make sure to finish the game within the timeline.

2.4. Other Aspects of the Project Plan

2.4.1. Quality Control

We have the following quality procedures and standards that will be used in this project.

1. This P2P game could support computer of different platforms listed above.
2. This P2P game procedure is logical and convenient for players to follow and play.
3. This P2P game can give right judgement for each game procedure and decide the winner of each game.

2.4.2. Staffing

There are two members to build this P2P game and they have the following skills and experience.

1. Shuidie Yao: College level software engineering and design.
2. Hui Duan: College level software engineering and design.

2.4.3. Communications

We have the following communication rules to develop this game.

1. Each team member should have successful and efficient communicate with each other regarding the project process and what risk and requirements achieved.
2. Each team member should stay on project timeline and communicate with others.
3. Each team member should make quick reaction with bugs and communicate to solve fix the bugs.

2.4.4. Development Tools

1. We update code and control version on Github.
2. We communicate through Wechat.

2.5. Conclusion

After considering the project plan in detail, we believe the game's potential outweighs the risks and the requirements can be reasonably met within the time period set out by the schedule.

3. ELABORATION PHASE

In the elaboration phase, our main focuses are fleshing out the use cases, building an executable architecture and writing iteration plans for the subsequent phases. Since the iteration plans would share a lot with this report, we have opted to not write down the specifics of the plans to save time. This section will therefore focus on the first two topics.

3.1. Use Cases

The use cases we identified are provided below in standard use case documentation format.

3.1.1. Player Login Use Case

Use Case ID: 4.1.1

Use Case Name: Player Login

Relevant requirements: Functional specifications section 2.2.2.1

Primary Actor: Player

Pre-conditions: Player opened our game through a terminal

Post-conditions: Player will be able to explore the game platform

Basic Flow or Main Scenario:

1. The system prompts to request player to type in username and password, also showing an option of new user registration
2. The player enters username and password subsequently
3. The system queries the central database for the correctness of email and password.
4. The credentials are correct, so the system retrieves player's records in the database.
5. The system reports player for a successful login and present main menu to the player

Extensions or Alternate Flows:

1. The system prompts to request player to type in username and password, also showing an option of new user registration
2. The player choose to register as a new user
3. The system prompts to let player enter username, password, and an e-mail address.
4. The system goes back to step 1.

Exceptions:

1. The system found that the username in the user input doesn't match password.
2. The system prompts the player to enter username and password again, also giving an option of resetting the password through email.
3. The system will send a link to the email. As the user type in new password, storing this password in the central database instead.

Related Use Cases: 4.1.1.2

Revision History --

Date	Description	By
5/10/2019	Player Registration Use Case	Hui Duan
5/11/2019	Update system behavior	Shuidie Yao

3.1.2. Player Encounter Use Case

Use Case ID: 4.1.1.2

Use Case Name: player encounter

Relevant requirements: Functional specifications section 2.2.2.1

Primary Actor: Player

Pre-conditions: Player already logged in the platform though correct credentials.

Post-conditions: Player will play the game with the encountered co-player.

Basic Flow or Main Scenario:

1. The system displays the main menu of a list of the games as well as an option to quit.
2. The user chooses the game to play
3. The system prompts the player that pairing procedure is on-going.
4. In the first iteration of the peer list, the system found another peer also in pairing status and build a pairing relation between the two peers.
5. The system notifies the player that a co-player has been found and start the game

Extensions or Alternate Flows:

1. In main flow step 4, the system finds that there is no available players to pair.
2. The system asks the player if the player want to try again.
3. If the player says yes, the system goes back to main flow step 4.
4. If the player says no, the system goes back to main flow step 1.

Exceptions: None

Related Use Cases: 4.1.1.3

Revision History --

Date	Description	By
5/11/2019	Play Encounter Use Case	Hui Duan
5/12/2019	Update system behavior	Hui Duan

3.1.3. Rock–Paper–Scissors Game Play Use Case

Use Case ID: 4.1.1.3

Use Case Name: Game Play

Relevant requirements: Functional specifications section 2.2.2.1

Primary Actor: Player

Pre-conditions: Player must be successfully pair with another player

Post-conditions: The system will redirect the player to the main menu.

Basic Flow or Main Scenario:

1. The system starts a round of game and notify user.
2. The system prompts to request player to type in some message for the opponent.
3. The player types in words for the opponent.
4. The system presents the opponent's message.
5. The system prompts to request players' to show hands
6. The player types 0, 1 or 2 representing rock, paper and scissor correspondingly.
7. The system reports the opponent's hand and report which player wins
8. The system reports the player's cumulative score.
9. The system decides whether there is a new round. If so, go to step 1.

Exceptions:

1. If the user input is invalid, the system will ask again.
2. If the opponent lose connection during the game, the system need to end the game immediately and redirect the user to main menu.

Related Use Cases: 4.1.1.2

Revision History --

Date	Description	By
5/14/2019	Game Play Use Case	Hui Duan
5/14/2019	Modify system behavior	Shuidie Yao

3.2. Architecture

We use an event-driven architecture for our P2P game platform. The event producers and consumers are the peers in the network, and the events are messages sent through internet channels (e.g. sockets). The application logic then consists of the different type of messages and their associated handler functions. The variety of the messages and handlers ensure that our architecture is both flexible and easily expandable. For example, adding another game such as Dice Roll to the platform would only require adding/changing a few messages and handlers. Implementation wise, the sample Python P2P file transfer application given by the instructor (http://cs.berry.edu/~nhamid/p2p/framework-python.html) has already provided an event-driven framework for the P2P network, so we simply copied them in our code.

4. CONSTRUCTION PHASE

Originally we planned to have two iterations in the construction phase, with the first one focusing on finishing the game logic and the second one on adding the user account interaction (username and password, game history, etc). Unfortunately we ended up only finishing the first phase.

The construction phase is where the bulk of our coding takes place. We closely followed the use cases established in the elaboration phase as guidelines in our development.

5. TRANSITION PHASE

For the transition phase, we mainly focus on bug-fixes and testing our feature requests. The test cases corresponding to the use cases given in the elaboration phase are given below.

5.1. Player Login Test Case

Test_Case_ID	TC_LOGIN_01	TC_LOGIN_02	TC_LOGIN_03	TC_LOGIN_04

Test Scenario	Verify the login system	Verify the login system	Verify the login system	Verify the login system
Test Case	Enter valid username and valid password	Enter valid username and invalid password	Enter invalid username and valid password	Enter invalid username and invalid password
Pre-Condition	Need a valid user account to do login	Need a valid user account to do login	Need a valid user account to do login	Need a valid email account to do login
Test Steps	1.Enter player username 2. Enter password 3. Click login	1.Enter player username 2. Enter password 3. Click login	1.Enter player username 2. Enter password 3. Click login	1.Enter player username 2. Enter password 3. Click login
Test Data	1.<Valid player username > 2. <Valid password>	1.<Valid player username > 2. <Invalid password>	1.<Invalid player username > 2. <Valid password>	1.<Invalid player username > 2. <Invalid password>
Expected Result	Successful login	A message "The email and password you entered don't match" is shown	A message "The email and password you entered don't match" is shown	A message "The email and password you entered don't match" is shown
Status(Pass/Fail)	Not implemented	Not implemented	Not implemented	Not implemented

5.2. Player Encounter Test Case

Test_Case_ID	TC_ENCOUNTER_01	TC_ENCOUNTER_02	TC_ENCOUNTER_03
Test Scenario	Verify the pairing system	Verify the pairing system	Verify the pairing system
Test Case	Enter valid game number and prepare valid peer	Enter invalid game number and don't prepare valid peer	Enter valid game number and don't prepare valid peer
Pre-Condition	Need a valid user account to login	Need a valid user account to login	Need a valid user account to login
Test Steps	1.Enter game number to play 2. pairing	1.Enter game number to play 2. pairing	1.Enter game number to play 2. pairing
Test Data	1. <Valid player input> 2. <A potential peer in the	1. <Invalid player input> 2. <No potential peer to be	1. <Valid player input> 2. <No potential peer to be

	peer list is waiting to be paired>	paired>	paired>
Expected Result	A message “Found player XXX. Press enter to continue” is shown	A message “Invalid Input. Please choose the game you want to play” is shown	A message “Can’t find another peer pairing. Try again?” is shown
Status(Pass/Fail)	Pass	Pass	Pass

5.3. Game Play Test Case

Test_Case_ID	TC_PLAY_01	TC_PLAY_02	TC_PLAY_03	TC_PLAY_04
Test Scenario	Verify message communication between the pair	Verify message communication between the pair	Verify the player action	Verify the player action
Test Case	Player message on both sides	Player message on one side	Enter valid action	Enter invalid action
Pre-Condition	Two players are already paired in a game	Two players are already paired in a game	Two players are already paired in a game	Two players are already paired in a game
Test Steps	On both sides: 1. Start the game 2. Input message	1.Start the game on both sides. 2. Send message on one side.	On both sides: 1.Start the game 2. Enter player message 3. Enter player action	On both sides: 1.Start the game 2. Enter player message 3. Enter player action
Test Data	2. <Some non-empty string>	2. <Some non-empty string>	2. <Some non-empty string> 3. <0,1 or 2>	2. <Some non-empty string> 3. <Invalid action input>
Expected Result	Both sides present message received and continue for next steps	The sending side waits the other side for reply; the receiving side still waiting for user input message	Both sides received the opponent’s action and the system present the result.	A message “Invalid Input, please try again” is shown
Status(Pass/Fail)	Pass	Pass	Pass	Pass