D4: Testing Plan – Group B

1. Scope Statement

The following document pertains to Group-B's test suite for the Communications Platform (CP) component integration delivered by Group-F to the UU-Game development project and the Game Engine (GE) and Game Platform (GP) components being implemented. However, as the GP development is not yet completed, there is no complete test suite for the GP yet.

The CP is responsible for setting up games and tournaments between different user and AI players both locally and online. To validate that this is what the CP does, we have implemented a test suite containing reliability tests and integration tests based on the user stories created for the CP integration. The reliability tests are used for testing specific functionalities of the CP, such as simulating an online game between two users. Whereas the integration tests are a bit vaguer, only testing things such as the processing of the input of invalid commands and other expected behaviour when running the program with specific input. As well as running the game multiple times with random input and making sure that it does not crash.

For the already developed GE component, we created a test suite of seventy-six already implemented unit tests of which test the reliability and validity of each GE implementation in respect to the user stories in the D2 Design Deliverable document.

2. Definitions

BASH:

: A command line interface terminal that connects the user to the UU-Game, enables input of commands and shows the current state of the game.

Tournament:

: A multi-game structure, where several games are played in parallel as well as consecutively. Winning players advance and continue with a new game. At the end one person is the tournament winner, when no more opponents remain.

PO:

: The product owner; entrusted by the customer to represent them in delivering their requirements by meeting with the development team and answering their questions.

D2:

The second deliverable focusing on the development design for each group's component during the first phase of the Agile development project.

AI:

Artificial Intelligence. In this case, an automated player controlled by the GE.

Local:

UU-Game implementation on a local system without the requirement on connecting to any online network.

Online:

UU-Game implementation on a simulated online network system.

Host:

The user providing the services and resources for a simulated online networked UU-Game.

Client:

The user accessing the services and resources from a hosted simulated online networked UU-Game.

3. Reliability Tests

For each of the six-reliability black-box tests conducted, each test was based on the user and system requirements formulated by Team-B in respect to the CP component. These user and system requirements are formulated from information sourced from the PO meetings and the available D2 documents detailing the design of the CP components during the first phase of the Agile development project.

The first of the reliability tests focused on the user story; 'As a user, I want to have a selection of game play options so that I can choose the type of game that I want to play'. For this user story, the first black-box testing conducted was primarily in focus of the game play options provided by the integrated communication platform at the initialization of the UU-Game. The objective of this test was to assess the reliability of the game menu options and prompts being displayed to the user at game initialization and immediately following each game that has completed being played. The procedure of the first black-box reliability test comprised of running the 'main' file of the integrated CP development in a BASH terminal by entering the command 'python3 main.py', of which generated the main game options to display on the terminal with a 'Welcome to <game>!' message at the top. Of the initial options, there was either a choice of playing a 'singles' (1 vs 1) game, a tournament game, or to quit the program. When selecting each of the singles or tournament game play options, there was then a provided option of selecting either to play a 'local' or a 'online' game, or to

return to the previous menu or to quit the program. From each of the ‘local’ or ‘online’ options, the CP took the user to further options of selecting whether each player in a game is a ‘human’ or an AI player and prompted the user to enter names for each player, whether ‘human’ or AI player. Under normal operation, the outcome of this testing was expected as each game play option was provided to the user. However, the ordering of the options seemed to be in a backward order than custom, while the prompting of a user to enter a name and player type for each player, whether ‘human’ or AI, seemed to be over demanding as this would be required for each game being played by a user. In comparison to this, it would seem more appropriate to prompt the user once at the program initialization for a name, and to then only prompt the user to enter further names for any additional players that are not AI players and are not other users in online games. Aside from this, the first reliability test resulted in the CP meeting the specifications for the user story requirements.

The second of the reliability tests focused on the user story; ‘As a user, I want to be able to start a local tournament game of any combination of AI or user-controlled players so that I can play tournament games against other players locally’. For this user story, the black-box testing followed from the prior testing in that the local option provided in the tournament option menu was selected while running the ‘menu’ file in a BASH terminal. The objective was to assess the reliability of the tournament and local tournament menu options, prompts, and generated output of tournament game play. When selecting to play a local tournament, the CP generated a message to declare that the user is now playing a tournament game and prompted the user to choose how many players, and of these players, how many are AI, and to enter names for user-controlled players. Following these options, the CP declares all the players in the starting tier, and then plays rounds between two players, declaring the players playing and the winners, or a draw, of which the play repeats until there is a winner, for each round until only one player remains as the overall tournament winner. Under normal operation, the outcome of this testing was expected in terms of game play options, and of each player in each tournament tier and each tournament round winner being declared until the tournament concludes with a final, single winner.

The third of the reliability tests focused on the user story; ‘As a user, I want to be able to either host or join an online tournament game of any combination of AI or user-controlled players so that I can play tournament games against other players online’. For this user story, the black-box testing followed from the first reliability testing in that the online option provided in the tournament option menu was selected while running the ‘menu’ file in a BASH terminal. However, in this case two BASH terminals were required to run as a host and as a client simultaneously while they are connected to the same online tournament game, otherwise black-box testing of an online tournament game would not be possible. The

objective was to assess the reliability of the tournament and online tournament menu options for each of the host and the client, prompts, connectivity and data transmission between host and client and generated output of tournament game play. When selecting to play an online tournament, the CP generated a message to declare that the user is now playing a tournament game and provided options for either hosting or joining an online tournament. Following this each user was prompted with the same options as in the local tournament and the tournament output was also the same. Under normal operation, the outcome of this testing was expected. The connectivity and data transmission behaviour were also as expected between host and client with connectivity being uninterrupted and instant in connection and disconnection, while each data transmission was reliable and completed in well under a second. Also, in terms of the output of the CP, the provided options for either being a host or a client were not so clearly depicted, and instead the CP merely prompted the user with, 'Are you the first to start the game? [Y]es [N]no'. Aside from this, the third reliability test resulted in the CP meeting the specifications for the user story requirements.

The fourth of the reliability tests focused on the user story; 'As a user, I want to be able to start a local 'singles' (1 vs 1) game of any combination of AI or user-controlled players so that I can play a 'singles' (1 vs 1) game against another player locally'. For this user story, the black-box testing followed from the first reliability testing in that the local option provided in the 1 vs 1 option menu was selected while running the 'menu' file in a BASH terminal. The objective was to assess the reliability of the 'singles' 1 vs 1 and local 1 vs 1 menu options, prompts and generated output of local 1 vs 1 game play. When selecting to play a local 'singles' (1 vs 1) game, the CP prompts the user to enter names for each of the two players followed by whether each player is human or AI. Messages are then generated to either declare a game won, or a game drawn, and if so repeats until a game is won, then another message thanking the player for playing. Under normal operation, this outcome was expected. However, alike to the prior tests, the repetitive prompting for the user to enter names for each player and define whether they are human or AI is unnecessary when there could be one prompt for a user to enter a name at program initialization, and following this, after entering the number of user-controlled and AI players respectively, prompts to enter names for an additional user-controlled player if there is more than one selected for the game. Aside from this, the fourth reliability test resulted in the CP meeting the specifications for the user story requirements.

The fifth of the reliability tests focused on the user story; 'As a user, I want to be able to start an online 'singles' (1 vs 1) game of any combination of AI or user-controlled players so that I can play a 'singles' (1 vs 1) game against another player online'. For this user story, the black-box testing followed from the first reliability testing in that the online option provided in

the 1 vs 1 option menu was selected while running the ‘menu’ file in a BASH terminal. The objective was to assess the reliability of the ‘singles’ 1 vs 1 and online 1 vs 1 menu options, prompts and generated output of online 1 vs 1 game play. Alike to the third reliability test, two BASH terminals were required to run as a host and as a client simultaneously while they are connected to the same online 1 vs 1 game, otherwise black-box testing of an online 1 vs 1 game would not be possible. Alike to the fourth reliability test, the user is prompted to enter names for each player and declare if the player is human or AI, and alike to the third reliability test, the user is prompted to declare if they are starting the game or not, instead of the likes of being offered to start or join a game, so it is more clearly understood what is being asked. Aside from this, under normal operation, the outcome was expected with the user that won being notified of winning and the user that lost being notified of losing and each being thanked for playing with messages generated by the CP. The connectivity and data transmission behaviour were also as expected between host and client with connectivity being uninterrupted and instant in connection and disconnection, while each data transmission was reliable and completed in well under a second.

The sixth of the reliability tests focused on the user story; ‘As a user, I want to have an option available to me while implementing the UU-Game menu and not playing a game so that I can quit the application at will’. For this user story, the black-box testing followed from the first reliability testing in that, aside from the main menu being of interest, each of the main options were also selected while running the ‘menu’ file in a BASH terminal. The objective was to assess the reliability of the ‘quit’ option in each of the main menu and ‘singles’ (1 vs 1) and tournament menus and the processing of the quit command in each case. Under normal operation, the outcome was expected for when in the main menu, 1 vs 1 and tournament submenus, and the online 1 vs 1 menu. However, when in the online tournament and local tournament menus there is no quit option available. Aside from this, the reliability test met the specifications for the user story requirements.

4. Integration Tests

The first integration test involved the local tournament game play which is tested in the second reliability test following its integration with the GE. The objective of this test was to not only assess the reliability of the local tournament game play, but also the validity of the menu options, prompts, processing of commands and the generated output of the tournament game play for each of user vs AI, user vs user, and AI vs AI in easy, medium and hard difficulty settings with tournaments of between 3 and 8 players. Under normal operation, the outcome was not expected for when entering the number of total and AI players, if entering a non-integer command, the application will crash due to there being no

implemented exception in the validation of the command. Following a fix of this bug, the outcome was expected in respect to the integration of the CP component with the GE for each of the AI vs AI, AI vs user, user vs user in any AI difficulty and for any number of players between 3 and 8.

The second integration test involved the online tournament game play which is tested in the third reliability test following its integration with the GE. The objective of this test was also to not only assess the reliability of the online tournament game play for each of the host and client, but also the validity of the menu options, prompts, processing of commands, the connectivity and data transmission between host and client and the generated output of the tournament game play for each of user vs AI, user vs user, and AI vs AI in easy, medium and hard difficulty settings with tournaments of between 3 and 8 players. Under normal operation, alike to the first test, the outcome was not expected for when entering the number of total and AI players, if entering a non-integer command, the application will crash due to there being no implemented exception in the validation of the command. Following a fix of this bug, the outcome was expected in respect to the integration of the CP component with the GE for each of the AI vs AI, AI vs user, user vs user in any AI difficulty and for any number of players between 3 and 8. The behaviour was also expected, with each data transmission not only being reliable and efficient, but also valid, and the connectivity did not fail in its connecting and its teardown.

The third integration test involved the online ‘singles’ (1 vs 1) game play which is tested in the fifth reliability test following its integration with the GE. The objective of this test was to not only assess the reliability of the online ‘singles’ (1 vs 1) game play, but also the validity of the menu options, prompts, processing of commands and the generated output of the game play for each of user vs AI, user vs user, and AI vs AI in easy, medium and hard difficulty settings. Under normal operation, the outcome was expected in respect to the integration of the CP component with the GE for each of the AI vs AI, AI vs user, user vs user in any AI difficulty.

For each of the integration tests, there is still an integration error causing games to not correctly conclude which does not relate to the CP development, in which the refactoring to include the checking if a game is won or drawn is currently in error and was unable to be fixed prior to the completion of the third sprint. The GE implementations themselves are all already proven to be reliable and valid with abundant and thorough unit testing, and it is merely the refactoring of the CP to integrate these GE implementations that is in error. This is expected to be fixed early next week.

5. Other Tests

The only other type of testing implemented in the development of the UU-Game by Team-B is unit testing, of which there are currently eighty-five implemented unit tests with passing results for each of the GE and GP implementations. Seventy-nine of these unit tests are for the GE, of which development has been concluded. Six of the unit tests are for the GP, five of these being for the scrapped GUI GP development. The remaining test is for the first implementation of the CLI GP development, which is to validate the ASCII representations generated in the GP Piece class for each of the sixteen unique pieces with four binary characteristics. There are a further four tests which were integrated with the CP, of which only three are passing, and each of which these unit tests are testing have already been tested in the reliability and integration testing.

The seventy-nine GE unit tests thoroughly test almost each method in each implemented class in the GE development. The Game class is first tested for reliability and validity in the generating or sixteen unique game pieces with four binary characteristics, the game board, and that the Game class returns correctly if there is a next game play. Further Game class testing is of the ‘bin_count()’ method which counts the similarities between pieces on the board for both checking if a player has won a game and for the easy and hard AI algorithms with twelve unique tests for this. The next method tested in the Game class is the ‘has_won_game()’ method which checks if a player has won a game at the end of each game play turn with twenty-two unique tests for this.

The next class tested is the Play class, with tests for methods such as ‘play_auto()’ ad ‘change_player()’, and variables such as the ‘players’ list and the ‘current_player’ and ‘selected_piece’ strings. The ‘init_players()’ method is next tested with seven unique tests for each of the available game modes and difficulties.

The next class to be tested is the PlayerAI class, with ten tests for methods such as ‘max_similarities()’, ‘select_best_sim_count()’, while the other methods in PlayerAI are tested in the tests for the PlayerEasyAI and PlayerMediumAI subclass of PlayerAI. PlayerEasyAI has five tests, of which two are tests for the average game play wins out of 100 tests for PlayerEasyAI vs PlayerMediumAI and PlayerEasyAI vs PlayerHardAI to test PlayerEasyAI loses at least 90-95%, while two other tests are for the ‘choose_piece()’ and ‘place_piece()’ methods. PlayerMediumAI has six tests, three of which being for the average game play wins out of one hundred tests for PlayerMediumAI vs PlayerMediumAI to test the average wins are approximately 50% and wins at least 90% against PlayerEasyAI and loses at least 90% to PlayerHardAI, while there are also two other tests for each of the ‘choose_piece()’ and ‘place_piece()’ methods. The PlayerHardAI only has four tests, and

none of these tests include any methods in PlayerAI as PlayerHardAI implements the Minimax class and does not extend the PlayerAI class. The first test is to test that PlayerHardAI on average doesn’t win against itself more than 10%. Th next is to test it wins at least 95% against PlayerEasyAI. Of all the unit testing for the UU-Game development, aside from PlayerHardAI testing, each unit test runs in optimal millisecond time without issues. However, the PlayerHardAI can take close to thirty minutes for each test that includes one hundred games to calculate an average win value, whereas for each of the PlayerEasyAI and PlayerMediumAI tests including one hundred games that are not tested against the PlayerHardAI algorithm take approximately ten seconds each, with each game tested taking on average only a tenth of a second.