Testing Report

Testing Methods and Approaches

Throughout the implementation process, the group will be testing the game using unit tests to ensure that the new code being written runs as expected. This test-driven strategy also means that if more code is added, the earlier tests can be quickly run to confirm that the new addition does not break previously working sections. Also, the unit tests being used will be white-box tests, meaning that the tests have access to the code and are not just concerned with the 'front-end' inputs and outputs. This white-box method is the most appropriate for our implementation phase because it allows the developers to more easily find the source of an error and make amendments immediately after discovery.

Once the features required for the preliminary implementation have been completed, a general black-box testing phase will begin, which aims to test higher level aspects of the software, such as user requirements and GUI elements. The black-box method will be the most suitable for this stage of testing because it allows the group to see the game in the same way as the users.

Also, because this phase ensures that the game works as expected for the users, if the group were able to view the software whilst testing it, accurate results may not be achieved as each segment of the game could be assessed individually. Consequently, this prevents the group from amending the software during the general testing phase, which ensures that passed tests are not affected by any changes until the end of the phase. This also allows the whole group to see which tests failed before any amendments are made, which could outline and help prevent any possible domino effects.

The tests will be displayed in a table, each table will be categorised depending on what tests it contains: Unit, Requirements or GUI tests. Using these tables to display the tests means that it is easy to see which tests were carried out, as well as the results and evidence for each. Furthermore, these tables will allow the group members and the client to see which requirements have been met and which tests have failed at a glance. For the requirements testing, some tests are not testable at this stage, these tests are identified in the Requirements testing document.

The series of tests will be split between all the group members and the results will be recorded alongside the test, stating whether the test passed or failed. If a test fails, it will need amending, not just ignored, and if a test were to fail, additional notes will be required on the table to record what the issue was. Then during the amendment phase the notes can be used to help fix the issue correctly.

Once the general testing phase has been completed, any amendments would then be made. Depending on the number of failed tests, the work would be split between all the group members and the corrections could take place. It would then be the job of each group member to effectively test their amendments, not only to ensure that the test can now be passed but also to ensure that the other tests do not fail as a result. Any resulting issues could then be recorded and amended as required.

Whilst the game is only partially implemented it would not be appropriate to use play-testing because the results gained would not be useful as only a fraction of the features will be developed in this stage.

Report on Tests:

Test Statistics:

Test Type	# Tests	# Tests Run	# Tests Failed	% Passed
Unit	23	23	1	95.7 %
Requirement	31	17	5	70.6 %
GUI	13	13	2	84.6 %
Total	67	53	8	84.9 %

Unit Tests:

The one unit test that failed was U23, which tests if players are assigned the correct number of gang members at the start of the game (= number of sectors owned * 10). This test failed initially because the player's GangMembersLeft attribute was fixed at 50.

This issue has been fixed with the addition of new methods that count the number of sectors owned by a player and successfully change a player's GangMembersLeft attribute.

Requirement Tests:

As part of the general testing phase, a number of tests were created to ensure that the requirements have been met. However, as this is just a preliminary implementation phase, only tests for requirements applicable to the current level of implementation have been run. The tests not run are REQ-1.4, 4.3, 6.3, 7.1, 7.4, 7.6, 7.7, 8.1, 8.3, and NFRQ-1.2.

Additionally, some of the requirements cannot be tested or are not necessary to test. For example, ORQ-1.1, which states that "A single game must last for at least 20 minutes," is not currently testable because it requires multiple players and turns to be implemented.

Furthermore, having a formal test for requirement NFRQ-1.3 is not necessary because there are no security features present in the game. Also, REQ-2.2 is not testable using black-box tests because the information needed for this test is hidden from the users.

Overall, five of the requirement test failed. One of these tests is REQ-5.2, which tests that the map should be separated so that at least 75% of the sectors should contain university buildings. This test failed because only 60% of the 110 sectors contained university-owned buildings. To pass this test, the map needs to be redrawn and the sector icons need updating to match. However, the campus map contains a lot of 'empty' space between the two campuses, so to balance the gameplay the space was split into smaller sections, creating a more even coverage of sectors throughout the map.

Another test which failed was REQ-6.2. This tests that the GUI displays the UoY map, a representation of Sectors, Sector ownership and number of gang members in each sector, and failed because the representation of the number of gang members has not been implemented. In order for this test to pass, the user must be able to see the number of gang members present in each sector.

Additionally, REQ-7.2, which tests if players have one turn per round and take their turns in order, failed. This is because currently there is no way of moving from a turn's attacking phase to the movement phase, and then to the next player's turn. In order to pass the test, a method for checking if a turn phase has been finished should be implemented. This functionality was not necessary for the preliminary implementation phase, so the additional programming was not carried out.

Also, the test for REQ-7.3 failed. This test is not required for the current, preliminary implementation, however it relates to the functionality of Sectors, which are a part of the Map. The test checks if a player is able to move gang members between sectors and does not pass because currently the GameState does not change automatically, and if the GameState is manually changed, the movement functionality has not been implemented (only the valid sectors to move to are highlighted). For this test to pass, the changing of turn phases (from attack to movement) needs to be implemented, as well as the movement of gang members between Sectors.

Finally, the last test to fail was for requirement ORQ-1.2 and tests to see if the user should be able to play two turns within 2 minutes. The test failed because there is currently no way of progressing turns past the allocation phase. For this test to pass, there must be a way to switch to the next turn phase, i.e. attacking to movement. Also, a time limit should be added to the turns so that the game progresses at a reasonable pace.

GUI Tests:

Out of the 13 tests that were run only two of them failed. These two tests were G8 and G9, which test what happens when an invalid input is entered into the gang member allocation input field. G8 tests if a player tries to allocate more gang members than they have left to allocate, the input field should flash red, and G9 tests if the flashing occurs when a player tries to allocate a negative number of gang members.

Both of these tests failed because the input field does not return to white after turning red. In order for this test to pass, this issue must be fixed, however, it is a rather low priority bug, so could be implemented at a later stage.

Testing Material:

To see the tests run during the testing phases and the outcome of each test, please use the following links.

Unit Tests:

https://github.com/as2378/unlucky/raw/master/docs/files/Assessment2/Unit_Tests.pdf

Requirements Testing:

https://github.com/as2378/unlucky/raw/master/docs/files/Assessment2/Requirements_Tests.pdf

GUI Tests:

https://github.com/as2378/unlucky/raw/master/docs/files/Assessment2/GUI_Tests.pdf