1.0 Testing Methodology and Justifications

During this phase we have refined our testing approach after considering the advantages and disadvantages of both our own testing methods, and that of the previous team (DRTN). Our previous testing methodology can be found here [1], whilst that of DRTN can be found on here [2].

While considering both testing methodologies we discovered that DRTN had a similar approach to conducting Unit Tests, and as such this section of our methodology has remained unchanged. The migration of technologies (from C# to Java) required the re-configuration of Travis so that the JUnit tests were run automatically, as we had for our previous project. Despite this change, our testing workflow for the production and maintenance of unit tests has not been affected by our change in project.

DRTN faced did not document integration testing on the project, and found issues testing certain classes - which were overcome through manual testing [2]. However, we have chosen to undertake a different approach to handling these issues. We however believe that the combination of progressive and big bang testing that we have been using since the previous phase [1] to be more efficient and better suited to our team. Big bang testing has the potential to be quicker than conducting exhaustive manual testing, but can provide difficulties when attempting to isolate specific errors. However, due to the small amount of changes made, it's unlikely that isolating errors will prove challenging. In addition, through conducting progressive testing during development we reduce the likelihood of facing complex integration issues, meaning that what errors proceed to the big bang phase are likely to be relatively simple. Despite our different views on integration testing, we will continue to use the previous tests provided by DRTN alongside our own methods (applied to all of our additions/changes to the project) to ensure that all classes receive sufficient testing and monitoring.

Similar to DRTN we recognise the importance of conducting acceptance testing to ensure that the produced product complies with the client's requirement. In the previous project we followed a similar procedure to that of DRTN, with the exception of multiple testers. During that project, acceptance testing was conducted as a team with the consensus on whether a requirement was met being the test result. We have adapted our methodology such that each acceptance test requires all members of the team to post their opinion on whether our implementation have met all use cases and their functional/non-functional requirements. A test is said to pass if 5 of the 6 team members believe we have met the requirement, it is consider a weak pass if 3 of 6 agree (i.e. we're unsure as to whether a requirement has been met sufficiently) and fails otherwise. The user cases and the results may be viewed here [3]. The results are conditionally formatted, with passes coloured green, weak passes yellow and failures red.

2.0 Testing Summary

Through the use of unit and integration testing we hope to ensure that our product is of the highest quality. The team determine a product to be of high quality when each class has been tested extensively (with all of these tests passing) and that the physical implementation of these classes are efficient, concise and well documented. To determine that the final product is faithful to the customer's expectations we have created an acceptance test for every requirement.

These tests have also been used to detect, and help in resolving, any errors or semantic issues. Of these tests most were inherited from the previous team, having reviewed these tests, and being satisfied with the quality we have made several additions to address our extension to the product. The unit and acceptance tests were expanded to include new classes and requirements respectively.

The unit tests are comprised of tests for the ResourceGroup. We felt it important that our tests of the ResourceGroup were thorough as this new data representation class is used throughout most classes. These test were focused on ensuring that the mathematical operations functioned as intended and in a predictable manner, thus ensuring that features such as market trading and random events function as intended.

Despite the work conducted on the AI and Market classes there are no new unit tests. These classes underwent refactoring and as such there was no change to their core functionality. However, in the case of the new helper methods, their use of external classes, such as RoboticonQuest, mean that they cannot be covered by unit tests. Therefore, these methods were tested by progressive testing during development and went largely undocumented (see previous testing report for rational).

In regards to the implementation of the chancellor phase (i.e. the ChancellorScreen and ChancellorActor classes), testing was conducted by visual inspection and through the use of break points. The testing of this class was conducted in this way due to its integrated nature which could not be reliable tested through a series of unit test. These results are available on our website here [4].

Due to the requirements change during this phase, new acceptance tests were made to ensure that the implementation of these requirements were to specification. As such, acceptance test 12 and 13 were created which correspond to the requirements of multiple players and a chancellor mode. The non-functional/functional requirements were elicited from the client during the development period, and the user story was generated by our Product Owner.

3.0 Test Results and Discussion

3.1 Unit tests

All 37 unit tests passed results [5].

3.2 Integration Tests

As discussed above the majority of the new integration tests have not been documented, but we ensured that all classes underwent this testing with all issues having been addressed. Again the results for the tests conducted on the chancellor phase, can be found here [4].

3.3 Acceptance Tests

All 13 acceptance test have passed.

3.4 Conclusion

In conclusion, due to the passing of all tests we consider this product to be of high quality and are confident that it will meet the client's expectations.

Bibliography

[1] SEPR, “Test3.pdf,” [Online]. Available: https://seprated.github.io/Assessment3/Test3.pdf. [Accessed: 29 April 2017]

[2] DRTN, “Test3.pdf,” [Online]. Available: https://nicopinedo.github.io/SEPR4/Assessment3/Documents/Test3.pdf. [Accessed: 29 April 2017]

[3] SEPR, “Acceptance Testing,” [Online]. Available: https://seprated.github.io/Assessment4/acceptance-testing.html. [Accessed: 29 April 2017]

[4] SEPR, “Chancellor Testing,” [Online]. Available: https://seprated.github.io/Assessment4/chancellor-testing.html. [Accessed: 29 April 2017]

[5] Travis, “Tim020/Seprated-DRTN-Fractal,” [Online]. Available: https://travis-ci.org/Tim020/Seprated-DRTN-Fractal. [Accessed 4 May 2017]