1.0 Change Management

The first stage of our change implementation involved understanding the original developer's software that we were taking on. To do this we retrieved BugFree's code from their GitHub repository and looked through it as a team. To further understand the working of the game we also downloaded the documentation from their website. Multiple members of the team worked on making changes to the game, therefore we adopted formal approaches to coordinate and track these changes. Our change management process utilised methods and tools, such as regular meetings and GitHub, to follow the IEEE Software Configuration Management Plans [1].

Both as a group and individually the team decided which elements of BugFree's project we thought we should adopt, and where we might be able to merge our Assessment 2 requirements and features into the project**.** This was done for the most part on a continuous basis, however some sections such as requirements were largely decided upon before work began. All team members analysed BugFree's documentation and code, feeding back anything notable or that needed to be changed for the current assessment using the Delphi Technique [2]. These changes were then assessed against project goals and if accepted, assigned to a team member to implement. Regular meetings and discussions made sure that every team member was up to date with any changes in order to mitigate any confusion.

For the team to work on development concurrently, we have used branch management as part of the IEEE Software Configuration Management Plans [1] so that different features of the game could be developed at the same time. Each new feature was developed in it's own branch and then merged into the main development branch. This is essentially the same method as used last assessment, however the extent to which we branched in conjunction with GitHub issues together greatly increased this assessment. We also used ZenHub to track bugs and features that needed to be fixed or implemented.

Traceability is key when making changes to our deliverables, hence as a group we have highlighted anything that has been changed, removed or added to all documents. This can be seen in the updated method, risk, GUI report and testing documentation.

2.0 Changes to Documentation

2.1 GUI Report Changes

The updated GUI report can be viewed here [6]. All changes can be identified by their formatting, any new additions will be black text on a green background like so: changed text. You will be informed of any removals by black text on a red background like so: REMOVED NOTIFICATION. We have included removals to ensure traceability.

The GUI report provided by Bugfree summarises and justifies their initial design for the game. We did not make any changes to their initial GUI design and therefore few changes needed to be made to the GUI report provided. In fully implementing other features of the game, new GUI elements were added. This meant we have expanded the GUI report to explain and justify these new features and formatted the report in line with the rest of our documentation. All changes have been made traceable using the formatting shown above and linking to requirements [Requirements Document 3].

The main additions to the GUI report involved the market. The player can access the market on the auction or purchasing stage of the game. The market GUI shows the market buying and selling prices, the number of resources or roboticons owned by the market and player to player trading (requirement 2.2.2.b). BugFree had not yet implemented many features of the market as it had only been developed to a basic level. The new GUI now includes dynamic buying and selling prices that work on demand and supply economics (requirement 2.4.a). These can be seen in the pop up window at the side of the screen. The player can easily input the amount of a resource or roboticon they wish to buy or sell. Player to Player trading has also been implemented in the auction phase of the game. The market then includes a large window where the user can decide which resource they wish to trade, the total amount and set their own selling price. All trades can be viewed in this window from the opponent, with quantity, unit price and total price.

Another feature of the GUI that we added was the casino. The casino can be accessed when the player is in the auction phase of the game by clicking on the button in the GUI bar. The casino includes a random number generator where the player can input the amount of money they are willing to gamble. By clicking ‘Roll’ they will randomly be pronounced a winner or loser.

We have also added random events to the game. Random events are also incorporated in the GUI to make it clear what has happened and who is affected. These events occur randomly in the acquisition phase of the game. The events are made obvious and clear (requirement 2.5.c) in red writing. One event also features a large Donald Trump image when he deports all of a player’s roboticons.

2.2 Testing Report Changes

The updated testing report can be viewed here [7]. All changes can be identified by their formatting, any new additions will be black text on a green background like so: changed text. You will be informed of any removals by black text on a red background like so: REMOVED NOTIFICATION. We have included removals to ensure traceability.

In adopting a different game from another development team, new tests needed to be designed and run. The majority of the Test Summary section of the Testing Report has been changed to include the results for our new game. The results for our previous game have now been removed because they were irrelevant to our current software. The methods and approach have not changed and we have kept the same, clear presentation format to demonstrate our results. With the feedback from the previous assessment in mind, we have updated our report to fully justify why the kind of testing we used was justified. This has been added to the three different types of testing that we used to explain in more depth to the client.

Tests to do with networking were removed as the game we adopted from the original developers does not involve networked play. This removes the issue in conducting unit tests that involve networking, which slowed our testing progress in the previous assessment. The new acceptance tests were added to the report, some of

these failed, however this were largely due to updated or removed requirements - making these tests redundant.

The testing environment that includes the hardware and software used to execute test cases was also added to give context into the conditions used. It is important that the client understands the conditions used in case they are having issues running the game in a different environment in which the game has not been fully vetted.

The previous team did not use Unity Test Tools (UTT) to conduct unit and integration testing so we decided to port their tests to UTT. In doing so, we did not remove any of the original tests but this did allow the tests to be run through the continuous integration server or from within the Unity editor itself. We also added our own tests following the same specification as used last assessment.

After implementing the AI for our game, more unit tests were needed to observe the operation of our logic. This allowed for testing of automated actions seen within the human class guided by our crafted logic. However this proved problematic when considering the random nature of certain actions (as discussed in the implementation report). To simulate and observe the possible behaviour of our AI we ported the logic into python and provided it with sample inputs and observed its actions. Whilst this provided a certain insight into the performance of our probability functions and the frequency of certain events, it required large amounts of time and detailed analysis to ensure that it was behaving as necessary. To overcome this, and to expose the AI to more realistic environments, we created an environment where two AI could play against each other this allowed us to observe their behaviour visually and also allowed for a quick means of monitoring other elements of the game such as; changes in market prices, event functionality and frequency, etc.

In order to write some of our unit tests we had to create dummy classes which extended the normal ones, for example the class DummyTile is a child class of Tile. The reason for needing these classes is that some methods interact directly with the user interface, in the case of the tile when it is acquired it also changes the overlay colour to match. This behaviour causes the test to fail because there is no UI game object instantiated, so in our dummy classes we implement the methods in the exact same way and just remove any calls to the UI. We decided this was allowable as the unit tests were to test the logic of the functions, and we would observe and test any UI changes through acceptance testing.

2.3 Method and Plan Changes

The updated method documentation can be viewed here [8]. All changes can be identified by their formatting, any new additions will be black text on a green background like so: changed text. You will be informed of any removals by black text on a red background like so: REMOVED NOTIFICATION. Lastly the changes made during the last phase are represented by black text on a yellow background like so: previously changed text. We have included removals and previous changes to ensure traceability.

The first change is the choice to adjust our implementation methodology (*see section 1.2)*, that being the creation and adoption of open plan programming (can be found here [3]). From the previous phase we discovered that while pair programming proved beneficial in terms of code quality and the reduction of erroneous code, implementation was progressing slowly. Therefore it was only natural that we adapted the way we approached code creation/maintenance. The creation of open plan programming is built on the strengths of pair programming (i.e. people in the same place able to work on issues that may arise together) but allows for individuals to be working on different tasks. Whilst this method has proven effective, it was only possible as all members are now confident in their abilities with the development language (C#) and are comfortable asking for assistance when required. This may have not been possible earlier in the project as this requires a high level of trust and confidence in both your team's and your own proficiency.

Whilst not a change that affects our methodology, and as such has not been entered into the method document, we thought the development should be recorded nonetheless. There are now two primary IDEs in

use by the team members (MonoDevelop and Visual Studio), whilst this has yet to lead to any compatibility issues (for example different encodings of line endings) this is a possibility that must be considered. To avoid any possible issues the team's use of the software is restricted to common functionality found in most IDEs (refactor variables, auto-generated comments etc). Any unique IDE features, such as the Visual Studio code generation (from diagrams), should be avoided lest they cause unforeseen issues. Aside from this there has been no change to the set of tools as all of our current functional needs are being fulfilled.

The next change in our methodology concerns our organisational structure (*see section 2.0*). The assignment of method manager to Matthew Dunn has ensured that we now have a clear and structured approach to the core of our change management. This assignment along with the inclusion of new responsibilities for the Risk Manager and Product Owner (found here [3]) mean that all changes are now documented by the responsible individuals, as they happen with detailed information regarding their justifications. This change was made to accommodate the feedback received on our previous assessment and ensure that our updates are clear and explained in detail.

The first removal in our method documentation (*see section 2.0*) has ensured that interested parties are presented with only the relevant job information i.e. for project switching; potential 'buyers' will only be interested in who completed what in this phase should they have any queries, and not the work that took place during previous assessments.

The overall project plan presented in assessment one has not changed. We have provided links to the detailed plan for assessment four which contains a Gantt chart showing task relations and a backlog containing task details (this can be accessed here [4]). Through producing a detailed plan we hope that the next phase will proceed in a timely and predictable manner. The plan will once again be hosted on a public website such that any interested party may access details about our estimated progress and position in the project's lifecycle.

It was clear that the previous development team's method document was specific to the way their team worked, rather than our own. Hence we decided to keep our own method documentation and add to this any changes in the way we worked as a team. As well as this, many of the risks in the previous development team's documentation were specific risks to their team. We have decided to keep our own risk documentation and add any risks from BugFree's document that we felt were important.

2.4 Risk and Mitigation Changes

No changes were made to our risk presentation. We believe the tabular, colour coded format previously used to present our risk tracking and mitigation scaled well to accommodate newly identified risks as well as those previously identified by BugFree.

No changes were made to the team's approach towards risk management. As a team we continued to use the condition-transition-consequence (CTC) format [5] to allow a standardised systematic approach to defining risks as they arise. As done in previous assessments the team engaged in a discussion and brainstormed new risks. Our current system allows each team member an opportunity to identify a risk applicable to them. We decided this was important because of our varied skill levels and approach to dealing with the changes in assessment 3. The team has decided that our current approach was sufficient for managing and tracking risks. Team members didn't want to adopt an unknown method used by the previous team. This was decided as this would have created unnecessary work and was itself a risk.

Changes were made to the risks tracked. Assessment three gives the team code written by another team, this provides a plethora of new risks to manage. New risks that were not applicable to the project previously were identified and added to the risk tracking spreadsheet and managed in the standard manner.

Following are the newly identified risks:

- Given that we are now working with code written by another team there is a concern that it might not function as advertised.
- Given that we are now working in 3D there is a concern that team members might not be able to work with 3D graphics resulting in delays.
- Given that the team has to work with unknown code there is a concern that the additional workload required to understand it will delay the work

The only changes made to our mitigation process and techniques were to accommodate new risks. The team followed the same method for risk identification, mitigation and avoidance as in previous assessments. This approach to risk mitigation fit with our agile methodology as some tasks were completed in groups, and as such we have identified methods for mitigating and avoiding risks that work for the team. Some tasks, however, were completed individually which was reflected in the mitigation methods that the team decided upon. The mitigation techniques which the team has decided to use work effectively for both individual and team work tasks. Individual team members were asked to report back to those with a greater understanding of the code to better help everyone understand and work with the inherited code. As a regular team meeting became more difficult to organise, the team has started to use online communication (slack) more often.

Bibliography

[1] IEEE, "IEEE Standard for Configuration Management in Systems and Software Engineering ," 20 October 1998. [Online]. Available: http://ieeexplore.ieee.org/document/6197683/ [Accessed 18 February 2017].

[2] Gupta, U. G. and R. E. Clarke (1996). "Theory and Applications of the Delphi Technique: A bibliography (1975-1994)." Technological Forecasting and Social Change 53: 185-211.

[3] SEPR, “Agile team roles,” [Online]. Available: https://seprated.github.io/agile.html. [Accessed: 19 February 2017]

[4] SEPR, “Assessment 4,” [Online]. Available: https://seprated.github.io/assessment4.html. [Accessed: 19 February 2017]

[5] Gluch, D. P., "A Construct for Describing Software Development Risks," Software Engineering Institute, Pittsburgh, PA CMU/SEI-94-TR-14.

[6] SEPR, “GUI3,” [Online]. Available: https://seprated.github.io/Assessment3/GUI3.pdf. [Accessed: 20 February 2017]

[7] SEPR, “Test3” [Online]. Available: http://seprated.github.io/Assessment3/Test3.pdf. [Accessed: 20 February 2017]

[7] SEPR, “Plan3” [Online]. Available: http://seprated.github.io/Assessment3/Plan3.pdf. [Accessed: 20 February 2017]