Formal Approaches to Change Management

In order to optimise the time spent on each change, we first chose a group who used the same Java engine as us for creating their game (LibGDX). This meant that we were not only comfortable with implementing any changes needed, but confident in doing so. We did this in order to speed up the change implementation process as the language was already familiar with all programmers in our group.

In our very first meeting of Assessment 3, we first decided to carefully scrutinise the requirements system for a completed game and outline all the changes necessary to have a finished game. From this initial analysis, we were able to create a list of implementation changes in order to track progress and ensure no requirement was left unfulfilled. Please find this table of changes on the group website[1].

Once we had a list of criteria to be implemented, we began to consider the need for updates to existing code and possibly elements of the game that were unnecessary and could be cut, in order to create a more fluid and coherent gameplay experience. During this process, we found changes that needed to be made to the screens in order to keep memory usage minimal. We then ran the game, checking how much memory was used at each point in order to determine what changes needed to be made behind the scenes for a smoother game.

At this point we now had two lists of changes to work with; a list containing features to be implemented in order for the game to be considered ‘complete’, and a list of changes to be made to the existing code in order to help improve the way the game runs. Now we could begin implementing changes.

Due to the fact that we are working in a smaller group than others on this project, we found it easiest to keep track and manage change documentation informally via a simple change log on Google Drive. This helped us negate the extra steps of having to use a much more complex third party software when it was possible to keep track of only 2 other programmers changes through a simple text document. To supplement this abstraction, we also opted to use Facebook’s messenger service in order to keep one another updated on changes being made and requests for changes to be made. This greatly reduced the time between requesting, performing and then documenting changes. Google Drive was ideal for our method of progressive updating after each change and enabled the entire group to review changes made and collaborate on them.

Throughout our entire changes process, members of our group regularly met with those from Pi-rates to ensure the current changes made and planned were still in following with their visions and where they intended to take their game. Their subjective opinions also helped us keep our eyes on the overall picture of the game and how it was forming, ensuring our changes were not getting unnecessary or ‘over the top’.

Testing Report

Please find referenced our new Testing Method Selection and Planning document [2]. To create this, we took the Pi-rates' exact documentation for their assessment 2 and added new highlighted changes to their method selection. We also crossed out any methods we did not agree with and always left reasoning as to why this was not used, both in the document and below here in the relevant section.

After reviewing the Pi-rates Testing Methods and Approaches, we actually decided to change from our previous testing methods to theirs. Their use of Requirements Testing was something we had not greatly researched into before and feel that we missed out in using it for our previous assessment.

In terms of JUnit testing, we did not need to stray far from their methods and used many of their JUnit tests in our own documentation. This is down to the fact that we were required to alter as little as possible aspects of the game that already worked, and changes were only applied where necessary. Therefore we decided to use the Pi-rates' JUnit tests whenever possible.

Many of the Pi-rates' requirements tests were left unfulfilled in the last assessment as not all functions needed to be implemented into the game. These have since been amended in our own edit of their Assessment 3 Requirements testing document [4].

Methods and Plans Report

Please find referenced our new Method Selection and Planning document [3]. To create this, we took the Pi-rates' exact documentation for their assessment 2 and added new highlighted changes to their method selection. We also crossed out any methods we did not agree with and always left reasoning as to why this was not used, both in the document and below here in the relevant section.

We did not have access to the Pi-rates' assessment 4 plan, it was no longer available as one of the links on their website, and when confronted, they did not have another copy. This meant we could not link to their assessment 4 plan and change it, so we continued to make changes to their overall methods plan.

Our Tools

Our group decided to continue using the same tools as we have done for previous assessments, however these differed from the group whose project we took on. For instance, The Pi-rates used Discord in order to hold meetings when they were unable to meet in person. We did not find this necessary as we had pre-established dates and times to meet weekly so there was never a point in which we couldn't meet face to face. Asana was another tool we decided not to use, as we were already familiar with ProjectLibre we decided to stick with what we knew and not change our software for creating Gantt style timelines and task dependencies. A similar problem arose with the creation of UML diagrams and flowcharts. The Pi-rates group used Lucidchart to create these however as we were already comfortable with the free and intuitive STAR UML, we therefore stuck with what we knew and created diagrams with that. We continued to use Photoshop as did the other group, however some members also used GIMP, a free alternative photo editing software.

Our Team Organisation

Since assessment one, our group has always worked on a flat team structure whereby everyone in the group is equal. This works well as we our a small group and it's easy to keep everyone informed on changes and decisions towards the game. However, where we differ from the Pi-rates is that we do have a tall structure in place, just in case there are any major disagreements we find that we cannot work through. In circumstances such as these, final decisions rest on Alex to make as he was voted in as the overall team leader. This helps decision making concise and avoid elongation of small discrepancies in the group.

Our Software Engineering Methods

The methodologies behind our software engineering barely differ from the group whose project we inherited. This is most likely due to the fact that both of the groups are of similar size (having lost members) and intend on spending the same amount of time weekly on our projects, we both also share the same ideology that the working environment should be 'close and casual' with the client (as mentioned in their method selection and planning document). This is a large amount of the reason why we both chose to adopt the Agile methodology.

References

[1] Rear Admirals Software Engineering Project Site, 2019. [Online]. Available: https://therandomnessguy.github.io/SEPR/Assessment/3/Original_Change_List.pdf [Accessed: 17- Feb- 2019]
[2] Rear Admirals Software Engineering Project Site, 2019. [Online]. Available: https://therandomnessguy.github.io/SEPR/Assessment/3/Edited_Testing_Methods.pdf [Accessed: 17- Feb- 2019]
[3] Rear Admirals Software Engineering Project Site, 2019. [Online]. Available: https://therandomnessguy.github.io/SEPR/Assessment/3/Edited_Method.pdf [Accessed: 17- Feb- 2019]
[4] Rear Admirals Software Engineering Project Site, 2019. [Online]. Available: https://therandomnessguy.github.io/SEPR/Assessment/3/Requirements_Testing.pdf [Accessed: 17- Feb- 2019]