Project Review Report

Team Management and Structure

Initially, we split the work down into individual tasks, so each person was responsible for making sure their section was complete. However, we found that working individually caused communication problems, and the poor attendance of some individuals made it difficult for us to schedule meetings with the entire team present. This communication breakdown made it hard for certain tasks to be completed or even started if knowledge of previous sections was needed.

Although it was on our initial risk register, this risk occurred more than what we had accounted for and had a more detrimental effect than first thought. When it became a problem, we had a review meeting to review the mitigation and prevention strategies. In this meeting, we decided to split the whole team into two groups (each group consisting of 3 people). These two teams were identified as 'Development Sub-Team' and 'Design and Documentation Sub-Team'. Team members were assigned to sub-teams by their preference and expertise.

Since there were fewer people in each sub-team, it was easier to find suitable times for meetings. However, to make sure everyone was kept up to date, we decided to have a minimum of one meeting with the entire group a week to share information in a scrum [1]. This was held during the assigned practical slots in our timetable, to ensure everyone would be available. The sub-teams were only based on preference so the group was very open to assisting in other sections of the project if help was needed or if they had a specific interest in it and as certain sections of the project involved coding and documentation, e.g. Testing -- members of the team were involved in many different sections in the end; the structure of the team was actually quite fluid. This was deemed to be okay as people work better when working on tasks they enjoy [2].

At the start of the assessment we assumed a team leader wouldn't be necessary given that everyone cooperated, but we quickly realised that more guidance and structure was needed to make sure tasks were completed on time as the project continued. When the sub-teams were formed, Tobias and Connor stepped up and volunteered to take on leadership roles over the 'Development' and 'Design and Documentation' sub-teams respectively. Although no official voting took place, everyone agreed that they were happy with this arrangement. The sub-team leaders' tasks involved coordinating and assigning work to members of their sub-teams and advising and assisting in the work as following the standard responsibilities of a leader [2].

Another common issue that we faced was that work was not getting completed on time. This was for a variety of reasons -- poor communication, laziness, or even simply work not being marked as 'complete' on the shared Google Drive spreadsheet.

A risk we hadn't even identified also occurred in that sometimes work was being done by one person, who made significant changes to the software without consent from the rest of the group. This caused disagreements and setbacks. To solve these problems we decided to start working in pairings. This reduced the chances of people going off and completing work without agreement from the rest of the team, as there were now requirements that they must at least confirm with their partner before making significant changes (we did ask all team members to use our Facebook group chat to ask the entire group before making changes, but this did not always happen). This was an attempt at using some of the principles of pair programming across all sections of the project, not just the programming [3] in that the pair would both be working in the same place but on different sections and could discuss their sections with each other.

Methods and Tools

We used a variety of different tools for communication purposes, depending on what section of the project it was for. Facebook Messenger [4] was used by the entire team to be able to discuss issues and tasks that needed completing. We also used it to arrange meetings, as it is used regularly by everyone in the group. The feature of showing who had seen messages was particularly useful for this as we could see if people were paying attention to the chat or awake in the mornings when we were supposed to meet. We also used Slack [5] for the 'Development' sub-team to discuss their work so that we could set up notifications for push and pull requests on our Github [6] repository. This was useful as the separation of the sub-team discussions allowed for the chat to not become messy because of multiple different conversations taking place at the same time.

We used Google Drive [7] to store our documentation and any other files that all team members needed access to, e.g. character sprites. This was useful as it allows for concurrent work online so team members could work on the same document at the same time even when we were at home for the holidays. This also meant we could all access the files if any of us needed them without having to ask for permission, so we could work when we felt like it.

We used Google Docs [8] and Google Sheets [9] to make the documentation as when used with Google Drive they allow for commenting to suggest changes to teammates and they also track changes and when they occurred so we could see who had made changes and when for reviewing each other's work. They also allowed us to revert to previous versions if we were unhappy with our changes.

We used a Github Repository to store the files relevant to the game itself. This also uses online storage so we could all access the files at any time. Another feature we ended up using was push requests as we made it so that if someone wanted to push a change they had to have a confirmation from another team member. This was useful as we had issues with people changing the code without the team knowing and so that people didn't accidentally update the code after the submission deadline which would have lost us marks. As well as this, the branches were useful as we could have team members working on different parts of the code at the same time without having to constantly cause problems for the other team members due to bugs, conflicts etc. Because we were using Github we also decided to use Zenhub [10] partway through the project. This was more useful for the coding team than the to-do list we had on Google Drive as they had fewer windows to open and Zenhub is also specifically designed for the purpose of keeping track of coding so had more useful tools. These included the 'Issue Board' which allowed us to detail each task we had to do and show their progress easily as well as which tasks were linked to each other.

We used an Agile [11] approach for the entire assessment using Scrums [12]. We thought that since we had timetabled meeting times each week, we could use these as the sprint lengths for the Scrum process. It provided us with a guide to use this meeting time the most effectively by performing a scrum each time and using the remainder of the session to work while discussing our work. At first we decided to meet a second time each week for an additional scrum, but due to poor attendance, we instead decided to revert to a compulsory one meeting a week. Notes from each scrum were written in a word document on the shared drive for people to review if they wanted.

Another method we used was Extreme Programming (XP) [13]. Since did lots of pair work, the principles of very short iterations from XP were used more as well as using pair programming. We found this useful, as our work progress fluctuated greatly and we were reaching the point where the deadline was dangerously close frequently. Working in pairs helped to even out the workload.

For the entire project, we used the Rational Unified Process (RUP) [14]. RUP also promotes iterative development which was, like Scrum, suitable for our schedule. The main benefits we saw in using RUP were in requirements traceability and elicitation and in the documentation. This is because RUP promotes the tracing of requirements through the rest of the project and utilising use cases and prototypes. We saw no issues with this so we continued using it as we were for the entirety of the project.

Bibliography

[1] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*, 1st ed. Pearson, 2001.

[2] "Software Engineering Teams", University of Kansas, 2004.

[3] "Pair Programming", *Cs.utah.edu*, 2017. [Online]. Available: http://www.cs.utah.edu/~germain/PPS/Topics/pair_programming.html [Accessed: 27- Feb- 2017].

[4] "Messenger", *Facebook*, 2017. [Online]. Available: https://en-gb.messenger.com/ [Accessed: 05- Apr- 2017].

[5] "Slack: Where work happens", *Slack*, 2017. [Online]. Available: https://slack.com/ [Accessed: 05- Apr- 2017].

[6] "Build software better, together", *GitHub*, 2017. [Online]. Available: https://github.com/ [Accessed: 05- Apr- 2017].

[7] *Drive.google.com*, 2016. [Online]. Available: http://drive.google.com/ . [Accessed: 8- Nov- 2016].

[8] "Google Docs - create and edit documents online, for free.", *Google.co.uk*, 2017. [Online]. Available: https://www.google.co.uk/docs/about/ [Accessed: 05- Apr- 2017].

[9] "Google Sheets - create and edit spreadsheets online, for free.", *Google.co.uk*, 2017. [Online]. Available: https://www.google.co.uk/sheets/about/ [Accessed: 05- Apr- 2017].

[10] "ZenHub - Agile GitHub Project Management", *ZenHub*, 2017. [Online]. Available: https://www.zenhub.com/ [Accessed: 05- Apr- 2017].

[11] "Principles behind the Agile Manifesto", *Agilemanifesto.org*, 2017. [Online]. Available: http://agilemanifesto.org/principles.html [Accessed: 30- Oct- 2016].

[12] “Learn About Scrum", *Scrum Alliance*, 2016. [Online]. Available: https://www.scrumalliance.org/why-scrum [Accessed: 30- Oct- 2016].

[13] "Extreme Programming", *Cs.usfca.edu*, 2016. [Online]. Available: http://www.cs.usfca.edu/~parrt/course/601/lectures/xp.html . [Accessed: 30- Oct- 2016].

[14] S. Jacobson, “The Rational Objectory Process - A UML-based Software Engineering Process”, *Rational Software Scandinavia AB,* 2002. [Online]. Available: http://www.iscn.at/select_newspaper/object/rational.html [Accessed: 9- Nov- 2016].