Project Review Report

Team Management and Structure

The team for most of the project had an implementation and documentation split. This meant that whilst the implementation was taking place, the documentation team would write up any appropriate documents for the current state of the implementation. Once implementation was complete, the remaining documentation was split accordingly.

This remained consistent for both Assessment 2 and Assessment 3, however, during Assessment 4, everyone completed some implementation and then proceeded to complete the documentation.

The team started with a named leader. However, this idea quickly became unnecessary as the natural skills of each team member became clear. Therefore, rather than having a leader that would essentially run the group, the group collectively organised meetings and the skills of the team members helped assign the correct work. As a result, many aspects of the project could be taken up by the team member(s) that felt they could apply the greatest skill (either in their own opinion or in the team's opinion), which often lead to work not being assigned but rather taken on. Even though the group did not necessarily delegate tasks, the role of 'Co-ordinator' [1] was still required to cover the background work, such as keeping everything up to date, organising the communal group work and managing the to-do lists, this role was shared by two team members.

From Assessment 2, the group began to use very detailed to-do lists to keep track of the work required. These lists included a breakdown of the implementation and documentation, meaning that everyone could see what work they needed to complete and could then 'tick it off' once they had finished it.

During Assessment 1, the issues with time management were not recognised because the team met every week, which meant that the work could be easily traced. Although the work may have been left until closer to the deadline, it still felt manageable. For Assessment 2, there were greater time management issues. The implementation was completed before the Christmas break began, which put us in good stead for completing the rest of the work on time. However, once the holidays started, the work drew to a standstill. After returning for the second term and having to balance it with exam week, all of the work seemed more rushed. To prevent this happening again, the idea of early deadlines was established. As a group, we decided when we wanted the work finished before the official deadline. For Assessment 3, this deadline was 4 days before and for Assessment 4, the deadline was set at 3 weeks before the official deadline.

As the project went on, the risks of the project itself remained fairly similar, however, the risks about the group and how the group worked did change. The risk of work not being completed on time was the greatest issue facing the group, and as a result, we established early deadlines. Additionally, for Assessment 3, a client meeting was organised to encourage the group members to adhere to these deadlines.

Software Engineering Development Methods and Tools

At the start of the project we thought it would be important to establish some of the tools we would use to ease the process of software development. These tools were fundamental to communication, as well as the initial coding stages.

The first tools we decided to use were Facebook Messenger, Google Drive, GitHub and Gantt charts. These were chosen because all four tools excel at the task they are designed for, and we also felt it was unnecessary to go overboard with the methods of communicating between each other, because if many tools were used, the majority would simply go unused. Over the course of the project we relied mainly on these communication tools, although email was occasionally used for communication with the client and other groups. They provided us with enough utility to get the job done in the vast majority of cases.

However, when it came to drawing UML diagrams, it was decided that the previous tool (ArgoUML), which had been used for Assessment 2, should be replaced with draw.io. This was done because it made the UML diagram easier to be shared, which was increasingly important when the work on software became distributed across a greater number of the team members.

Otherwise nobody in the team had any issues with the tools we chose originally. During Assessment 4, members of the team who had previously only focussed on documentation were easily taught how to use tools like GitHub by the rest of the team when the whole team was focussing on coding.

When it came to software engineering we decided to use a Scrum-like method. This is because the idea of sprints seemed to fit the four, short-period assignments very well. In addition to this, the Scrum methodology deals with changing requirements with small teams efficiently [2] because of the low documentation overhead and short development cycles.

Initially our team had been working with a loose interpretation of the Scrum methodology, as we were unclear as to the roles everyone in the team should take on. However, as the work continued and everyone's strengths became more apparent, we adopted a Scrum-like methodology in which there was a leader who oversaw assignments, as well as members of the team who were more suited to coding and members who were more suited to documentation.

Additionally, if a member of the team was beginning to fall behind, this would be discussed in a meeting or over Messenger, and the team as a whole would either help the member with their problem (in a meeting) or decide whether a single member who was further ahead could help them with the problem.

Also, another risk that became more apparent in the code development stage, was when some of the members who had not previously worked on the code began to do so. To deal with this, they were given smaller parts to code and the rest of the code was described to them so that it could be written about in the documentation without uncertainty. Members of the team were allocated in the risk section to attend to this.

References

[1] Belbin Associates. *Belbin Team Roles*. [Online]. Available: http://www.belbin.com/about/belbin-team-roles/ [Accessed: *April*. *10*, 2018].

[2] L. Rising and N. Janoff, "The Scrum software development process for small teams", *IEEE Software*, vol. 17, no. 4, pp. 26-32, 2000.