Team Structure and Management

Team Structure

During the course of this project, our team structure evolved in order to meet the changing needs of the project. At the start of the project, we decided to allocate certain roles to team members in order to divide up the responsibility of the project. We had a project manager, secretary and technical lead. These initial roles were allocated based on a discussion of our own perceived strengths and what was required for assessment one. For assessment one these roles worked well as the workload was fairly light and the tasks were quite similar to one another. However, at the start of assessment two we realised some team members were lacking critical knowledge for the project (coding skills), this is a risk we had initially thought about and added to the risk register (#10). This lead to us allocating a documentation lead who would focus solely on the deliverables and a graphics designer. By allocating team roles it helped to divide up the workload as each member in the team was able to take responsibility for a part of the project and oversee its' progress. Furthermore, it also helped in answering inquires for examples, if someone had a question regarding the requirements they would directly contact the documentation lead rather than distracting the whole team. The roles were allocated on a strengths basis because we found if someone is doing something they are good at, they will enjoy doing it and complete the work to a higher quality. During assessments two, three and four the team roles remained unchanged (project manager, developer, technical lead, documentation lead, secretary and graphics designer) as everyone was happy with their role and completing the work to a high standard while meeting the deadline. This was helpful as we did not have to spend time re-allocating team roles but instead were able to divide up the tasks into subtasks and get started as soon as possible.

Team Management

We found our team meetings from assessment two onwards to be a success as all team members were able to get work done in the meetings, discuss design decisions and help out one another. Our meetings had a time-limit of 15 minutes and stuck to the three question format [1]. This enabled everyone in the team to get a understanding of what work has been done and what work is left to be done. During assessment one, our meetings were less successful as they were too time-consuming (we did not impose a time limit) and not very scrum-like (topics were discussed in great detail). However, one failure which related to team meetings throughout the project was punctuality and we had to send multiple message reminders to team members who were late.

During assessment 3, we realised that some team members were only working on SEPR during the meetings, not at home. We felt this was not enough time as during this assessment we had a tight deadline with an increased workload. Once we realised this was happening, we increased the number of face to face meetings to two a week instead of one. This was useful as during assessment three there were more design discussions required while coding as we had to implement all the requirements. In addition, if people had not made progress between meetings and were not able to justify why, the team encouraged them to do better by the next meeting. We felt it was important to address these issues as soon as occurred, before they escalated to uneven workload across the team, risk #13.

Throughout the project, we used a Gantt chart to split the project into more manageable sub-tasks which was generally useful however, for assessment two the majority of our coding was done very close to the deadline which put a lot of pressure on the team. This occurred because not all team members worked hard towards the start of assessment three, and we regard this as our biggest failure during the project due to the stress it put on all the team members.

During assessment one we relied on meeting minutes[2] as a useful way to keep track of everyone's contributions to the project as we had not yet started coding. However, from assessment two onwards instead of meeting minutes we used the 'issues' feature on GitHub [3] to keep track of everyone's contributions. This was a useful approach as we could now assign members to certain issues and assign task priorities. However, during assessment two while we were still new to GitHub issues, there were a few occasions where team members forgot to assign themselves to the issue they were working on. This was a problem as it sometimes led to two people working on the same task without the other knowing but this was improved from assessment three onwards when everyone was more used to using the feature and keeping it updated.

At the start of the project we all decided to make an account on Slack [4] as we had decided to use this as our method of communication outside of meetings to update members on our progress, ask queries, make design decisions or organise meetings. However, as most of the team had not used Slack before, it did take some time to get used to it and make sure everyone regularly checked their Slack. This meant we used Facebook messenger [5] for urgent messages (i.e. member unable to attend meeting). It was not until assessment three when all team members got into the habit of checking Slack regularly for notifications. In general, communication regarding design decisions and organising meetings was good however, there were certain times when a team member was unable to attend a meeting and did not inform the team in advance. Luckily, during this project there were no major disagreements or falling out within the team as we all realised the importance of good teamwork necessary in completing this project as highlighted by many studies[6].

Development Methods and Tools

Development Tools

Tool	When Tool was Used	How Tool was Used	Reason for using Tool
Google Drive [7]	Throughout Project	Organise all deliverables. Comment on each other's documents with suggested improvements.	Allows more than one member to work on the same document at the same time. Used throughout project as files are updated in real-time and there was no better alternative.
Visual Studio [8]	Assessment 2 and 3 - Only used when we were coding in C#	IDE for coding in C#	Allows GitHub integration and was already installed on department PCs
Unity [9]	Assessment 2 and 3	Game Engine	Has a considerable amount of documentation and forums. Drag and drop GUI implementation. Intuitive for novice developers.
IntelliJ [10]	Assessment 4	Java IDE	For assessment 4, we chose a project in Java so no longer needed to use Unity. IntelliJ is the most popular Java IDE and is very powerful
JUnit [11]	Assessment 4	Java Testing	Lots of online support via forums and documentation on unit tests.
Atom [12]	Throughout Project	Developing website and keeping it updated	Intuitive design for novice developers. Used throughout project due to good documentation and familiarity with Atom.
Travis [13]	Assessment 4	Continuous integration service with GitHub.	By automatically building and testing requests, it saves us time testing the code ourselves. Notifications for commit pushes and pull requests are easier to track/read than emails are.

Development Methods

Throughout the project, we used the Agile model as we believe it was most suited to the nature of the project due to its' flexibility with changing requirements. We found it useful to have a product backlog to help prioritise tasks. We also found it useful to split the project into 'sprints' which during assessment one lasted one week however, once we started coded in assessment two we realised we needed to have shorter sprints. During assessment 2,3 and 4 the team members who were more familiar with testing adopted a test driven development approach as mentioned in the testing lecture. However, we found that this approach slowed down the progress of team members who had not used TDD so they continued to write their tests after their code. Furthermore, during assessment 3 and 4 we estimated the time to complete the subtasks before starting the assessment. We did this by breaking down the requirements into issues of GitHub and discussing how long we all thought it would take to complete the task with the maximum time being 2 days. If the task was to take longer than this then the requirement was broken down into smaller tasks/issues. During the project, we felt we did well in avoiding 'gold plating' [14]. For each assessment, we only implemented what the client and requirements stated, nothing extra. This helped to save wasting time implementing things the client did not ask for.

Throughout the project, another one of our key strengths was our method of peer review. Towards the end of each assessment, we had set aside a couple days for the team to review all the code/deliverables and suggest any improvements before the submission. Before the initial review process we discussed that we needed to avoid the most common mistakes: criticising the producer not the product, reviewers not prepared and meetings drifting into problem solving [15].One weakness we identified with our development method is that during assessments 2 and 3, team members were not informing others when they needed help. This lead to a significant amount of time being wasted and put pressure on the team to meet the deadline. For assessment 4, we chose a Java game which meant we had to switch to using a new game engine/library, this increased the risk of team members lacking the required technical skills (risk #10). Therefore, the technical lead decided to set aside half an hour in each meeting to provide any necessary help or direct the team member to relevant documentation.

Bibliography

[1] "Daily Scrum Meeting," Mountain Goat Software, [Online]. Available: http://www.mountaingoatsoftware.com/agile/scrum/meetings/daily-scrum . [Accessed 27 February 2017].

[2] "Enhancing teamwork in software projects," November 2011. [Online]. Available: http://repositorio.uchile.cl/tesis/uchile/2011/cf-marques_ms/pdfAmont/cf-marques_ms.pdf . [Accessed 10 March 2017].

[3] GitHub, [Online]. Available: https://github.com/ . [Accessed 15 March 2017].

[4] Slack, [Online]. Available: https://slack.com/ . [Accessed 15 March 2017].

[5] Facebook, [Online]. Available: http://facebook.com/ . [Accessed 15 March 2017].

[6] E. Weimar, A. Nugroho, J. Visser, A. Plaat, M. Goudbeek and A. P. Schouten, "The Influence of Teamwork Quality on Software Team Performance," [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1701/1701.06146.pdf . [Accessed 12 March 2017].

[7] Google Drive, [Online]. Available: https://www.google.com/drive/ . [Accessed 15 March 2017].

[8] Visual Studio, [Online]. Available: https://www.visualstudio.com/ . [Accessed 15 March 2017].

[9] Unity, [Online]. Available: https://unity3d.com/ . [Accessed 20 February 2017].

[10] IntelliJ IDEA, [Online]. Available: https://www.jetbrains.com/idea/ . [Accessed 15 March 2017].

[11] JUnit, [Online]. Available: http://junit.org/junit4/ . [Accessed 15 March 2017].

[12] Atom, [Online]. Available: https://atom.io/ . [Accessed 15 March 2017].

[13] Travis CI, [Online]. Available: https://travis-ci.org/ . [Accessed 15 March 2017].

[14] K. H. Pries and J. M. Quigley, in *Scrum Project Management*, CRC Press, p. 16.

[15] K. Wiegers, "Seven Deadly Sins of Software Reviews," [Online]. Available: http://www.processimpact.com/articles/revu_sins.html . [Accessed 16 November 2016].