Method Selection and Planning

Stakeholders: Richard Paige, University of York Communications Office

Team: Barney Morgan, Cameron Smith, Harry Berge, Jake Phillips, Matthew Wilkie, Rob Weddell

Method selection

Early in the development stages of the project, it was decided that the team would follow an agile method of development [1]. In general, agile methods are deemed ideal for flexible projects taken on by small teams - perfect for the working environment of SEPR. This method choice was justified by the fact its top priority was to 'satisfy the customer through early and continuous delivery of valuable software' [2]. In addition to this, the fourth principle was especially important given the initial vague requirements given by the clients as it encourages: 'business people and developers to work together daily throughout the project' [2]. Lastly, whilst the requirements for this project are relatively stable and not subject to change, the agile method is sustainable through possible changes in requirements later in development. This flexibility is not found in other plan-driven methods and thus ensures that the project is readily adaptable should the requirements unexpectedly change [4].

In particular, the 'Scrum' method of agile development was chosen [3]. This method insists that teams comprise of three distinct roles – the 'Product owner', the 'Scrum master' and the 'Dev-team members' [3]. It also requires teams to attend the four prescribed meetings: The 'Sprint planning meeting' - a meeting which outlines the agenda for the week's sprint; the 'daily scrum' - a meeting for the teams to come together and discuss their day's work; The 'sprint review' - a more formal meeting for the teams to come together and review the work completed during their sprint (week's worth of work); The 'sprint retrospective' - a meeting for the group to reflect on the work completed and suggest possible improvements and tweaks [3].

As a team, it was decided that slightly altering this traditional Scrum model was needed in order to accurately meet the project's needs and constraints. For example, it was unrealistic for all members to attend the daily scrum meeting outlined in the original model due to each individual's busy schedules and commitments. It was decided that, in compromise, the team would meet twice a week: once to plan the week's sprint, and once to review and reflect the work completed during the sprint. It was also decided that the daily scrum was not essential and that it, and any other correspondence, could be effectively done through online messages or video. Lastly, it was unrealistic to assume that our 'product owner' (stakeholder) could be present at every meeting given his commitment to other groups and outside work.

This methodology was beneficial for the team as it enabled each individual (or small group) to solve each problem as they saw fit without requiring the confirmation of other team members. It also encouraged independence and intuitive problem solving through the Scrummaster's effective delegation of required tasks to the dev-team members and ensured that no development time was wasted with insistent questions outside of the original sprint planning meeting. This was especially beneficial to ensuring the development of the project was done in an efficient, timely manner without missing any deadlines.

The Scrum model also ensures that no team member gets excessively overwhelmed by large chunks of work as each task is broken down into many smaller sub-tasks. This contributes to ensuring team morale and work ethic remains high throughout development as no team member should feel as though they have an unfair portion of the workload. If they are unable to complete a specific task, they simply need to choose one which they feel more comfortable completing. In addition to this, morale is kept high through the constant reinforcement of progress by the completion of small tasks making a noticeable impact on the final product.

Tools

The choice of software for development and collaboration was an extremely important decision that, if wrongly chosen, could severely hinder the project. Thus, much research was done into each choice to ensure it was the best fit for the team's needs. The fact that multiple members would be working on each part of the project simultaneously meant that the collaborative features of each chosen tool were of high priority. It was also decided that it was extremely important for the chosen tools to be quick to learn, or for the team members to have prior experience in their use. This would ensure that minimal time would be spent learning how to use specific software and thus maximise the time spent using the tools to complete the project.

- ***File sharing - Google Drive*** **[5]**

Google drive was chosen to ensure that the team could simultaneously work on documents, review others' work and suggest improvements in real-time. In addition to this, the cloud-based system ensures the safety of each file as every document has each version of itself archived. This means that the group's work is kept secure. The choice was solidified by the fact every team member had prior experience with the software

- ***Version control and team management- GitHub*** **[6]**

GitHub was the clear choice for our repository as it offers free, private repositories to students. Notwithstanding this, Github provides a fantastic service and is the clear industry leader. GitHub makes things easy for team members with a varying range of experience with Git by providing native GUI applications for controlling your repository (GitHub Desktop). GitHub also provides a built-in solution for a Kanban board. In addition to a user-friendly interface, it allows you to link Kanban objects to issues and particular code commits - a particularly useful feature for Agile software development. Finally, as our repositories for this project will be of vital importance, it is important that GitHub has redundancy (3 locations) in the event of service outage or data-loss meaning that our code is very safe.

- ***Flowcharts - LucidChart*** **[7]**

LucidCharts was used as the software for creating flowcharts as it has the features required for speedy creation and allows for edits whilst still keeping the prior architecture and formatting (such as keeping the arrows to assigned to blocks). The fact that LucidCharts is a browser-based application ensures that edits can be made remotely by all team members with ease. Finally, the fact that the team was able to export the flowchart as a .PNG was necessary as it ensures the file can be uploaded straight into the documentation.

- ***Project planning - ProjectLibre***

ProjectLibre was chosen as it makes creating effective and visually appealing gantt charts very easy with no prior experience. The application provides a vast range of features specialised to making gantt charts that stand out such as:colour coding and critical path markers. The charts can also be exported directly as a .png image and easily uploaded straight into the documentation.

- ***UML - StarUML*** **[8]**

StarUML is an excellent UML tool that was chosen for its particular ease of use and out-of-box setup. We required a tool that would produce high-quality, professional-grade diagrams that can be used to complement our submission - StarUML meets these criteria and is available on most major platforms.

- ***Communication - Facebook messenger / Google Hangouts***

A method of communication was required from day 1 of development, thus choosing something that the whole team is familiar with was important. Facebook Messenger is available on most platforms which means team members will be able to stay in contact with ease. It also ensures that proposals and required actions can be decided on very quickly which is of key importance when developing software with an Agile method. Furthermore, Messenger has a complete feature set: enabling quick file sharing, voice and video calling. Google hangouts was used over Facebook for meeting with the shareholder in order to maintain a professional relationship. Hangouts is also more fit for purpose, providing a more reliable video feed than alternative web-based video calling.

Team Organisation

Initially, it was decided that it was not necessary to assign roles to individual team members. This ensured that no member felt forced into a role they were not comfortable with and encouraged maximum participation and effort on all fronts of the project. Later in the development cycle, each individuals strengths and weaknesses were discussed fairly and without judgement. From here, natural roles arose for each member as they excelled in specific areas of the project.

Despite this, it was still expected for every member to equally contribute to each area of the project, the roles were only assigned to each member in order to ensure that tasks were completed to the best possible standard. This meant that for example, although the secretary was not the person always taking the meeting notes, they were the member in charge of ensuring they were done. This was highly effective for the project as each member had transferable skills in all aspects of development.

- **Project Manager - Rob Weddell**

As the project manager, Rob was in charge of leading the Scrum sessions and ensuring that all team members were working productively. Rob was chosen for this role as a result of his previous experience in similar software engineering projects and ability to naturally organise the team. The project manager role also takes an active part in the development of the software - producing the core code of the game.

- **Secretary - Jake Phillips**

The main role of the secretary is to keep track of the team's progress and organise the work. This involved taking notes of each meeting and outlying the contents of each week's 'sprint'. The secretary also took charge in ensuring all deliverable assessment documents were complete to a high standard. Jake was chosen for this role given his experience with literature and talent of producing high-quality essays.

- **Client Interface - Matthew Wilkie**

The role of the client interface is to ensure that the developed product aligns accurately to the client's desires. Matthew was chosen for this role based on his friendly and approachable personality which enabled him to comfortably lead the conversation with the clients.

- **Software Architect - Harry Berge**

As the software architect, Harry was in charge of the higher levelled constructs of the project. This involved making important design decisions as to how the fundamentals of the software will be developed. Harry was chosen for this role for his natural ability to make tough design choices and previous experience with software development

- **Web Developer - Cameron Smith**

The web developer has the role of creating the front face of our project in the form of a website. This task required strong web creation skills and the ability to constantly maintain the site throughout the year. Cameron was chosen for this role for his experience developing web pages, and promise to ensure that the web page will accurately represent the work done by the team in order to effectively market to project.

- **Technical Consultant - Barney Morgan**

The role of the technical consultant is to provide technical expertise to the development. This meant that whenever a team member had any queries on their individual projects, Barney was the point of contact. This ensured that the project manager was not overwhelmed. The technical consultant also holds an active role in all other team endeavours. Barney was chosen for this role as he proved highly adaptable to any situation.

Should this organisation prove not to work effectively, a group meeting will be held in which the issues will be discussed and how we can appropriately resolve them. However, as a team, it was decided that this is unlikely to be needed given the highly adaptable nature of each role and the good level of teamwork/sharing of the workload up until the point of which these roles were decided.

Project Planning

As is the case with most large software engineering projects, it was necessary to have an extremely precise plan in order to ensure that the large workload was manageable. Thus, it was decided to follow the industry standard Work Breakdown Structure (WBS) in order to break down large tasks into more manageable smaller ones [9]. By doing this, it enabled the team to create a backlog of small, menial tasks to be completed during each 'sprint' as described within the agile method of development. This was done through the integrated planning application within GitHub. However, whilst GitHub offers sufficient features to manage the team, it lacks the ability to view a concise overview of the project. The ability to view a summary of the project plan is extremely important as it allows the team members to gauge the scope of their tasks, and provide a sense of urgency to get them completed within deadlines. To resolve this issue, it was decided to create Gantt charts within ProjectLibre [10].

Within the Gantt charts, the main tasks are highlighted in bold and assigned a priority (1-6). Each main task was then broken down into relative sub-tasks which were allocated an 'earliest starting date' and a 'latest finishing date'. As the team was following an agile method, it was impossible to set strict, concrete start and end dates, given that the workload was fluid and not completed linearly.

However, the critical path (red flow of tasks) was linearly outlined through the gantt chart which indicates the dependencies of each task (represented by black arrows) and indicates any workflow blocks. This enabled the team to accurately distinguish the highest priority tasks (mainly the ones which were holding up the completion of later work) and thus gave consistent direction and guidance to each members choice of task to complete during each week. For example, within assessment 1 [11] the requirements elicitation was necessary to be completed before most other aspects of the project could begin. Thus, the 'latest finishing date' for the eliciting requirements was set relatively early meaning less overall time dedicated to that aspect. Another example is the 'run tests' portion of the software testing within assessment 2. It is predicted that these tests will be needed to be completed alongside the implementation of the code. Thus, it would be impossible to complete these tests before the code is finished. This means that the critical path of assessment 2 is almost wholly focused on the testing section of the plan.

At the time of writing this document, it was unrealistic to expect solid plans for the method of the later assessments given the fact that they would most likely need to be re-written later in the year when the team has a greater understanding of the proposed tasks. However, a detailed plan for assessment 2 was created which highlights the proposed method for the completion of the immediate tasks.

Assessment 2 [12] -
The duration of the Assessment 2 plan spans 56 days and begins immediately after the submission of assessment 1 (08/11/18 → 13/01/19). It is planned that work will continue into the christmas holidays, yet expected to be taken lightly. (the only task planning to be completed during this time is the completion of the code and implementation report). The critical path for this period is as follows: confirm architecture, create software tests, run tests, write test report and proofreading.

Assessment 3 [13] -
Assessment 3 is planned to span from the 21/01/19 → 17/02/19. The time period for this section is very short, meaning that there is much overlap between tasks. This means that the critical path simply follows the choosing of a project and the implementation section

Assessment 4 [14] -
This assessment spans 53 days (18/02/19 → 12/04/19). Currently, the plan for this assessment is rather vague given that it is so far away. Consequently, the critical path could not be determined, but rather, a rough deadline was set to ensure all tasks were completed simultaneously in accord to the Agile method.

References

[1] Agile method guide book [Online]. Available: http://agilehandbook.com/agile-handbook.pdf [Accessed 22 Oct. 2018]
[2] Agile manifesto main website [Online]. Available: http://agilemanifesto.org/principles.html [Accessed 22 Oct. 2018]
[3] Scrum description [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum [Accessed 22 Oct. 2018]
[4] Agile methods compared to traditional methods [Online]. Available: https://apiumtech.com/blog/agile-project-management-benefits/ [Accessed 22 Oct. 2018]
[5] Reasons for using Google's team drives [Online]. Available: https://spinbackup.com/blog/google-team-drives-benefits-enterprise/ [Accessed 26 Oct. 2018]
[6] Reasons for using GitHub [Online]. Available: https://dzone.com/articles/5-reasons-why-devs-love-github-and-microsoft-buys [Accessed 26 Oct. 2018]
[7] Reasons for using LucidCharts [Online]. Available: https://www.lucidchart.com/blog/8-reasons-why-lucidchart-is-the-perfect-microsoft-visio-replacement [Accessed 26 Oct. 2018]
[8] ArgoUML compared to StarUML [Online]. Available: https://www.predictiveanalyticstoday.com/compare/argouml-vs-staruml/ [Accessed 26 Oct. 2018]
[9] Work breakdown structure [Online]. Available: https://en.wikipedia.org/wiki/Work_breakdown_structure [Accessed 30 Oct. 2018]
[10] Gantt charts [Online]. Available: https://www.projectmanager.com/gantt-chart [Accessed 30 Oct. 2018]
[11] Element of SEPRise!, Assessment 1 Mockup Gantt Chart [Online]. Available: https://sepr4.github.io/web/submission/assessment1/gantt/Mockup-Gantt-Ass1.png [Accessed 1 Nov. 2018]
[12] Element of SEPRise!, Assessment 2 Gantt Chart [Online]. Available: https://sepr4.github.io/web/submission/assessment1/gantt/Gantt-Ass2.png [Accessed 1 Nov. 2018]
[13] Element of SEPRise!, Assessment 3 Gantt Chart [Online]. Available: https://sepr4.github.io/web/submission/assessment1/gantt/Gantt-Ass3.png [Accessed 1 Nov. 2018]
[14] Element of SEPRise!, Assessment 4 Gantt Chart [Online]. Available: https://sepr4.github.io/web/submission/assessment1/gantt/Gantt-Ass4.png [Accessed 1 Nov. 2018]