Method Selection and Planning

Module	SEPR
Year	2019/20
Assessment	3
Team	York Fire Marshalls
Members	Samuel Whitehead, Andrew Connor, Alex Dawson, Finn Jackson, Fred Dodd, Phuong Kha
Deliverable	Method Selection and Planning

Outline and Justification

There are several methodologies we could consider when planning our project. The first decision we made was whether we should use an agile method or a static method. Each style has different benefits and risks; Agile methodologies prioritise flexibility and communication. Small parts of the project are developed, tested and ensured they work before moving on to the next small part. This allows for a very scrutinised approach to a project,which works better when developing smaller or very specific products. A static method, however, prioritises planning and structure - for example in a method using the waterfall lifecycle, each section (Planning, Development, Testing, etc.) is completed one at a time. This works much better for larger projects and teams as there is a lot more structure to the development process. Due to us being only a smaller team - and working on a project that depends heavily on requirements and testing, we decided an agile method suited us better. For this, we chose to use scrum.

A scrum style methodology splits each week of the project into sprints and relies upon members of the team to be cross-functional (developing each part of their assigned task to fruition) within a limited time frame. This allows for a review of each working part - for example, a piece of code - to be completed, ensuring the program is working to the extent it needs to be. Sprints in our project will be a week-long, with the scrum meeting occurring in our weekly SEPR labs, allowing us to overview and discuss our progress as a group, as well as assign team members new tasks. This meeting also means we can note any risks or places we have fallen behind on and solve smaller issues before they become bigger. Another benefit to us using an agile methodology is that we don't have to change the entire project if our requirements change. We would only have to adjust the specific sprints which cover the changed requirements. In contrast, using a static methodology would mean we have re-plan the entire project. There are risks involved in using the scrum method - which will be covered in more detail in the risk assessment and mitigation section.

Despite all the benefits of agile, we will be adjusting our method to utilise elements of the waterfall method. Due to our project being split into four assessments, we will be creating a brief overview/plan of each assessment and assigning our sprints to fit this plan, adding a bit more structure to the project. We will also be assigning a team leader - something uncommon in a scrum method. This is so allocation can be done in a frictionless manner and we have someone who can overview any issues that arise, interpersonal or technical.

Changes to plan:
Initially we utilised sprints that were a week long - which fit well with the structure of the course and timetable. With a majority of this assessment being completed over a break, however, we found the sprints were much harder to implement. Coding also took much longer than initially thought to complete and implement, meaning our timeline for assessment 2 was reevaluated and changed so that we could still complete all necessary requirements within the given timeframe. Mainly, these changes included adding in consideration for graphics creation and implementation, alongside an increase of allotted weeks for development of the feature side of the coding. We also replanned weeks to

include documentation updates in more depth, as we felt the plan we initially had was not comprehensive enough to cover assessment 2. These changes are also available to view on the Gantt chart.

Tools:

GitHub [1]**:** (Collaboration/Development):

We have chosen to use github pages to manage the website that we are submitting the deliverables for. The code for the website is written in markdown and is pushed to a repository. Since we are using git, the team is familiar with adding changes to the website. We tried to accomplish the same process with Azure but it was not possible to do so.

Google Drive/Docs [2]**:** (Collaboration):

Using Google Docs to create the documentation for our project allows us to work simultaneously on the same document and see updates as it is being written. Since Google Docs stores documents online we can access them from anywhere with an internet connection, meaning that team members can work on the documentation from anywhere and at any time. Making use of this collaboration tool should encourage teamwork while simplifying the documentation process.

Messenger [3]: (Communication):

Another collaboration tool we will be using is Messenger. We chose to use this for communication as all of us already make use of it, and there is no need to complicate this by using another messaging tool. This tool allows us to communicate via text, voice, or video meaning that we can communicate easily and appropriately depending on the situation.

IntelliJ [4]**:** (Development):

The IDE we have chosen to code our project in - supports Java and the LibGDX engine we will be using in our project and has a relatively simple learning curve.

Tiled [5] (Updated):

Tiled is the tool of choice to create a map for the game. LibGDX has good integration with Tiled. We can do interesting things with tiled such as make the code load game objects based on which sprites are on a specific layer on the map. This allows the development speed to be increased as the location of towers do not have to be specified in the code but they can be loaded on the map. An alternative to this would be created a Tiled Map in a program such as photoshop but it is much easier to deal with programs that specialise in vector graphics opposed to raster graphics.

LibGDX [6] (Development):

We will be using the game development framework libGDX to create our game. This is a well known and highly recommended framework to use when creating games in Java. Using this framework to create our game should cut down on development times as we can make use of premade assets, it should also mean that our game runs well as functions within the framework have already been optimised. LibGDX provides cross-platform functionality, meaning that we can create one version of the game and it should work on both desktop (Windows, Linus, Mac OS X) and mobile devices (Android, iOS etc...). This would save much of the development time as only one copy needs to be made.

Piskel [7]**:** (Updated)

Used for graphics creation, Piskel allowed us to make simple and effective sprites and textures for the game and export in a usable file.

Team Approach

As discussed previously we intended to use the Kanban development methodology. Our immediate action then was to define a team leader. Their role would be to initially assign people to tasks and the general organisation of the kanban board. Although all members can assign themselves to tasks having a team leader allows us to better coordinate our efforts and ensure that all members are working towards the same goal the same way. This is needed on a software development project due to the high risk of delays impacting various peoples work. Following on from this having a team leader allows someone to coordinate peoples current level of work completion alongside the status in kanban. Providing input and if needed, assistance through their own help or through the team.

We use communication tools such as Messenger in addition to meeting up to provide ways of talking to each as and when needed. The opportunities for physically meeting up provide a better experience for the team and provide a way of looking at and assisting each others work in an easier manner. By using an online method such as Messenger you can project ideas both to the entire group and specific people through private messages. Messenger in particular allows for voice communication providing methods for the group to talk and interact at any time. If any of the group members are having issues or need assistance private messages through messenger allow this without publicising it if/when appropriate.

The team as work progresses will be split into more specific groups for particular tasks. This offers a level of versatility and as different group members will have varying strengths in different areas we can move people if needed.

To choose which team member would be best fitted for each role, during the first meeting, we all took the Belbin test to see what type of personality that person tends to be within a team structure. The roles will be based on the strength and weakness of each member, keeping in mind the goals of our team. However we also agreed these roles may change throughout the project if we feel it will help our team’s efficiency.

Based on the results and our previous experience working together, we eventually decided the roles would be divided as follows:

Meeting chairs:	Finn Jackson, Alex Dawson
Secretary:	Daisy Kha
Librarian:	Fred Dodd
Report editors:	Andrew Connor, Sam Whitehead

Taking these and the into account when we assign tasks means that overall we are playing to our strengths and will produce an overall stronger project. Allowing someone to work to their strengths also increases their enjoyment and work ethic towards the project. Providing a much better environment for the team as a whole. Before we assigned roles we confirmed with each person that they were happy to work loosely around their role. As a team we were also confident that the selected roles across the team cover the majority of what we would need.

Future Plan

This section outlines the future plans for the SEPR projects. The key tasks from assessment 2-4 will be described and accompanied with a critical path. The tasks are ordered in their priority with the dates they should be completed. The tasks underlined are part of the critical path. This is an example of target dates but we will be following the gantt chart for flexibility **[Assessment 3/Supplementary/Gantt chart.png]**

Assessment 3 (21/01/2020-14/02/2020) (Complete)

Key Tasks:

1. Select another teams implementation **(21/01/2020-27/01/2020)**
2. Formalise Change report methodology & tactics **(27/01/2020-03/02/2020)**
3. Extend Software to meet brief **(27/01/2020-14/02/2020)**
4. Create user manual and Update Deliverables **(10/02/2020-14/02/2020)**
5. Update Change report **(10/02/2020-14/02/2020)**

Software Engineering Tasks:

1. Overview chosen code, ensuring understanding of updates **(21/01/2020-27/01/2020)**
2. Complete any refactoring necessary **(21/01/2020-27/01/2020)**
3. Implement changes to boilerplate code **(27/01/2020-03/02/2020)**
4. Test and assess changes & justify if keeping **(03/01/2020-07/02/2020)**
5. Implement additional (lifestyle) changes **(07/02/2020-10/02/2020)**

6. Test and assess **(10/02/2020-14/02/2020)**

Assessment 4 (15/02/2020 -25/04/2020)

1. Select another team implementations **(15/02/2020- 24/02/2020)**
2. Extend Software to meet brief **(15/02/202-15/04/2020)**
3. Update Deliverables **(15/04/2020 - 25/04/2020)**
4. Update Change report **(15/04/2020-25/04/2020)**
5. Create Presentation **(15/04/2020 - 29/04/2020)**

References

[1] Github, “Github,” Github, [Online]. Available: https://github.com/.

[2] Google Drive, “Google Drive,” Google Drive, [Online]. Available: https://www.google.com/drive/.

[3] Messenger, “Messenger,” Messenger, [Online]. Available: https://www.messenger.com/.

[4] Jetbrains, “InteliJ,” Jetbrains, [Online]. Available: https://www.jetbrains.com/idea/.

[5] Tiled, “Tiled,” Tiled, [Online]. Available: https://www.mapeditor.org/.

[6] LibGDX, “LibGDX,” LibGDXX, [Online]. Available: https://libgdx.badlogicgames.com/.

[7] Piskel, “Piskel,” Piskel, [Online]. Available: https://www.piskelapp.com/.