# Method Selection and Planning

SEPR Team: Berbils

| | |
|---|---|
| **Module** | SEPR |
| **Year** | 2019/20 |
| **Assessment** | 1 |
| **Team** | Berbils |
| **Members** | Dylan Henley-Marshall<br>Edward Demkowicz-Duffy<br>Emily Wisher<br>Samkeliso Kimbinyi<br>Joshua Waha<br>Peter Robinson |
| **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 for a project which works better when developing smaller projects or projects to fit a very specific brief. A static method, however, prioritises planning and structure - for example in a waterfall methodology 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 means that if our requirements change then we do not have to, as we would in a static methodology, replan our entire project, instead we only have to adjust which specific sprint covered the requirement changed. 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 to ensure we are completing it to our full potential. 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.

**Tools:**

**Azure DevOps**: (Collaboration):

Our primary collaboration tool; DevOps is a development suite that hosts a variety of tools useful in our project planning. This suite allows us to use a Kanban board, allowing our team to individually select tasks to complete based on priority, level of completeness, and time frame. This also allows us to approach tasks in a top-down manner, as tasks can be split into subtasks. As an online tool, this also means we can collaborate in real-time, and link our DevOps repositories to Git (see below)

**GitHub:** (Collaboration/Development):

GitHub's repositories automatically allow us to version control our project so we have backups in case something breaks – and being linked to our DevOps page means we can manage our repositories from one location. Being Open Source, GitHub also allows us to collaborate with team members more easily and upload to individual GitHub accounts.

**Google Drive:** (Collaboration):

As a writing tool, Google drive allows multiple team members to edit documentation at once and organise our files in a team drive. It also has a basic version of version control – useful in case a document is lost or edited incorrectly.

**Discord:** (Communication):

Primarily a communication tool, this offers a way for team members to communicate outside of labs in either text, voice, or v\eo. Discord also allows for a list of important links to be available to each team member at all times (as it is also a mobile supported application).

**Modelio:** (Planning):

Our modelling tool can be used by the team members allocated the task of modelling all of our classes/diagrams. This tool, in particular, is useful as it supports a range of different modelling styles alongside UML, which we will be using to plan our design

**IntelliJ:** (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.

# 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 discord 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 discord you can project ideas both to the entire group and specific people through private messages. Discord 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 discord 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. We assessed our team members strengths in a team environment using the Belbin test, with the allocation as follows:

| | |
|---|---|
| Sam: Meeting Chair/Team leader | (Belbin: Coordinator) |
| Emily:  Report Editor | (Belbin: Plant) |
| Edward: Developer | (Belbin: Implementer) |
| Josh: Librarian | (Belbin: Shaper) |
| Pete: Developer | (Belbin: Team-worker) |
| Dylan: Secretary | (Belbin: Specialist) |

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 **[Images/Assignment1/GanttChart.PDF]**

## Assessment 2 (31/10/2019 - 17/01/2020)

### Key Tasks

1. Create Architecture Report (31/10/2019 - 06/11/2019 )
2. Begin Implementation(06/11/2019-10/01/2020)
3. Create a software testing report(06/11/2019-15/01/2020)
4. Update Assessment 1 Deliverables(06/01/2020 - 17/01/2020)

### Software Engineering Tasks

1. Implement Boilerplate code(06/11/2019- 27/11/2019)
2. Make Assets(Graphics, Sound)(06/11/2019-20/11/2019)
3. Create Executable Jar(17/01/2020)

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

1. Select another teams implementation **(21/01/2020-27/01/2020)**
2. Extend Software to meet brief **(27/01/2020 - 10/02/2020)**
3. Update Deliverables **(10/02/2020-14/02/2020)**
4. Create Change report **(10/02/2020-14/02/2020)**

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

1. Create Presentation **(15/02/2020- 25/04/2020)**
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)**