Method Selection and Planning

Persons involved:
Adrian Reyes Frias
Grisha Hushchyn
James Prenter
Jordan Layton
Patrick Maloney
Benjamin Leo Chandler de Waal

Table of Contents:

Outline and Justification of Software Engineering Methods

We have decided to use an Agile methodology for project management as we think it would work best for a project of this size due to its core values, emphasis of individuals and personal interactions, working software, customer collaboration and responding to change [1]. This is as we think a less structured approach work well for us, as this is the first project we are attempting of this nature and as we are a small group we value the freedom of flexibility instead of having things unnecessarily set in stone at this point. This is compared to other models such as the waterfall model which make change much harder to account for.

As for the specific methodology used, we decided to go with Scrum [2]. Scrum is a team based agile method that relies on the team having no overall leader who decides what happens, but instead has a self-organising team that decides on problems as a whole. Scrum bases its development on a series of sprints which work requirements listed in the product backlog. These are development windows in which developers work out what they will do within the timeframe, creating a sprint backlog which specifies what needs to be done for the sprint, and then take those ideas to completion by the end of the sprint and then reflect on the outcome. We have decided to have sprints of two weeks (though these may vary slightly in terms of time) during development, with (at least) bi-weekly meetings to ensure that everything gets done on time. This is instead of the recommended daily meetings as we have other work to do as well, and bi-weekly still seems frequent enough to get everything necessary done. Scrum is advantageous for a number of reasons. As it prioritises a large number of fast iterations, it makes it easy to get work done efficiently and see what more needs to be done. As well as the fact that because we are relatively inexperienced, should we make mistakes it will be quicker to rectify any problems. It also focuses heavily on collaboration within teams, which appeals to everyone within the team and should complement our aim of non-rigid team structure. Scrum is also good in that it has a small number of team roles. There is the Scrum Master that acts as a coach for the team, and the product owner who represents the customer, understands what needs to be done for the project and manages the backlog. This simplicity reduces confusion, as it means people can do what tasks are necessary rather than having to stick to what are assigned.

Additionally, if for whatever reason something in the project isn't working out, we plan to try and schedule a team meeting for as soon as possible to try and resolve it. This will broadly involve everyone stating their interpretation of the problem, and then trying to work together to resolve it. Our idea is that if we catch issues at an early stage we can try to resolve them without them snowballing into larger problems later on.

For the sprint backlog and product backlog we simply have a google drive document that we are editing, because as of now we don't have any particularly complex tasks and it seems unnecessarily difficult to obtain and check different software on a regular basis. This allows us to easily look up what needs to be done while working on any other documents within the project.

For UML and flowchart production we are using StarUML, as it is a free to use modelling tool with a good range of features. It is also relatively straightforward to use, allowing for easy creation of elements and relationships. The only downside is that the free version does put watermarks on exported images, but since this isn't a commercial project that isn't too problematic. Additional justification for StarUML can be found within the architecture document.

For a website we decided to use GitHub pages as since we are already using it to store code and handle version control, it seems the simplest solution as you can turn a repository

directly into a website. Since GitHub is primarily used by developers it also allows for easy managing of structured documentation and uploading files is simple.

We have decided to use Google Drive to help us store any documents we might need for the project, this is helpful as it is an easy to use a cloud based service which allows for joint ownership of documents and allows editing of documents without the need to download, change and re-upload. It even allows simultaneous editing of documents and on top of this it is free. Furthermore, the fact that it is cloud based means that there is no risk of losing documents and it allows retrieval of old versions should we want to revert to something previous.

For sharing code and collaboration we decided to go with GitHub as it allows for large free repositories to store code, as well as easy version control so we can revert to a previous version if a new version starts to cause problems. Additionally, as it has external cloud based storage our code is much safer than if directly handled by one of us in a more local solution.

In terms of framework we decided to use LibGDX. This is because we are required to use Java for the project, and LibGDX is a fantastic framework for Java based games as it has extensive Java libraries, as well as a large amount of documentation and community support from forums. Additionally there aren't any other Java engines or frameworks which work as well for the sort of two-dimensional games we want to make, other engines such as jMonkeyEngine are more suited for three-dimensional games. Also, since this is our first game, we decided the support granted by such a framework would be much more preferable to starting such a project from complete scratch.

We decided to use Facebook Messenger for text communication. This is as it is very easy and simple to use, can store pictures and send notifications if you have a message. And since everyone already had Facebook accounts it seemed a good choice to use to communicate with.

If we need to discuss something by speaking to each other, we will use Discord since it is a free easy to use service which allows team calls with a number of people. It is an improvement over several competitors, such as Skype, as it is much more reliable in that video calls have increased quality and are less likely to break up.

For making the Gantt chart, Excel was used as it is understood and accessible by the team, making using it easy. Additionally, it is easily customisable and fast to use.

Approach to Team Organisation

We have decided to implement a very loose team structure for the project. This consists of having some roles, such as the Scrum ones filled in, but remaining flexible as to assigning roles and tasks or changing existing ones should the need arise. We chose the initial roles by conferring about what we thought were our personal strengths and weaknesses. Then reflecting upon that we ascertained who seemed to be a good fit for the initial roles. While Scrum projects usually only have two roles, the Scrum Master and the Product Owner, we felt the additional roles would help as individuals have work they specifically and definitely have to do themselves.

We have decided, though we have a Meeting Chair, to have no leader in a definitive sense given the small number of people involved and the fact that this is a university group project rather than a business project which might be structured with a more traditional hierarchy. The advantages of maximising flexibility rather than a conventional structure mean that we can respond quickly to any new job that needs to be done, which is especially important in this case given that none of us have worked on a project like this before so we have little idea as to what we can expect going forward. This means we are open to changing organisation should that be necessary, as we wouldn't want to limit ourselves unnecessarily early on given our general unfamiliarity with the whole process.

Justification of Role Assignment:
Scrum Master/Meeting chair (Jordan): The role of the meeting chair is to make sure meetings are carried out smoothly and that everything that should happen does, and the Scrum Master is not a project manager but helps to facilitate the Scrum process. Jordan has a strong personality and is very good at time management which makes her able to encourage active participation and chase people should they fall down on this. This allows her to use her organisational skills to help in delegating roles and balancing schedules.

Secretary/Product Owner (Ben): The role of the secretary is just to record decisions and attendance within meetings for future reference and the Product Owner manages the product backlog. Being a secretary requires diligence about noting things down and the Product Owner requires cognizance as to what needs to be done for the project overall, both of which Ben is good at because of his high level of organisation.

Librarian (Grisha): The librarian helps to ensure that any documents and useful resources are well kept in an accessible way for all team members and to guide version control. Grisha is a good fit as he has excellent organisation, an ability to keep track of what's where and the willingness to comb through files and ensure that are all managed correctly.

Report Editor (James): The report editor is needed to make sure that reports are produced and that they are error free. James is a good fit as he is exceedingly meticulous when checking documents and has an excellent grasp on English, able to easily and fluently inspect work.

Risk Manager (Patrick): The risk manager is needed to keep in mind potential issues that could occur during development and how to avert or work around them. Patrick is very risk aware and is able to cope with what might transpire, so he is a good fit for this role.

Systematic plan for SEPR project

For software engineering in assessment 2, we intend to have two main groups working on the project. One for the sailing mode and one for the combat mode, since they are mostly separate. Additionally to that we intend to break down each mode into as basic components as we can, while figuring out any reliance's between them. The next stage is to get each individual to work on a subdivided part, slowly bringing divided parts together once they have been completed. Culminating in connecting the two modes fully, after which we'll have to have a meeting regarding what to add or change to the game additionally, as from this point it is impossible to tell. This is a rough idea of what we have so far:

Sailing Mode:
Map system -> distinct map nodes (e.g. Colleges, encounters, unknowns)
-> Quests; -> Misc. (e.g. Currency); -> Main objectives (Winning and losing the game)

Combat Mode:
-> Basic card functionality -> Basic AI -> Winning and Losing -> Misc.
-> Bringing both game modes together

The Gantt chart attached [3] shows the general plan for the rest of the SEPR project identifying dependencies and giving a timespan for each of the key tasks of the projects. The amount of time we gave each part of the project depended on that part's dependencies and how difficult we thought it would be to finish. We intend to stick to this plan as much as possible, though we may make changes later on, should we need to, as the further the assessment is away the less we have a good idea about what to do for it.

Assessment 2:
We thought the updated assessment 1 deliverables wouldn't be very difficult since we already had a basis for them, so they were only given two weeks. Whereas the implementation we expect will be much more difficult. As it is a relative unknown to us, we have assigned that a much longer span of time, with the architecture being updated alongside it and being finished after. These lead into the software testing report and updated website, which haven't been given much time as we have a good idea of what we want to test and the website is just updating an already existing site.

Assessment 3:
For assessment 3 the beginning of the time is taken up with choosing a project. Of the time left, implementation takes up most of it since we will need time to understand and work out what to do with the code given to us. The change report is worked on twice, firstly to reflect our initial vision having received a new project and then updated to match what we have done with the implementation. The updated website again is not given much time, for the reasons given before.

Assessment 4:
Again for assessment 4 the initial time is taken up by choosing a project. After this, most time is taken up by implementation since we are getting used to a new project and implementation is the section of the project which contains the most unknowns and potential difficulties to do with timing. Afterwards the evaluation and project review report get two weeks each at the end, since that should be enough time to think about what we want to write for them.

Bibliography:

[1] K. Beck, *Manifesto for Agile Software Development* (2001) http://agilemanifesto.org/

[2] K. Schwaber and J. Sutherland, *The Scrum Guide* (2016) https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

[3] https://github.com/7SeasOfSomething/All-Hands-On-Deck/blob/master/docs/assessment1/GanttChart1.pdf