4. Method selection and planning

Any changes to the document are written in blue text.

4.1.1 Development method

We are using the Scrum development method. We believe this is a good choice as we can incrementally produce the specified product, with each increment in a releasable state. This means that for each Sprint, we have a working codebase to fall back on. The roles given by Scrum (Product Owner and Scrum Master) provide clear points of contact in the team for all team members and help provide a clear vision on what the game requires and what tasks each team member should be undertaking at any one point.

During Assessment 2 we deviated slightly from the Scrum methodologies as the team was separated over the Christmas period and everyone was available to work at different times, so instead of more concise Sprints, everyone did their work in their own time. The team still worked toward incrementally improving the product in a Scrum-like fashion. This allowed the team to be even more flexible in terms of working time, which was necessary to accommodate the change in circumstance.

4.1.2 Development lifecycle

We will use Scrum alongside an evolutionary approach (using the exploratory development style, as described by Sommerville [3]), as this allows for time to re-evaluate our requirements and promotes involvement with the customer. Our increments produced by Scrum can be analysed to see what features fit with the customer's wishes, and can then be added to with new features. The scrum method of software development recognises that customer requirements, and therefore development processes, are often unpredictable. It is a highly flexible process and is also well-suited to teams of our small size [4].

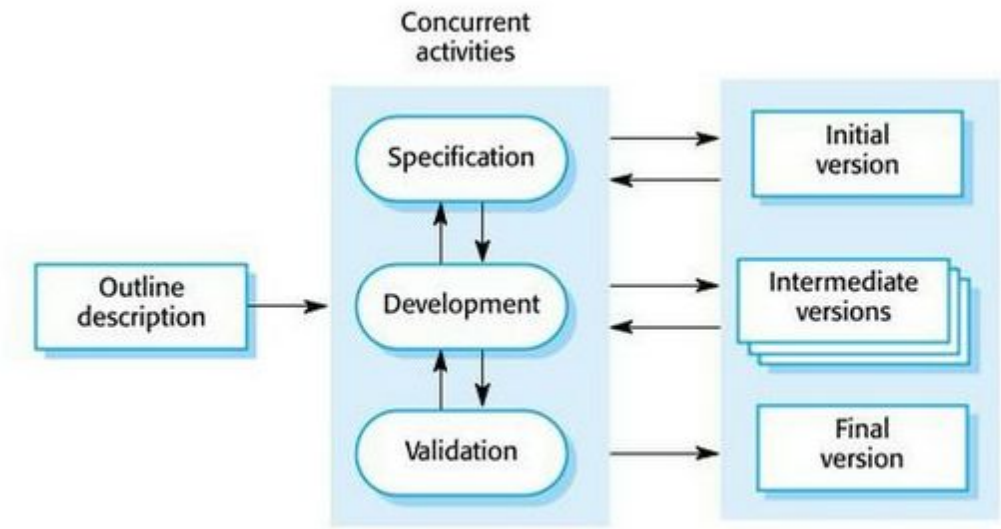

Diagram showing evolutionary approach. [3]

An alternative to the Scrum method is the waterfall model. This model is simple to understand and work with, and each stage produces well-defined goals. Phases such as design and implementation do not overlap, creating a method of working where phases are completed to a high enough standard to support another phase at the end of each stage - while they can also be returned to in the future.

However, we found that this model was not suitable for our team. This model is not as flexible as Scrum and therefore does not allow us enough opportunities to make changes to our requirements or design during the development process. Due to the working software being produced late in the waterfall life

cycle, there is more risk and uncertainty associated with this method. Scrum eliminates much of this risk by transforming software development into a process where small sections of code are completed during sprints. This code is then added to the project and tested so that it is clear that it is working code. Therefore team members can be confident that the code base at any one time is solid.

In addition, scrum's style matches our availability and meeting preferences. We are often available to have fairly short meetings on campus, and our team members can then work in their own time between meetings. This allows a flexible time management program for each person, allowing them to prioritise game development with other work.

4.1.3 Version control

We will be using git as our version control system, with a GitHub repository as our main repository. We chose it as it is fairly simple to use after an initial learning curve, but allows for development along branches, which can be merged with ease. It was also chosen as it is truly distributed, meaning that we always have at least seven copies of our code (one for each team member) which can be accessed without access to the GitHub server. It has built-in version control functionality where previous working code is preserved as changes are made, therefore reducing the risk of rendering previously good code unworkable.

4.1.4 Website

We are using GitHub pages as our web hosting, as it is free and is connected with our GitHub project. There is also a simple editor that can quickly generate a good looking website with minimal effort.

4.1.5 Communication

Our primary tool for communication is through use of the Slack messaging app. This tool allows us to communicate instantaneously with team members. Messages can either be posted to the entire team in one of several channels or privately to an individual team member. As new messages on a different topic can go into a different channel, conversations are easier to hold without discussions being disrupted. All messages are archived and searchable, so we are able to access all previous communication at any time.

Slack became particularly useful during assessment 2, due to the team being split up over the Christmas period. Meetings were held via Slack rather than in person and work was distributed during these meetings, along with progress reports.

4.1.6 Collaboration

We are using Google Drive to host documents that are currently being edited, as this service allows team members to immediately view the latest iteration of the document and post comments for others to consider. All edits to documents happen in real time and are visible on any computer, making it easy to collaborate with teammates.

4.1.7 Development language

The selected language for the game is Java. We chose this because it is a language with multiple well-written game development libraries to choose between, and is well-known to the whole cohort. Other suggestions from the team included Python and C#. However, Python only has one game library of repute, while very few of the team or the SEPR cohort has any experience with C#. As the project swap is of large concern, we decided that it would be best to stick with a language that is well-known to us and the rest of the SEPR teams, and also allows a fairly wide range of choice in development strategy.

Eclipse IDE would be used to code with Java as it is already installed in the university computers and everybody has experience with using it. It also contains useful tools for refactoring code which would speed up development. It also has JUnit built in so the game can be tested easily.

4.1.8 UML software

JsUML2 was used to plan the architecture of the game as it is free online and can be exported to different file formats. It also seemed the easiest to use when tested by various members of the team as it uses a drag and drop UI.

4.2 Team Roles

As we are following the Scrum principles, we have chosen the roles that Scrum dictates. These are:

Scrum Master - The Scrum Master is responsible for ensuring the team can always do the best work they possibly can, and that the principles of Scrum are followed. They help by organising meetings, making sure the backlog is in a workable state by working with the Product Owner, and making sure the team doesn't take on too much (or perhaps too little) work during a sprint. The Scrum Master can change the Scrum process, for example by changing the sprint lengths. If the team deviate from the Scrum principles by, for instance, not finishing a sprint with a potentially shippable result, it is the Scrum Master's job to address this.

Product Owner - The Product Owner is responsible for having an overall vision of what the Scrum is trying to build, and prioritises the product backlog accordingly. However, the Product Owner doesn't have any direct control over the sprints, such as how much work the team does in a sprint or how many sprints are run. This is down to the team, as they know best how much they can do in a sprint, and how best they work. The Product Owner's job is to motivate the team by providing clear goals for them to achieve. Ultimately, the Product Owner decides what features the product will actually have, and shapes the direction that the product takes.

Scrum Team Members - The Team Members are responsible for building the product. Scrum differs from more traditional software engineering in that the team members are not split into different roles such as tester, architect, and programmer, but rather everyone works together to complete the work that was agreed upon for each sprint. This requires all of the team members to have an understanding of the project requirements and actively participate in the development, rather than waiting to be told what to program. Each member may work on a variety of different things within the project to further the goals of each sprint.

We will have one Scrum Master, one Product Owner, and everyone else will be a Scrum Team member.

These are the standard Scrum team roles. However, our team situation is slightly different from many other Scrum teams in the industry. Everyone in the team, including the product owner and scrum master, will be writing code. This is atypical of Scrum. As every member of our team will be involved in the development of the software, our product owner will be involved in helping the scrum master manage sprinting. We feel that this is the correct decision because each member will be directly involved in the sprints, and have their own workload. It would not make sense to arbitrarily separate a member from the coding process.

We have chosen these roles because they are the roles that Scrum dictates. They suit our team well as we are a small team and iteratively adding functionality to a potentially shippable game is the most safe and efficient way to develop it, and allows us to follow the requirements in a systematic way.

4.3 Project plan

An updated version of the Gantt chart reflecting the current plan and progress is provided in figures 2.1 and 2.2 in the appendix that is specific for Assessment 2. The chart was created with the intention to reflect the groups more realistic vision of the project based on how the work went in the first assessment.