UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG1 Group Assessment 2
Team 1

Auber

Plan1.pdf - Method Selection and Planning

[Based on the Assessment 1 deliverable created by Team 4]

Group Members:
Jonathan Davies
Jamie Hewison
Harry Smith
Zee Thompson
Mark Varnaliy

Selecting a Software Engineering Methodology:

Approaching this project as a small team, we knew that first and foremost a choice had to be made as to which methodology we would be adopting throughout the development process. During a discussion in our first meeting, we looked at a couple of different methodologies that are available when developing a piece of software.

Waterfall was the first to get eliminated- whilst it is the easiest to implement and forces us to plan thoroughly before we perform any designing or coding, we knew that concrete plans were unlikely to be kept for very long with this being the first time creating something of this scale in the Java language for all of us. This uncertainty in the early development phase, coupled with the nature of the Waterfall Model which only provides a working version of the system in the later stages of the development would overall prove detrimental to us. The Spiral Model was the next to go, because it required a certain level of prior knowledge beforehand to create rules and protocols which should be strictly adhered to throughout the development. We simply did not have this expertise as relative beginners to Java game development, so we thought we would struggle to make it work.

This led us to the Agile Model, and we agreed that Agile appealed to us the most. This was because the ability to deliver incremental progress on our project would not only allow for constant self-evaluation of how we are performing and where we would like to be, but also provide a close and inclusive relationship with our customer. This usually helps groups who are dealing with volatile requirements, and whilst our requirements may not have been volatile, they were quite broad in some respects. Being able to have frequent meetings with the customer to discern if our product was heading in the right direction would be a huge help in the production stages. Furthermore, the nature of the Agile Model where the project is broken down and divided into its smaller parts would also work in our favour, because this would allow each member of the group to have something to work towards between meetings and be actively involved.

With this in mind, we believed the basic Scrum development best catered to our needs. As previously stated, the project would entail completely brownfield development in an area none of us are particularly familiar in, so therefore our early changes and ideas may not be the best ones in the end. This is mainly down to not immediately knowing how certain parts of our chosen game engine fully function yet, or us potentially making false assumptions based on previous projects- creases which could only be ironed out through gradual research, development, and weekly evaluation with other members in our group. Using scrum provides the flexibility we need to allow these changes to happen whilst also allowing us to keep track of our gradual progress and risks along the way, without having to completely redo prior sections of our program, which is supremely beneficial.

Tools Used in the Development:

In order to begin our development, we also needed to think about some of the tools and programs we may use along the way to aid us, both collaboratively and individually. GitHub was the first program that we all chose to use. We saw this as the foundation on which our project could be built on for a few reasons, but mostly because some members of our group were already familiar with its features and interface. It facilitates group collaboration by creating a code repository for our work which all members of the group could access, add to, or modify at any time, allowing us to keep up with the dynamic nature of the project and its different iterations. This is especially necessary in our current situation due to our group members being in different places, so a cloud platform was not only suitable, but imperative.

GitHub’s suitability for us as a team only grew further when one of our members suggested we use GitHub Pages as the platform to produce the website for our implementation, something which was a requirement in the deliverables. The main benefit of using this GitHub feature over other programs is the seamless updates between the website and our existing Git repository. Our website would be automatically up to date throughout our coding- a crucial factor when the Scrum method which our group selected would lead to very frequent updates to our code.

With a workspace set up, we turned our attention to the need for group planning to take place. We decided to use Google Drive to tackle keeping track of what our group needed to do. At the beginning of the assessment, we created a central To-Do document and clearly listed all changes that needed to be implemented in the game, and what documentation needed to be produced. This allowed group members to see what had been done, what people were working on, and what was still left to be done. Furthermore, within Google Drive, we were able to use Google Sheets to create Gantt charts, allowing us to see our progress developing throughout the assessment.

We returned to Google Drive and instead decided to use it as a platform on which we could host our team’s research and ideas for the project. Because we would be using a game engine none of us had experience with (namely LibGDX) except from Assessment 1, before development could begin we needed to research and familiarise ourselves with the package and its functions to get us started. Documents were made which set standards for our group with regards to how functions should be used, which is helpful when everyone is working separately. Not only that but ideas and assets that we wanted to utilise in our game also needed a place to be stored so that anyone in the group could access them, so Google Drive was suitable because these items could be published for the whole team.

As our IDE to develop the game in, we chose to use IntelliJ IDEA. We decided to use this as it has extensive compatibility with Java, LibGDX and Gradle, which is used to compile the game. It was very easy to install and begin using, and was free to use since we can use a student licence. IntelliJ IDEA also supports the Javadoc commenting system, allowing us to easily comment our code to ensure it is clear what each part of it does. Furthermore, a lot of the documentation for LibGDX is based on IDEA, so it is fairly easy to research help when we’re unsure about something. As an alternative, we considered using the Eclipse IDE. However, this proved harder to set up and understand as new users, so we decided to stick with IDEA. IntelliJ IDEA also supports the outputting of the Javadoc to HTML, making it easy to share this documentation on our website, if we choose to do so.

Finally, we plan on using draw.io in order to complete our UML diagrams for this project. It is much more suitable for this purpose compared to drawing them ourselves in another vector program, because it facilitates the symbols and features needed for these diagrams natively. Not only that, but it is free, and has strong links to Google Drive (a platform our group is already using) which allows for easier real-time collaboration between members – something which will no doubt be necessary along the planning and production of our game.

Team Organisation:

After choosing to use the Scrum software development method, it gave us the principles required to begin organising our team and thinking about how we will proceed with the project. We knew that we would have to divide our work into sprints which we decided to employ weekly, due to our pre-existing Thursday practical slots. This meant that each group member would have something to work towards through the week, which could then be presented to the group and evaluated when we all met in the next scrum during the following Thursday session. Due to Scrum being an Agile methodology, we knew that for these small improvements to occur weekly we would have to break down our requirements into smaller, more manageable tasks. Complex requirements, like the player needing a map to walk on, could be tackled by a group of 2 team members for example – one working on the visual aspect of the map, and one working on how the map is actually represented in our code for the player - and these smaller units within our group could also have scrums of their own in order to ensure the end product when the task is finished is cohesive and can easily be integrated. We realised that this communication between members of the whole team and these subunits is a key part of Scrum and any Agile methodology because without it, everyone working on their own part of the program could lead to integration issues at a later point in the development.

Scrum also gave us the opportunity to involve our customer in the development process more than other methods. We had a meeting available to us once a week, and in our scrum meetings we discussed and evaluated whether we would need any customer input to aid us in the next stage of production. If we agreed that this would be favourable, we would have the meeting with our customer and then immediately go into another scrum to assess the information given and how our plans for the current sprint would change or stay the same. This approach to organisation was suitable for us, because it allowed our group to focus on creating a product sooner in the development cycle which we could then build upon through consistent research and feedback from our customer meetings. It becomes much easier for a customer to understand what exactly it is they want from a product if they have something tangible which could show them features being implemented and progress on a weekly or bi-weekly basis, and this organisation would make the project easier for us.

Systematic Plan:

When devising a roadmap for this project, we settled on certain key tasks which needed to be worked on in order for our project to be a success and to meet the requirements given by our customer. These included “generating requirements”, “finding a game engine”, “creating a game screen”, “generating UML for the project”, “producing players”, “producing operatives”, “producing teleporters”, “testing and bugfixing”, “producing the website”, “deliverables”, and “submission”. As a group we understood that these key tasks would need to be broken down further in order to more effectively progress through the development cycle by splitting the workload.

At this point, it became clear that a Gantt chart could be produced using those tasks so that we could have a clear visualisation of the work ahead, and what time-frames we would expect each part of the project to take. Our Gantt Charts can be found on our website, here: https://eng1-eclipse.github.io/website2/gantt-charts/. When changing team management, we were not provided with an editable copy on Team 4’s Gantt chart, only static images of them, and their former version didn’t include Assessment 2. For this reason, we have created our Gantt charts from scratch.

Some tasks took priority over others, and so a systematic approach needed to be taken, where certain tasks needed to be finished in order for work on the next stages to ensue. In order to represent this, we have our critical path. We chose not to show this explicitly on the Gantt chart so that it can easily be updated later without having to change too much. This path can, however, be identified by the starting dates of each task. For example, tasks which start later can’t be done until at least a proportion of the tasks above it have been completed. This proportion can be identified by the delay in the starting dates - if tasks A and B were both 4 weeks long and B is listed as starting 1 week later than A, then this implies that ¼ of task A must be complete before task B can begin.

Plan Evolution:

Our Gantt chart was updated weekly during the project to reflect any changes in how we were approaching each task. If we became delayed on one part of the project, the Gantt chart would be updated to show this task taking longer than expected, as well as any delays on parts of the project relying on this part. This method evolved somewhat during the course of the first assessment, as we approached the hand in date. At the beginning, the chart was just used as a rough outline of what we expected to do over the course of the term, however, by the end of the term, it evolved into a more rigid plan, showing what we still needed to get done in time for the hand in date.

Our Gantt charts, along with the reasons for adaptation over the project, can be found on our website, here: https://eng1-eclipse.github.io/website2/gantt-charts/.