Method Selection and Planning

Hannah Thompson
Kyla Kirilov
Ben Hayter-Dalgliesh
Matthew Graham
Callum MacDonald
Chak Chiu Tsang
Doaa Doukh
Surbhi Lahoria
Sean Abong
Peter Beck
Isaac Ohara
James Cretney
Lloyd Newton

1.1 Software Engineering Methods

Our team is adopting an agile approach to software development because of its flexible and adaptive nature. An iterative approach allows for regular reassessment of our actions in response to changing requirements and priorities. The Agile method we used was Scrum.

By aligning our use of agile methodology with the timetabled practical sessions provided by the department, we . In order to have completed the deliverables by the six-week deadline, we treated each week as a 'sprint' and held meetings twice a week to discuss the progress and outcomes of that sprint.

In the first week, we held the project's 'kickoff meeting', in which we addressed the briefing, discussed ideas and allocated resources, as detailed below in '2.1 Approach to Team Organisation'. In this meeting, we discussed the sequence of tasks involved in each deliverable. A high-level view of our work breakdown structure can be found on our website under 'Method Planning and Selection, Fig. X', and is elaborated on in section 3.1 of this document.

From here onwards, we will meet in-person on Tuesdays to check-in with the progress of the sprint, and on Fridays to define objectives for the next week's sprint. Tasks from the work breakdown will be assigned to either individuals or sub-teams. A record of these meetings can be found on our website under 'Method Planning and Selection - Meeting Minutes'.

When we picked up this group's work, we decided to have scheduled meetings on Mondays to update each other on what we had been working on, to help with our gantt charts, and to plan what we were going to do before our practical session on Friday and then updating our Meeting Minutes document. We also followed their methodology of breaking work down into separate tasks for our sub-teams, with work split between the documentation and coding focused parts of our team.

The main Agile principles that applied to our group was the adaptive and collaborative approach which we naturally had in assessment 1 and decided to continue with that in assessment 2. It gave us the opportunity to help each other complete unfinished tasks more efficiently. One thing we improved on in assessment 2 was making sure we fulfilled all deliverables' requirements for this assessment as we went along as after looking at our feedback from assessment 1 there were parts we could improve on and talk about further. So for assessment 2 we focused on checking every part of the assessment that needed to be done and creating lists and points every week on what had to be completed. Other Agile frameworks which we researched about was Kanban which we used in our last assessment, however, without using Scrum this would have been harder to structure and track individual tasks as Kanban doesn't involve sprint planning and reviews to organise our meetings which can lead to lack of urgency and cause tasks to take longer than needed.

1.2 Development and Collaboration Tools

We have chosen to use IntelliJ IDEA as our team's IDE as we find it to have the most appealing, refined interface, as well as being customisable. Compared to other IDEs (such as Eclipse) IntelliJ has better support for GitHub desktop and Gradle builds, which are discussed below.

Originally, we explored the functionality of Eclipse - however, we found it was incompatible with GitHub desktop. When cloning a repository that has LibGDX imported, all of the classes are not recognised as a type; therefore, we chose to switch to IntelliJ to mitigate this issue.

To facilitate reliable version control for the duration of our project, we unanimously decided to use GitHub's desktop application, which we installed onto our local machines and connected to our individual GitHub accounts. An alternative to GitHub desktop is the Git command line interface, or interacting with the repository directly through the IDE using Personal Access Tokens (PAT). However, these processes are cumbersome and time-consuming, which makes it difficult for all members of our team to follow the same process and avoid user errors that threaten the reliability of our version control system. Consequently, we are using GitHub desktop to take advantage of its visual interface and user-friendly nature, which will be particularly beneficial to the members of our team who have no previous experience using Git. We created an organisation in assessment 1 called Team21Eng1 and we forked the new repository so we can start editing the code. We also edited the website using GitHub and distinguishing it by adding our team logo. We used GitHub workflow as part of the continuous integration of the assessment.

Furthermore, our decision to employ LibGDX as our Java game development framework stems from its robust capabilities and intuitive implementation. Through our initial research, we concluded that LibGDX offers straightforward concepts, exemplified by the "render()" and "dispose()" methods that have clear functionality. Additionally, it is extremely well-documented and has a comprehensive Wiki page that boasts support from setting up a development environment to adding in-game music. Alongside LibGDX, we are using Gradle - a build automation tool for software development. This aids us in compiling, linking and packaging the code into a single application that runs on various operating systems. Gradle can be seamlessly integrated into IntelliJ IDEA with the help of extensions, which makes our adapted IDE an ideal environment in which to create our project. When conducting our research we found that Libgdx was more compatible with 2D games compared to JMonkeyengine which offers better tools when creating 3D games which is not our goal for this project. There is also extensive documentation and tutorials regarding Libgdx which was a useful guide for us in the last assessment. Both groups used Libgdx therefore the implementation team can easily continue developing the game for assessment 2.

For building the game map, Tiled appears to be the easiest software to use. Due to Tiled's free and flexible interface, we decided to use it as our designated map editor and builder. There are many reasons that can justify why Tiled is suitable, starting with ease of use of its intuitive drag-and-drop functionality to build maps in sections. Tiled also proved to be very compatible with our project as LibGDX has a plethora of libraries to handle the .tmx file that maps are generated in. Overall, Tiled is robust, with countless useful features: multiple layers, custom properties, tileset animations, and object grouping, among others. Using Tiled allowed us to integrate the other groups map with their assets easily as we are familiar with it and we could edit their map to add the library building and create interiors, and the town map.

We ended up using the same collaboration and development tools in the first part of the assessment, meaning we could continue on using them for this part of the assessment too, however the reason for using IntelliJ was different as we didn't attempt to use Eclipse at all as our team was already familiar with IntelliJ previously

We used PlantUML to create our structural, behavioural and class diagrams as well as the Gantt chart. We can create different diagrams such as sequence and use case diagrams (reference them) and as you are editing the code you can see the diagram being changed which is an advantage. We can also integrate it with google docs easily and others can collaborate on a diagram. Before using this we used

draw.io to create the work breakdown structure as we were familiar with this software previously and wanted to create it as soon as possible.

2.1 Approach to Team Organisation

Our team chose to divide its members into two departments in order to accommodate the workload described in the project briefing.

The development team, made up of Kyla Kirilov, Ben Hayter-Dalgliesh, and Matthew Graham, is responsible for writing the code for our system. Managed by Kyla, this team is building the game from the elicited requirements, and meets regularly to negotiate and develop new ideas. The team works horizontally on different sections of the code, and collaborates with Kyla to integrate them into the system.

The documentation team, consisting of Hannah Thompson, Callum MacDonald and Chak Chiu Tsang, is assigned to write up the required documentation for the project. Each member of this team is responsible for maintaining and updating one or more of the deliverables, such that the division of marks is equal. For every deliverable, the team meets to review changes and evaluate the progress, with all members of the team adding their own relevant contributions.

The teams meet twice a week, both online and in-person, to discuss and demonstrate development and make necessary updates to the documentation, such as reviewing the risk assessment and auditing the requirements. Both teams display the progress they have made in that week, and collaborate to solve problems with the code or documentation.

This approach plays both to our individual strengths - ensuring that those more proficient in writing code or documentation are utilising their skills - and helps to avoid overcrowded collaboration on each deliverable. Similarly, it allows us to divide the workload evenly between all team members, taking into account the weighting of each task and deliverable.

We also split our group up into two teams, with our documentation team being made out of James Cretney, Lloyd Newton and Doaa Doukh, and our development team being made of Surbhi Lahoria, Sean Abong, Peter Beck and Isaac Ohara. Similar to Group 25, we meet twice a week to sync up work between us.

These teams were assigned so that everyone in the group was working to their strengths and slightly changed from our work in assessment 1, where people better realised what they were best at doing. Towards the end of the project, we realised we probably should have assigned more people to the development team as in a few times, we found a disconnect between the speed of work of the two groups. We also kept track of our meetings using a Meeting minutes document.

3.1 Systematic Plan for the Project: Key Tasks and Dates

Our initial meeting produced a high-level work breakdown, which can be found on our website ('Method Planning and Selection, Fig. 1'). This is an overview of the components required for each deliverable, broken down into atomic tasks that can be completed by an individual or team. The deliverables, labelled D1-D6, correspond to section 3.3.1 of the ENG1 Team Assessment document. By taking each deliverable and breaking it into a series of smaller tasks, we were able to generate a systematic plan for the following weeks, taking into account the dependencies of each task.

From this diagram, we created an initial Gantt chart that lays out the key tasks and their ownership. This chart, which can be found on our website ('Method Planning and Selection, Fig. 2') shows how the project will develop over the first week. Each week, we reviewed our progress during the previous week and updated the Gantt chart to plan for the next weeks' tasks. Weekly snapshots of the plan can be found on our website ('Method Selection and Planning, Fig.2-6').

In week 1, we focused on setting up the collaborative tools discussed in the previous sections, as well as researching methods, assets, and techniques that would be useful for our project. We also created a prototype of the game, implementing only a sprite moving around a map. Due to issues with our IDE choice and integrating LibGDX, as well as waiting for a client meeting, we postponed implementation by a week to ensure that we were fully prepared to begin writing the code.

Our client meeting took place in week 3, which meant that our requirements could not be fully elicited until this point. Once the meeting was concluded, the requirements were defined and the process of writing the documentation was fully implemented. Therefore, during week 2, when we were unable to begin writing the code, the documentation team prepared the website, risk assessment, and architecture documents, while the development team prototyped simple modules and constructed the game map. Despite not being able to properly start the implementation, our development team was eager to become as comfortable as possible with the chosen IDE and game engine.

In week 3, the player animation and movement were completed, finalising the first prototype of the game. Some assets were selected this week, along with communicating concept designs for the game and its appearance. This week went very smoothly, and no issues arose.

In week 4, after the requirements elicitation had concluded, we held another meeting to plan the implementation and architecture going forward. The development team leader noticed that the pace of implementation was not up to speed and that certain individuals in the team were unsure of what needed to be done. To resolve this, we created a checklist of key features of the system, using the requirements as a starting point and decomposing them into smaller, more manageable tasks that included self-constructed deadlines. Since the workload at this point was heavy on development, the documentation team assisted by working on the code alongside the other deliverables. This meant that progress on documentation briefly stalled, but since there were dependencies on the code, this approach worked well for our team. This led to the completion of the map, energy bar, and activity counters.

In week 5, the project focused on completing the code, ensuring it met the requirements and that the system worked as expected from start to finish. This week, our aim was to complete map collisions, music/SFX, and all GUIs. Everything went according to plan, except for the loss of one GUI's code during the merging of code versions through GitHub Desktop. This setback was unexpected; however, it was quickly resolved by having one of the developers, who had already completed their tasks for the

week, redo the code. During this week, it appeared that the team member responsible for creating diagrams with PlantUML was struggling a bit, due to many of the diagrams being reliant on the code, which was only slowly coming together. This issue was addressed by the development team discussing plans for code that would specifically aid in the creation of diagrams. Overall, the architecture was more behind than other sections of the project, but this was simply resolved through some motivation.

Week 6 was when we made the finishing touches to everything, especially cleaning up the code and adding comments so that whoever may take over can understand everything clearly. Along with implementing a game end screen, a simple minigame to simulate studying was added, and we created our own eating sound effect for in-game use. After creating an executable for the game, the workload this week proved to be a bit too much for everyone. This didn't go according to plan, and we should've discussed how to even out the workload among everyone in the week leading up to submission.

Finally all the deliverables, executables, and documentation were added to the website for submission.

In our first meeting with the new work, we decided to spend the whole session breaking down what needed to be done, while evaluating how the previous team had formatted their work and how we would build upon it. From looking through assessment 2, we created smaller tasks by breaking down the deliverables, which we put into our Work Breakdown diagram, which helped us visualise the work remaining and how we were progressing through the project. This diagram can be found on our website in the ‘method and planning diagrams’ document.

In further sessions, we created Gantt charts, which can also be found on our website in the same document, that layed out what the key tasks were each week and assigned their ownership. These show how we developed the project throughout our time working on it, while also letting us review our progress throughout the weeks, and help us evaluate how much work was left to be done.

We didn’t need a client meeting due to most of the work being set off well, we instead only needed a few questions asked to the client in person in one of our first sessions, where we asked a few basic questions revolving around how much we were meant to change the previous deliverables. This meant we could start work quickly, as we had everything laid out for us, as well as having discussed what our basic plan was before selecting a team’s project in week 7.

In week 9, we started the week by planning out our approach to the change report in the documentation team, while the development team decided to split their time between improving the game, while adding additional features, and adding the testing to all of the code. With our approach to the change report, we decided that we would start with the Method Selection and Planning and Functionality documents, before starting on the others in later weeks. The implementation team also looked at the feedback from Group 25 and identified any bugs in the game before starting to create tests.

In week 10, we experienced some difficulties with testing, specifically automated unit testing for the game features, as we found it difficult to come up with ways to test some of the more visual and game-affecting functions. After getting some advice and feedback from our GTA, we figured out how to get it working, and that it was causing a lot of other groups problems as well. We also started planning out our User Evaluation process, figuring out which documents we needed to print out and get signed, as well as our process for recording the Evaluations, eventually deciding on written records of the process, as meant we didn’t need to apply for approval from the university. We created interior maps and a town map which we didn’t implement due to time constraints. Also, we thought of adding more mini games as

they were an enjoyable aspect from the previous version of the game. Other parts of the development team also began working on the leaderboard and achievements/streaks which were part of the new requirements.

Week 11 was when we started deciding on our User Evaluation users, as well as finishing up testing fully. We also started on our Continuous Integration, and also started moving our testing onto Github as well. As well as that, we needed some clarification on some of our documentation from our stakeholder, with there being a worry that we had not filled the pages enough on the change document, as well as some advice towards the weekly workload and untimetabled meetings. Furthermore, we decided to designate the architecture document to one person from each team, as the development team had a more complete understanding of the new code that we had developed throughout assessment 2. We then fixed any coding issues that users had when playing the game on the ‘first-release’ branch and started to merge the corrected versions onto the main branch.

In week 12, due to not having a university scheduled meeting, we decided to find a time to finish all our last second corrections before submitting the final documentation. We finished up our documentation for the User Evaluation deliverable and updated the change report with the new architecture diagrams which took longer than expected as the code was being updated continuously after our user evaluation. Then we uploaded everything to our website and zipped everything into a folder and submitted it.