Method Selection and Planning

ENG1 Team 9

Jacob Dicken
Bertie Cartwright
William Croft
James Dovener
Henry Chan

This document has been modified for Assessment 2 to reflect the changes made by ENG1 Team 6:

- Hussain Alhabib
- Ellen Matthews
- Minnie Poon
- Jason Ruan
- Daniel Smith
- Owen Smith

Method Selection and Planning

Engineering Methods

Engineering Methodology

We chose to use an agile software engineering methodology as this allowed us the most flexibility as the project is completed and allows for individuals to dynamically contribute to the progress of the system. The advantages of using agile are quick deployment, emphasis on collaboration and the ability to adapt to changing requirements. Quick deployment is important due to the short time scale of the project, collaboration is important to ensure that skills from all members of the team are being utilised to maximum effect and the ability to adapt as requirements change is useful as the features that we choose to implement may change as the project unfolds. The disadvantages of using agile are a smaller emphasis on testing, smaller emphasis on documentation and risk of burn-out. The smaller emphasis on testing can be mitigated by focusing on code quality and adhering to style guides and best practices. Burn-out can be reduced by allocating work based on team strengths, varying tasks that members do and working in pairs on sprints. These factors make agile the most suitable methodology for our project.

One of the alternatives that we considered was a waterfall methodology. An advantage of a waterfall approach is that it is easy to use and manage. It also provides definitive structure to each phase of the project and a greater level of documentation is required for each phase. However, changes to requirements cannot be easily accommodated due to the linear nature of the waterfall model, software development takes a back seat until requirements and design phases are completed and gathering accurate requirements before any prototypes have been made can be difficult. These disadvantages lead to waterfall being less suitable for our project.

----------OUR EDITS----------

Once we selected Agile as our chosen methodology, we explored the options and types of Agile - in particular Scrum and Kanban. Scrum is more suitable for teams new to agile, and ensures team members have specific roles to enhance productivity and ensure multiple tasks are running in parallel. Whereas, Kanban is less suited to teams new to agile, and utilises a Kanban board with three columns; To Do, Doing and Done. After discussing, we settled on Scrum as our chosen type of Agile Methodology primarily due to its enhanced suitability for teams new to agile, and its Scrum board which is similar to the Kanban board and organises tasks based on progress - something we like as a team.

Source Control

GitHub - We chose github for version control as it allows us to remotely collaborate with each other. It allows individual team members to pull code from a centralised repository to work on a section of code before pushing it to a side branch to be reviewed and merged with the main branch of code. This allows for developers to work on code without affecting the

work of the other team members and code can be verified before committing it to the main branch, saving time by minimising errors in code.

We also use GitHub projects to keep track of progress throughout the development stage. This allows us to break down large tasks into issues, which can be assigned to individual team members and integrated with pull requests, which aligns with our agile development methodology.

An alternative that we considered for source control is Azure. Advantages of Azure are its integration with the Microsoft ecosystem including the VSCode IDE, which is a popular choice among developers. However as we are using Google Drive for some of our documentation and many of our team members are already familiar with GitHub that is the choice we went with.

Development Environment

We chose to use VSCode and IntelliJ in the implementation of our project. One reason for this is that both IDEs have support for Java compilation & debugging and version control through GitHub. IntelliJ has this built-in while VSCode has a selection of extensions that can be installed. As LibGDX projects generally use their own build scripts, the project itself is mostly agnostic to the IDE used, meaning group members can use their personal preference.

Team Communication

When deciding between software for team communication, the two popular options among our group were Slack and Discord. We chose to use Discord for this project as it is software every member of the group is familiar with. Additionally, we were able to use webhooks to create a channel that tracks GitHub activity - this is useful to see at a glance when issues are created or closed off, and when code reviews have been requested.

----------OUR EDITS----------

When deciding our preferred method of team communication, we considered two popular and industry-leading platforms; Slack and Microsoft Teams. After careful consideration, and analysis of their features and functionalities when using the free plan, we settled on Slack. We chose Slack over Microsoft Teams primarily due to its enhanced collaboration abilities including the ability to integrate Google Drive (our chosen File Management platform) and GitHub (our chosen Version Control platform) into our teams channels, and the ability to set up separate channels for each deliverable and other aspects of the project which helped to foster transparency and collaboration.

Frameworks

LibGDX - We chose libGDX as the framework for our game development. It comes with ample features for developing a 2D game. It is widely used and comes with extensive documentation, which means we can get started with a simple setup very quickly. Additionally, LibGDX is Apache 2 licensed, a highly permissive licence that allows the use of their code for any projects with no fees.

Team Organisation

Project Lead - Bertie

- Coordinating team (Arranging meetings)
- Overseeing that team members are completing their responsibilities
- Manage task board on github so that tasks aren't left out

Documentation - Henry

- Ensuring code documentation is complete
- Ensuring that documentation will allow new team to take over project
- Create developer README so that developers understand the source code
- Project documentation

Source Control / Project structure - Jacob

- Maintaining the github repo (branching and merging onto main branch)
- Make sure that tasks within the project are not missed
- Code reviews
- Contribute to weekly plans regarding implementation tasks

Writeup - Will

- Making sure the write up is complete
- Summarise team decisions to add to write up
- Proofreading
- Ensure that any methods/programs used have been added to the write up

Quality Assurance - James

- Ensure that code is high quality and readable
- Debugging code
- Ensure that game is fun and engaging
- Document known issues with code

We chose to organise the team in this way to allow for clear definition of responsibilities with only small overlaps to avoid confusion as to who is responsible for certain tasks. This allows us to work more efficiently, by reducing time spent on delegating tasks. Roles were chosen based on our personal interests and skills to maximise participation and mitigate the risk of burnout. This approach is suitable for developing a small 2D game due to the relatively small scope of each part of the project, meaning one team member is able to lead each main area.

----------OUR EDITS----------

As a team that utilised the Agile methodology for development, we chose a deliverable-based approach to team organisation. Each team member is assigned as a Lead and is responsible for one of the six project deliverables. This approach helps to promote ownership and accountability, and enables each team member to focus on a deliverable that aligns with their expertise or interest - increasing productivity and enhancing the team's progress.

To further support the team's efficiency, we assigned a secondary role to each team member which addresses various operational needs within the project. Below is an overview of these roles

1. **Project Lead:** Responsible for overseeing the project's timeline and ensuring that all deliverables stay on schedule as per the Gantt Chart. They check in on team members' progress at the beginning of each meeting and help maintain focus on tasks.
2. **Head Developer:** Manages code production, coordinates feature integration, and establishes coding standards and formatting. They ensure that development aligns with the team's chosen methodology and oversees the technical aspects of the project.
3. **Quality Assurance:** Handles testing to ensure the product meets agreed-upon requirements, assists the Head Developer with identifying and fixing bugs, and ensures the quality of features within the product.
4. **Report Editor:** Oversee the finalisation of all written reports and deliverable documents, ensuring they meet the standards outlined in the assessment brief.
5. **Secretary:** This role is divided into two different positions:
    a. **Meeting Secretary:** Manages meeting documentation, prepares agenda, and takes notes. They also organise the team's working directory to ensure documents are up-to-date and easy to find.
    b. **Logistics and Communications Secretary:** Coordinates meeting logistics, including booking rooms and arranging schedules, and manages communication with the client by relaying project updates and feedback.

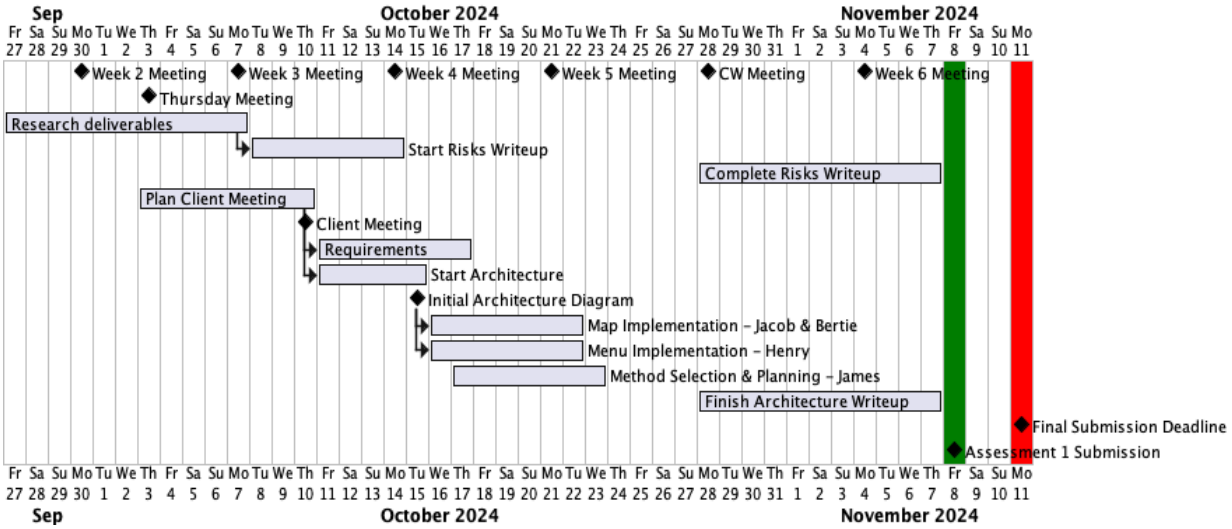
Our Team

Hussain Alhabib Software Testing Lead *Quality Assurance*	**Ellen Matthews** User Evaluation Lead *Meetings Secretary*	**Minnie Poon** Change Report Lead *Report Editor*
Jason Ruan Continuous Integration Lead *Project Lead*	**Daniel Smith** Website Lead *Logistics and Communications Secretary*	**Owen Smith** Implementation Lead *Head Developer*

Reasoning behind certain roles:

- As the person responsible for Quality Assurance, we felt Hussain would be best suited to be the Software Testing lead.
- As an aspiring Data Scientist, we felt Ellen should take the lead on the Evaluation section.
- As the person most familiar with LibGDX and Game Development, we felt Owen should take the lead on everything coding related.

Plan for Key Tasks and Deadlines



Note: Task - priority (Assignee)

Research Deliverables - High (All)
It is necessary for all members of the group to understand the deliverables to ensure that everyone is able to complete their sections correctly.
Plan Client Meeting - Medium (All)
Dependencies: Research Deliverables
The client meeting is to be planned to ensure that we use the time effectively and are able to elicit requirements from the client.
Start Risk Writeup - Medium (William)
Dependencies: Research Deliverables, Plan Client Meeting
The risk writeup will allow us to mitigate risk of failure throughout the project so that we ensure our success
Client Meeting - High (James)
Dependencies: Client Meeting Plan
The client meeting provides clarification on any confusions we had on the requirements and make sure whether a feature is necessary
Elicit Requirements - High (Will)
Dependencies: Client Meeting
Requirements write up allow us to manage requirements made by the clients thoroughly. As every demand is described in detail and assigned a priority, it allows us to easily plan and assign requirements to different teammates based on their strengths and preferences.
Start Architecture Design - High (Jacob & Bertie)
Dependencies: Elicit Requirements
The architecture plan allows us to decide on the themes of the game and the basic structure of the code. We focused on constructing a modulable structure as it provides easy modifications of codes and features, and allows team members to work on different parts of the code without a thorough knowledge of unrelated sections.

Create initial architecture diagram - High (Bertie)
Dependencies: Start Architecture Design
This will help us start work on implementation as if we don't have an architecture diagram we do not know how each class of our project should interact with each other. Without this, classes written by different members may not interact correctly.

Map implementation - High (Jacob & Bertie)
Dependencies: Create initial architecture diagram
The map will be made first as it is fundamental to the game and will contain all other entities that are added to the map.
Menu Implementation - High (Henry)
Dependencies: Map Implementation
The menu will be made after the map so that it is able to call for the map to be rendered once the play button is pressed
Method Selection & Planning - (James)
Dependencies: Elicit Requirements, Menu Implementation, Map Implementation
The method selection and planning stage of the write up will be left until the end of the project as we will be using github projects and google drive to keep track of team efforts
Complete Risks Write Up - High (Will)
Dependencies: Start risk write up
The risk writeup is to be completed at the end of the project as we want to be able to update the risk assessment if any issues come to light which were not previously anticipated.
Finish Architecture Write Up - High (Bertie)
Dependencies: Create initial architecture diagram
The architecture writeup will be completed at the end of the project as there may be tweaks made to method use and the way in which different classes may interact with each other.

Note: Due to the nature of these tasks and progress being dependent on completion of previous tasks, priorities are relatively high for all of these key tasks.

WEEKLY PLANS

How the plan changed throughout the project

Around 1 week into the implementation phase, Bertie joined the implementation side of the project to assist with writing the rendering code due to having previous experience with similar projects. This enabled us to move forward more quickly with development, meaning we would have sufficient time to prioritise code quality & documentation towards the end of the project. This led to Will taking over some of the writeup roles, particularly on the risks section to compensate for Bertie spending more time on implementation.

Overall, our efforts shifted towards implementation as the project progressed, with the most combined focus being towards the middle of the project - around weeks 4-5 and into Consolidation Week. It became clear that implementation would require significant effort to build a high quality product that can be easily extended and developed later. Towards the end of the project, Jacob took on a more supervisory role relating to development, with a particular focus on ensuring our code is compliant with javadoc conventions and the Google style guide.

----------OUR EDITS----------

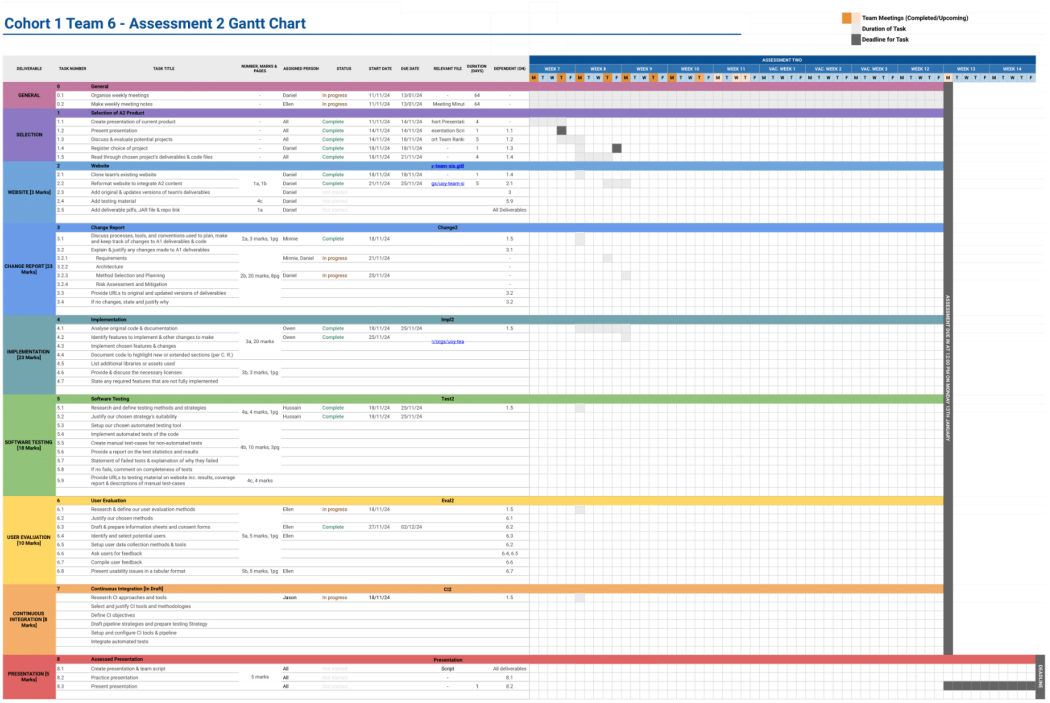


Figure: A screenshot of our Gantt Chart for Assessment 2 taken during Week 10.

Similar to Assessment 1, we utilised a Gantt Chart, the style and design of which can be seen above. Our chosen structure allowed us to ensure task transparency and ownership of tasks whilst also giving a clear timeline for the project to the team's members.
